ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО 13584-31— 2010

Системы промышленной автоматизации и интеграция

БИБЛИОТЕКА ДЕТАЛЕЙ

Часть 31

Ресурсы реализации. Интерфейс геометрического программирования

ISO 13584-31:1999
Industrial automation systems and integration — Parts library —
Part 31: Implementation resources: Geometric programming interface
(IDT)

Издание официальное

Москва
Стандартинформ
2016

gost.e-ecolog.ru

Предисловие

Цели и принципы стандартизации в Российской Федерации установлены Федеральным законом от 27 декабря 2002 г. № 184-ФЗ «О техническом регулировании», а правила применения национальных стандартов Российской Федерации — ГОСТ Р 1.0—2004 «Стандартизация в Российской Федерации. Основные положения»

Сведения о стандарте

1 ПОДГОТОВЛЕН Научно-техническим центром ИНТЕК на основе собственного аутентичного перевода на русский язык стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 100 «Стратегический и инновационный менеджмент»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 21 декабря 2010 г. № 874-ст

4 Настоящий стандарт идентичен международному стандарту ИСО 13584-31:1999 «Системы промышленной автоматизации и интеграция. Библиотека деталей. Часть 31. Ресурсы реализации. Интерфейс геометрического программирования» (ISO 13584-31:1999 «Industrial automation systems and integration — Parts library — Part 31: Implementation resources: Geometric programming interface»).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты Российской Федерации, сведения о которых приведены в дополнительном приложении ДА

5 ВВЕДЕН ВПЕРВЫЕ

*Правила применения настоящего стандарта установлены в ГОСТ Р 1.0—2012 (раздел 8). Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (*gost.e-ecolog.ru

© Стандартинформ, 2016

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения . . . 1
2 Нормативные ссылки . . . 1
3 Термины, определения, обозначения и сокращения . . . 2
3.1 Термины и определения . . . 2
3.2 Прочие термины и определения . . . 2
3.3 Сокращения . . . 3
4 Основные положения . . . 3
4.1 Требования к параметрическим описаниям . . . 3
4.2 Формат обмена параметрических описаний формы . . . 4
4.3 Внутреннее представление данных, созданных в приемной системе CAD . . . 4
4.4 Библиотека поставщика и ответственность пользователя LMS . . . 4
4.5 Совместимость . . . 4
4.6 Точность геометрических построений . . . 5
5 Интерфейс . . . 5
5.1 Спецификация и соответствие . . . 5
5.1.1 Допустимые уровни реализации . . . 5
5.1.2 Моделирование отсутствующих сущностей . . . 6
5.2 Таблицы интерфейса . . . 6
5.3 Создание данных модели продукта . . . 6
5.3.1 Ссылочные координатные системы видов (OVC) . . . 6
5.3.2 Геометрические единицы OVC . . . 7
5.3.3 Содержимое вида . . . 8
5.3.4 Временная база данных . . . 8
5.3.5 Процесс удаления невидимых линий . . . 8
5.3.6 Процесс представления . . . 10
5.4 Структура сущностей . . . 10
5.4.1 Структура группы в TDB . . . 10
5.4.2 Структура сущностей, направляемых в CAD . . . 11
5.5 Имя геометрической или структурированной сущности . . . 11
5.6 Координатная система и ее преобразование . . . 12
5.7 Состояние ошибки интерфейса . . . 12
5.8 Исправление ошибок . . . 12
5.8.1 Методология исправления ошибок . . . 12
5.8.2 Сообщения об ошибках . . . 13
6 Логическая модель целевой моделирующей системы . . . 16
6.1 Элемент геометрического представления . . . 16
6.1.1 Схема API_ABSTRACT_SCHEMA . . . 18
6.1.2 Определения типов схемы API_ABSTRACT_SCHEMA: основные понятия описания продукта и его поддержки . . . 19
6.1.3 Определения типов схемы API_ABSTRACT_SCHEMA: геометрические и топологические представления . . . 21
6.1.4 Определения типов схемы API_ABSTRACT_SCHEMA: геометрические модели . . . 23
6.1.5 Определения типов схемы API_ABSTRACT_SCHEMA: особые типы структурирования интерфейса прикладного программирования . . . 24
6.1.6 Определения сущностей схемы API_ABSTRACT_SCHEMA: основные понятия описания и поддержки продукта . . . 25
6.1.7 Определения сущностей схемы API_ABSTRACT_SCHEMA: структуры представлений . . . 27
6.1.8 Определения сущностей схемы API_ABSTRACT_SCHEMA: структуры геометрических представлений . . . 31
6.1.9 Определения сущностей схемы API_ABSTRACT_SCHEMA: геометрические математические сущности . . . 33
6.1.10 Определения сущностей схемы API_ABSTRACT_SCHEMA: сущности геометрических кривых . . . 38

6.1.11 Определения сущностей схемы API_ABSTRACT_SCHEMA: геометрические конические сущности . . . 46
6.1.12 Определения сущностей схемы API_ABSTRACT_SCHEMA. базовые кривые интерфейса прикладного программирования . . . 51
6.1.13 Определения сущностей схемы API_ABSTRACT_SCHEMA. дуги конических кривых интерфейса прикладного программирования . . . 52
6.1.14 Определения сущностей схемы API_ABSTRACT_SCHEMA: сущности кривых . . . 55
6.1.15 Определения сущностей схемы API_ABSTRACT_SCHEMA: заполненные области . . . 58
6.1.16 Определения сущностей схемы API_ABSTRACT_SCHEMA: сущности геометрических поверхностей . . . 60
6.1.17 Определения сущностей схемы API_ABSTRACT_SCHEMA: сущности поверхностей интерфейса прикладного программирования . . . 63
6.1.18 Определения сущностей API_ABSTRACT_SCHEMA: сущности геометрических тел . . . 65
6.1.19 Определения сущностей схемы API_ABSTRACT_SCHEMA: сущности структурирования интерфейса прикладного программирования . . . 72
6.2 Визуализация элементов геометрического представления . . . 74
6.2.1 Определения типов схемы API_ABSTRACT_SCHEMA: визуальное представление . . . 75
6.2.2 Определения типов схемы API_ABSTRACT_SCHEMA: типы визуального представления интерфейса прикладного программирования . . . 77
6.2.3 Определения сущностей схемы API_ABSTRACT_SCHEMA: визуальное представление . . . 77
6.2.4 Определения сущностей схемы API_ABSTRACT_SCHEMA: внешне определенные стили визуального представления . . . 84
6.2.5 Определения сущностей схемы API_ABSTRACT_SCHEMA: предварительно определенные стили визуального представления . . . 88
6.3 Определения функций схемы API_ABSTRACT_SCHEMA. . . . 91
6.3.1 Определения функций схемы API_ABSTRACT_SCHEMA: геометрические и топологические представления . . . 91
6.3.2 Определения функций схемы API_ABSTRACT_SCHEMA: поддержка ресурсов . . . 103
6.3.3 Определения функций схемы API_ABSTRACT_SCHEMA: структуры представлений. . . . 104
6.3.4 Определения функций схемы API_ABSTRACT_SCHEMA: функции интерфейса прикладного программирования . . . 107
6.4 Глобальные правила схемы API_ABSTRACT_SCHEMA . . . 112
6.4.1 Правило unique_shape_representation . . . 112
7 Функциональные спецификации интерфейса . . . 113
7.1 Соглашения об обозначениях . . . 113
7.1.1 Представления функций . . . 113
7.1.2 Представления типов данных. . . . 114
7.1.3 Имена сущностей и аббревиатуры. . . . 114
7.1.4 Имена функций . . . 115
7.2 Логическое описание функций интерфейса и привязки языка FORTRAN. . . . 116
8 Таблицы интерфейса. . . . 116
8.1 Таблица описаний интерфейса . . . 116
8.2 Таблица статуса интерфейса. . . . 116
9 Размерности реализации интерфейса. . . . 117
9.1 Минимальные размерности буферов интерфейса и структурированных типов данных . . . 117
Приложение А (справочное) Логическое описание функций интерфейса и привязки языка FORTRAN. . . . 119
Приложение В (справочное) Регистрация информационного объекта . . . 294
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации . . . 295
Библиография . . . 296

Введение

Целью комплекса стандартов ИСО 13584 является создание эффективного механизма передачи данных библиотеки деталей вне зависимости от выбора компьютерного приложения, использующего указанную библиотеку. Установленные стандартом описания облегчают обмен файлами деталей, создают основу для последующего применения данных библиотеки деталей и их совместного использования.

ИСО 13584 включает в себя: основные описания, логический ресурс, используемые данные, описания методологии, протоколы обмена видами, словари ссылок. Описания частей приведены в ГОСТ Р ИСО 13584-1—2006 «Системы промышленной автоматизации и интеграция. Библиотека деталей. Часть 1. Обзор и основные принципы». В настоящем стандарте рассмотрены ресурсы реализации.

В настоящем стандарте представлено описание интерфейса, позволяющего создавать модели продуктов внутри пользовательской системы с помощью прикладных программ, независимых от целевой пользовательской системы. Интерфейс может быть использован вне контекста стандартных данных библиотек деталей. Он позволяет разрабатывать прикладные программы, не зависящие от целевых CAD. В контексте ИСО 10303 настоящий интерфейс может быть применен на верхнем уровне стандартного интерфейса доступа к данным (SDAI) для геометрических построений с учетом имеющихся ограничений. Процесс создания данных модели продукта представляет собой прикладную программу (предлагаемую поставщиком библиотеки деталей), создающую геометрическую модель внутрь пользовательской системы. Данный интерфейс гарантирует ее независимость от целевой пользовательской системы.

ИСО 13584 разработан Техническим комитетом ИСО/ТК 184 «Системы автоматизации производства и их интеграция» и подкомитетом ПК 4 «Технические данные и языки программирования глобального производства».

ИСО 13584 состоит из следующих частей:

- часть 1. Обзор и основные принципы;
- часть 10. Общее описание. Концептуальная модель библиотеки деталей;
- часть 20. Логический ресурс. Логическая модель выражений;
- часть 24. Логический ресурс. Логическая модель библиотеки поставщика;
- часть 26. Логический ресурс. Идентификация поставщика информации;
- часть 31. Ресурсы реализации. Интерфейс геометрического программирования;
- часть 42. Методология описания. Методология структурирования семейства деталей;
- часть 101. Протокол обмена видами. Геометрический протокол обмена видами в параметрической программе;
- часть 102. Протокол обмена видами. Протокол обмена видами в спецификации соответствия ИСО 10303.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы промышленной автоматизации и интеграция

БИБЛИОТЕКА ДЕТАЛЕЙ

Часть 31

Ресурсы реализации. Интерфейс геометрического программирования

Industrial automation systems and integration.
Parts library. Part 31. Implementation resources. Geometric programming interface

Дата введения 2011—09—01

1 Область применения

Настоящий стандарт устанавливает интерфейс прикладного программирования, с помощью которого прикладная программа генерирует геометрические модели независимо от целевой пользовательской системы. Данный интерфейс обеспечивает мобильность прикладных программ, описывающих представления параметрических форм семейств деталей в соответствии с комплексом стандартов ИСО 13584.

Настоящий стандарт распространяется на:

- программы, генерирующие геометрические представления внутри моделирующих систем, не зависящих от целевых систем;
- программы, описывающие геометрические представления, созданные посредством геометрических определений с учетом ограничений;
- программы, структурирующие геометрические представления, созданные независимо от целевой системы;
- программы, описывающие атрибуты стиля воспроизведения для символической визуализации созданных представлений;
- программы, поддерживающие стандарты технического черчения для представления форм, включая 2D-механизм для невидимых линий;

Настоящий стандарт не распространяется на:

- точное управление изображением на мониторе приемного устройства;
- точное определение данных, создаваемых приемной системой;
- хранение параметрической модели в приемной системе.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты, которые необходимо учитывать при использовании настоящего стандарта. В случае ссылок на документы, у которых указана дата утверждения, необходимо пользоваться только указанной редакцией. В случае когда дата утверждения не приведена, следует пользоваться последней редакцией ссылочных документов, включая любые поправки и изменения к ним:

ИСО 128:1982 Технические чертежи. Общие принципы воспроизведения (ISO 128:1982, Technical drawings — General principles of presentation)

ИСО 1539:1991 Информационные технологии. Языки программирования. FORTRAN (ISO 1539:1991, Information technology — Programming languages — FORTRAN)

Издание официальное

ИСО/МЭК 8824-1 Информационные технологии. Абстрактная синтаксическая нотация версии один (ASN. 1). Часть 1. Спецификация базовой нотации (ISO/IEC 8824-1, Information technology — Abstract Syntax Notation One (ASN. 1) — Part 1: Specification of basic notation)

ИСО 10303-11:1994 Системы промышленной автоматизации и их интеграция. Представление данных о продукции и обмен данными. Часть 11. Методы описания: справочное руководство по языку EXPRESS (ISO 10303-11:1994, Industrial automation systems — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual)

ИСО 10303-41:1994 Системы промышленной автоматизации и их интеграция. Представление данных о продукции и обмен данными. Часть 41. Основные понятия описания и поддержки продукта (ISO 10303-41:1994, Industrial automation systems — Product data representation and exchange — Part 41: Fundamentals of product description and support)

ИСО 10303-42:1994 Системы промышленной автоматизации и их интеграция. Представление данных о продукции и обмен данными. Часть 42. Интегрированный ресурс: геометрические и топологические представления (ISO 10303-42:1994, Industrial automation systems — Product data representation and exchange — Part 42: Integrated resources: Geometric and topological representation)

ИСО 10303-43:1994 Системы промышленной автоматизации и их интеграция. Представление данных о продукции и обмен данными. Часть 43. Интегрированный ресурс: структуры представлений (ISO 10303-43:1994, Industrial automation systems — Product data representation and exchange — Part 43: Integrated resources: Representation structures)

ИСО 10303-46:1994 Системы промышленной автоматизации и их интеграция. Представление данных о продукции и обмен данными. Часть 46. Интегрированный групповой ресурс: визуальное представление (ISO 10303-46:1994, Industrial automation systems — Product data representation and exchange — Part 46: Integrated generic resources: Visual presentation)

3 Термины, определения, обозначения и сокращения

3.1 Термины и определения

В настоящем стандарте использованы термины по ИСО 13584-10:

- абстрактная деталь (abstract part);
- система автоматизированного проектирования (computer aided design system);
- язык EXPRESS;
- функциональный вид (functional view);
- система управления библиотекой (library management system, LMS);
- поставщик библиотеки (library supplier);
- деталь (part);
- библиотека деталей (parts library);
- поставщик деталей (parts supplier);
- продукт (product);
- данные продукта (product data);
- программа (program);
- представление детали (representation of a part);
- поставщик (supplier);
- деталь поставщика (supplier part);
- библиотека поставщика (supplier library);
- структура (structure);
- пользователь (user);
- пользователь библиотеки (user library);
- вид (view);
- переменная управления видом (view control variable);
- протокол обмена видами (view exchange protocol).

3.2 Прочие термины и определения

В настоящем стандарте использованы следующие термины с соответствующими определениями.

3.2.1 **интерфейс прикладного программирования** (Application Programming Interface; API): Множество функций, запускаемых из одной программы с помощью заданного синтаксиса, определенного в одной привязке.

3.2.2 **привязка языка программирования** (binding): Описание заданного синтаксиса, используемого в особом языке программирования для запуска различных функций, составляющих интерфейс прикладного программирования.

3.2.3 **локальная координатная система** (Local Coordinate System; LCS): Ортогональная правосторонняя координатная система, используемая для ориентации и расположения в пространстве геометрических сущностей. Локальная координатная система моделируется сущностью axis2_placement.

3.2.4 **параметр** (parameter): Переменная с описанными именем и типом значений.

3.2.5 **параметрическая модель формы** [parametric (shape) model]: Выражение параметрической формы посредством модели данных.

3.2.6 **параметрическая программа формы** [parametric (shape) program]: Выражение параметрической формы посредством программы, ссылающейся на API (интерфейс прикладного программирования).

3.2.7 **параметрическая форма** (parametric shape): Общее описание семейства родственных форм и множества параметров. Параметрическая форма дает описание особой функции, действующей из области параметров на множество форм.

3.2.8 **временная база данных** (temporary database): Механизм, позволяющий хранить данные о конструкции или промежуточные данные перед передачей их в соответствующую систему CAD.

3.3 Сокращения

В настоящем стандарте использованы следующие обозначения и сокращения:

2D — двухмерный (two Dimensional);
3D — трехмерный (three Dimensional);
API — интерфейс прикладного программирования (Application Programming Interface);
CAD — автоматизированное проектирование (Computer Aided Design);
EPS — допуск на значение величины (Epsilon), см. 4.6;
HLI — включение невидимых линий (Hidden Line Involved), см. 5.3.5;
LCS — локальная координатная система (Local Coordinate System);
LMS — система управления библиотекой (Library Management System);
MAX — максимальное значение величины (Maximal value), см. 4.6;
OVC — координатная система вида объекта (Object View Coordinate system), см. 5.3.1;
SDAI — стандартный интерфейс доступа к данным (Standard Data Access Interface);
TDB — временная база данных (Temporary Data Base), см. 5.3.4.

4 Основные положения

4.1 Требования к параметрическим описаниям

1) ИСО 13584 устанавливает механизм глобального описания форм различных деталей, принадлежащих одному семейству деталей библиотеки.

Пример 1 — В ИСО 4014 [1] приведены описания тысяч различных видов болтов. Описание формы каждого вида болта в отдельности нецелесообразно.

2) Каждое глобальное описание формы должно быть ассоциировано со множеством численных, строчных или булевых параметров, область значений которых характеризует каждую деталь рассматриваемого семейства. Механизм генерации каждой заданной формы (вне ее глобального описания с помощью особого множества *v*-значений параметров) должен быть детерминированным. Он определяет частную функцию *f*, действующую из области значений параметров *D* на множество форм *S*:

$f : D \rightarrow S$; $s = f(v)$.

Данное описание называется параметрической формой.

Пример 2 — Глобальное описание двухмерного изображения сверху различных видов болтов, соответствующих требованиям ИСО 4014, зависит от двух действительных параметров L и D. Для каждой пары допустимых значений (l, d) параметров L и D рассматриваемый механизм генерирует детерминированную уникальную форму.

3) Параметрическая форма описывается графическими взаимосвязями пользователя. Иначе говоря, рассматриваемый механизм дает описания геометрии с учетом ограничений. Вычислитель ограничений является частью данного механизма.

4.2 Формат обмена параметрических описаний формы

1. Программа, ссылающаяся на интерфейс прикладного программирования, может быть использована для обмена глобальных описаний формы, удовлетворяющих требованиям раздела 4.1. Интерфейс прикладного программирования описывает геометрические функции с учетом ограничений. Структура программного управления описывает глобальную функцию. Интерфейс прикладного программирования приемной системы представляет собой вычислитель геометрических функций с учетом ограничений. Указанная программа называется параметрической.

2. Допущение: используемая технология автоматизированного проектирования предусматривает генерацию параметрических форм в терминах параметрических программ, основанных на использовании интерфейса прикладного программирования с учетом стандартных ограничений на основе описания рассматриваемого семейства форм в диалоговом режиме с учетом особенностей системы.

Примечание — Это допущение демонстрирует отличие между форматом обмена, описанным в настоящем стандарте (программой на языке FORTRAN, обращающейся к стандартному интерфейсу прикладного программирования), и вычислительной средой, используемой для создания указанного описания (например, диалоговой графической системой, такой как параметрическая система CAD).

4.3 Внутреннее представление данных, созданных в приемной системе CAD

Спецификация интерфейса должна:

- быть достаточно точной для описания формы деталей их поставщиком;
- исключать какие-либо спецификации реализаций для обеспечения мобильности устройства моделирования системы CAD.

В настоящем стандарте эти две цели достигаются путем описания логической модели целевой моделирующей системы. Настоящая логическая модель определена как информационная модель на языке EXPRESS. Каждая функция интерфейса описана путем ссылки на эту логическую модель.

4.4 Библиотека поставщика и ответственность пользователя LMS

1. Если деталь использована в некотором продукте, то представление формы детали и воспроизведение данной формы должны быть выполнены LMS и направлены в геометрическую моделирующую систему.

Пример — Если винт выбирается из LMS пользователем для вставки в чертеж при работе в системе CAD, то чертеж данного винта должен иметь на экране заданный цвет и заданную толщину линий в соответствии с выбранным представлением.

2. Библиотека может содержать геометрические описания разных поставщиков. Такая библиотека используется в контексте различных приложений. Указанный интерфейс позволяет поставщику библиотеки давать описания формы деталей, а пользователю библиотеки гарантирует сохранение уровня их воспроизведения. В настоящем стандарте для выполнения поставленной цели разрешается логическое управление поставщиком детали при воспроизведении ее формы (например, выбор поименованного стиля кривой). При этом:

- посредством некоторого нестандартного процесса инициализации интерфейса пользователь LMS может создавать полное описание всего аспекта воспроизведения (например, толщины линий, их типа и цвета), соответствующего каждому логически определенному стилю;
- форма, генерируемая LMS, представляется на экране в соответствии с текущей визуализацией моделирующей системы.

4.5 Совместимость

1. Элементы представления, созданные внутри модели данных продукта с помощью интерфейса, описанного в настоящем стандарте, подлежат обмену с помощью файла обмена, удовлетворяющего требованиям ИСО 10303. Все атрибуты сущности, которые не могут быть описаны поставщиком библиотеки, должны быть ограничены данной сущностью с помощью интерфейса. При этом учитывается процесс инициализации интерфейса, выполняемый пользователем библиотеки.

2. Если геометрическая моделирующая система поддерживает интерфейс SDAI [2], удовлетворяющий требованиям ИСО 10303-22, то интерфейс, описанный в настоящем стандарте, должен быть использован как аппликативный слой на верхнем уровне интерфейса SDAI.

Аппликативный слой содержит:

- решающую программу (далее — решатель) для вычисления геометрических сущностей с учетом ограничений;
- таблицу значений по умолчанию для атрибутов, установленных пользователем LMS, которая должна быть ограничена для каждой сущности, созданной с помощью SDAI.

Пример — Если дуга заданного радиуса стиля plain_solid_line должна касаться двух линий, то аппликативный слой содержит решатель, который вычисляет касательную окружность, ее параметры настройки и таблицы, содержащие точные значения толщины и цвета соответствующей плоской сплошной линии.

4.6 Точность геометрических построений

Различные моделирующие системы обеспечивают различную числовую точность построений. Необходимо:

1) гарантировать, чтобы поставщик программ действовал надлежащим образом при каждой «корректной» реализации интерфейса;

2) гарантировать, что реализация интерфейса надлежащим образом обрабатывает «корректную» программу поставщика.

В настоящем стандарте вышеуказанные цели достигаются путем определения ссылочных числовых границ различных мер, включенных в геометрические определения сущностей.

Определяют три ссылочные числовые границы:

1) EPS — это минимальное значение меры, включенной в меру геометрической сущностью.

Пример 1 — Прикладная программа не может задавать отрезок длиной меньше EPS.

2) MAX — максимальное значение меры, включенной в меру геометрической сущностью.

Пример 2 — Прикладная программа не может задавать дугу окружности радиусом более MAX.

3) ZERO_value (нулевое значение) — максимальное значение (математически вычисленного) расстояния между двумя точками, которые считаются совпадающими.

Пример 3 — Прикладная программа не может задавать контур (то есть замкнутую комбинированную кривую composite_curve), для которого расстояние между конечной точкой предшествующего сегмента данной комбинированной кривой composite_curve_segment и начальной точкой последующего сегмента данной кривой превышает нулевое значение ZERO_value.

Все указанные ссылочные числовые границы созданного геометрического представления масштабируют:

1) единицы длины вида *view_length_unit* масштабируют с помощью масштабного фактора *view_length_scale_factor* меры длины *length_measure*;

2) единицы угла вида *view_angle_unit* масштабируют с помощью меры плоского угла *plane_angle_measure*.

В настоящем стандарте для ссылочных числовых границ установлены следующие значения:

1) EPS = 10^{-3};

2) MAX = 10^{4};

3) ZERO_value = 10^{-6}.

Программа, соответствующая требованиям настоящего стандарта, также должна удовлетворять ограничениям, определенным (для каждой геометрической сущности) путем ссылки на указанные числовые границы. Интерфейс, соответствующий требованиям настоящего стандарта, должен обрабатывать программы, также соответствующие требованиям настоящего стандарта.

5 Интерфейс

5.1 Спецификация и соответствие

5.1.1 Допустимые уровни реализации

В настоящем стандарте приведено описание трех уровней реализации в соответствии с установленной геометрической мощностью интерфейса, имеющей значения: 2D-, 3D-кривая, тело. Значения

мощности нумеруются: 1, 2 и 3. Любой интерфейс *i*-го уровня мощности должен содержать все функции *j*-го уровня мощности для $j < i$. Таким образом, он может создавать виды *j*-го уровня геометрической мощности *geometrical_power_level*, если указанный уровень мощности установлен посредством инициализации вида. Для любой геометрической мощности интерфейса вид может быть также создан для уровня геометрической мощности *geometrical_power_level*, равного 0.

Определены три уровня реализации интерфейса (от 1 до 3). Все функции классифицированы в соответствии с данными уровнями. Рассматриваемая реализация должна обеспечивать (для выбранного приложения) все функции уровня, которому она принадлежит. Доступ к уровню интерфейса обеспечивается с помощью функции запроса.

5.1.2 Моделирование отсутствующих сущностей

Все рассматриваемые сущности для каждого уровня интерфейса в соответствии с настоящим стандартом должны быть концептуально реализованы во временных базах данных. Если некоторые сущности не определены в целевых системах моделирования продуктов, то они моделируются с помощью прочих доступных сущностей. В настоящем стандарте данный процесс моделирования описан для каждой сущности.

5.2 Таблицы интерфейса

Текущие характеристики интерфейса хранятся в таблицах интерфейса. Значения указанных элементов таблиц запрашиваются прикладными программами с помощью функций запроса, описанных в настоящем стандарте, содержащем две таблицы:

1) таблица описаний интерфейса представляет все постоянные характеристики интерфейса (например, уровень интерфейса *interface_level*, характеристики невидимых линий *hidden_line_capability* и т. д.). Указанные значения запрашиваются прикладной программой, но не могут быть ею изменены. Эти значения зависят от реализации;

2) таблица статуса интерфейса представляет значения модальных переменных (например, атрибуты визуализации). Начальное значение указанной переменной определено в настоящем стандарте либо как зависимое от вида (устанавливается в процессе инициализации интерфейса), либо как принимающее особые значения. Значения переменных таблицы статуса интерфейса запрашиваются прикладной программой. Они могут быть изменены прикладной программой за исключением переменных, зависящих от вида (например, единица длины вида *view_length_unit*, атрибут невидимых линий *hidden_line* и т. д.).

Примечание — В первой версии настоящего стандарта уровень 3 соответствует созданию тел с неявной топологией (сущности твердого тела, очерчивание, булевы операции). Он известен как уровень «конструктивной блочной геометрии». В более поздних версиях создание явных топологических элементов (вершин, кромок, граней и т. д.) может быть представлено как уровень 4. Он известен как уровень «B-Rep» (уровень представления границ).

Описание содержимого таблиц интерфейса приведено в разделе 8.

5.3 Создание данных модели продукта

В настоящем подразделе приведена концепция создания геометрической модели внутри геометрической моделирующей системы с помощью прикладного программирования. В контексте библиотек деталей, соответствующих требованиям ИСО 13584, данный подраздел предназначен для разработчиков программного обеспечения библиотек деталей, создающих данные модели детали внутри некоторой системы CAD.

Если функциональный вид выбирается в LMS, то рассматривают программу поставщика детали. Данный функциональный вид связан с экземпляром детали (объекта), описание которого должно быть предоставлено поставщиком детали.

Программа поставщика детали строит заданный функциональный вид с помощью функций интерфейса.

5.3.1 Ссылочные координатные системы видов (OVC)

Функциональный вид, созданный с помощью функций интерфейса, состоит из элементов геометрического представления *geometric_representation_items*. Каждый вид создается с помощью прикладного программирования внутри собственного контекста геометрического представления *geometric_representation_context*. Данный контекст называется координатной системой вида объекта *object_view_modelling_coordinate_system* (OVC). Прикладная программа не зависит от относительного

расположения OVC в координатной системе CAD. Система управления библиотекой (LMS) отвечает за инициализацию вида. Считается, что после выполнения функции инициализации вида все элементы геометрического представления *geometric_representation_items*, посылаемые в CAD в их собственной OVC, точно позиционируются и/или преобразуются.

Функция инициализации вида *View_Initialisation* должна, следовательно, активизировать некоторый неописанный процесс позиционирования. На практике (в соответствии с особой философией CAD) процесс позиционирования может, например:

1) определять новые локальные координатные системы, если CAD использует механизм создания экземпляров;

2) подключать OVC к курсору для последующего позиционирования;

3) организовывать некоторые взаимосвязи с пользователем CAD, необходимые для запроса текущего положения и инициализации матрицы преобразования интерфейса;

4) вычислять правильное положение, если оно определяется положением экземпляра объекта, и инициализировать матрицу преобразования интерфейса;

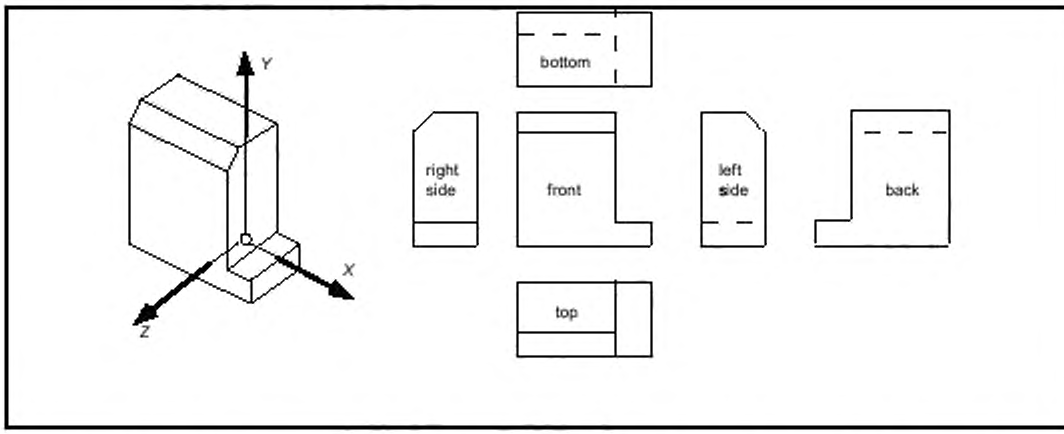
5) выполнять пустую операцию, если вид создан в начале глобальной координатной системы и позиционируется впоследствии пользователем CAD.

Если вид инициализирован как двухмерный, то 2D-пространство является плоскостью координат *x*, *y*. В данном случае использование координаты *z* не имеет смысла для геометрических сущностей. Для них координата *z* равна 0.

Если данные библиотеки деталей удовлетворяют требованиям комплекса стандартов ИСО 13584 и несколько программ поставщика детали ссылаются на различные функциональные виды одной детали, то OVC, используемые в данных программах, зависят друг от друга. Абсолютная координатная система ассоциируется с деталью с помощью поставщика. Все программы поставщика детали, создающие функциональные 3D-виды настоящей детали, должны использовать данную абсолютную координатную систему как их собственную OVC. Все программы поставщика детали, создающие функциональные 2D-виды настоящей детали, должны:

1) давать описание функционального 2D-вида, созданного каждой программой поставщика детали, в соответствии со стандартом ИСО 128 (см. рисунок 1);

2) выбирать в качестве OVC (для каждой программы поставщика детали) координатную систему, полученную из абсолютной координатной системы детали и из спецификации созданного функционального 2D-вида (см. рисунок 1).



Bottom — вид снизу; right side — вид справа; left side — вид слева; front — вид спереди; back — вид сзади; top — вид сверху

Рисунок 1 — Абсолютная координатная система детали (определяется поставщиком)

5.3.2 Геометрические единицы OVC

Используемые в OVC единицы длины и единицы плоского угла определяются тремя записями таблицы статуса интерфейса: единицы длины вида *view_length_unit*, масштабный фактор длины вида *view_length_scale_factor* и единицы угла вида *view_angle_unit*. Запись *view_length_unit* определяет

базовую единицу длины вида. Это может быть метр (metre) или дюйм (inch). Запись *view_length_scale_factor* определяет мультипликативный масштабный фактор, на который умножается базовая единица длины. Запись *view_angle_unit* определяет единицу плоского угла вида. Это может быть радиан (rad), градус (deg) или град (grad). В настоящем стандарте термин «единица OVC» означает либо единицу длины вида *view_length_unit*, масштабированную с помощью множителя *view_length_scale_factor* (единицу длины координатной системы *OVC_length_unit*), либо единицу угла вида *view_angle_unit* (единицу угла координатной системы *OVC_angle_unit*).

Значения единиц измерения OVC представлены в разделе 8.2 настоящего стандарта. Данные значения по умолчанию могут быть переопределены поставщиком детали (вне используемой программы). Новые значения единиц составляют часть функциональной модели программы поставщика детали. Значения по умолчанию единиц измерения устанавливаются в процессе инициализации вида. Они могут запрашиваться, но не могут быть изменены прикладной программой. Все геометрические размеры, которые могут задаваться (вычисляться) прикладными программами, описаны как текущие единицы измерения OVC.

Интерфейс гарантирует правильное масштабирование при переходе из единиц OVC в единицы моделирующего пространства CAD. Указанное масштабирование называется «преобразование OVC — CAD». Оно применяется для всех элементов геометрических представлений, созданных с помощью функций интерфейса.

5.3.3 Содержимое вида

Функции интерфейса создают данные внутри баз данных систем, моделирующих продукт. Базы данных систем, моделирующих продукт, отличаются друг от друга. Точного описания каждой функции не существует на физическом уровне. Для точного описания работы какой-либо функции интерфейса настоящий стандарт определяет целевую базу данных CAD посредством логической модели в виде информационной модели на языке EXPRESS (см. раздел 6). Принято, что данная логическая модель может быть использована некоторым физическим способом в целевой CAD.

5.3.4 Временная база данных

Временные базы данных (TDB) используются для промежуточных геометрических построений. Функции интерфейса позволяют создавать геометрические сущности либо во временной базе данных, либо в данных CAD. Сущности с временными данными могут быть ссылочными, их можно редактировать, использовать при геометрических построениях или направлять в CAD. Сущности внутри CAD уже не являются ссылочными. Если сущность временной базы данных направляется в CAD, то к временным данным она уже не относится.

Для геометрических сущностей с одинаковыми типами визуализации атрибуты могут быть созданы как сущности временной базы данных, так и в CAD.

Атрибуты визуализации прикрепляются к данным сущностям, если они созданы на модальной основе. Это может иметь место как в TDB, так и в CAD. В TDB атрибуты сущности могут быть изменены. Если сущность посылается из TDB в CAD, то атрибуты визуализации данной сущности сохраняют свои текущие значения в TDB.

Сущности с временными данными могут быть геометрически сдвинуты или дублированы. Эти геометрические манипуляции не изменяют атрибуты визуализации: модифицированные (дублированные) сущности сохраняют атрибуты визуализации исходных сущностей.

В TDB и CAD имеются различные структурные соотношения. Сущности, посылаемые в CAD, структурированы с помощью комплектов. Комплекты — это объекты, постоянно присутствующие в базе данных CAD. Используется иерархическая структура комплектов. Для структур элементов TDB используются временные групповые структуры. Групповые структуры могут быть использованы для создания геометрических элементов. Данные групповые структуры также являются иерархическими. Максимальное количество сущностей, допускаемое в TDB при реализации интерфейса, должно быть не менее количества, установленного в разделе 9 настоящего стандарта.

Настоящий стандарт не дает описания какой-либо формы реализации указанной временной базы данных. Функции поставщика деталей должны быть эффективными.

5.3.5 Процесс удаления невидимых линий

В настоящем разделе дано описание концепции невидимых линий 2D-видов, созданных интерфейсом.

1. В дополнение к имеющимся сущностям кривых 2D-интерфейсы могут создавать «непрозрачные» заполненные области.

2. Запись таблицы статуса интерфейса, называемая включенной невидимой линией *hidden_line_involved* (HLI), указывает, включена или нет сущность кривой или сущность заполненной области (созданной интерфейсом) в процесс удаления невидимых линий. Если значение HLI «true», то каждая сущность кривой или сущность заполненной области (созданная функцией интерфейса) должна быть включена в процесс удаления невидимых линий и может подключаться к предварительно определенному стилю затенения изображения *api_predefined_occlusion_style* интерфейса прикладного программирования. *Api_predefined_occlusion_style* должен содержать атрибут *view_level* (представляющий действительное значение высоты сущности в некотором виртуальном 3D-пространстве), а также атрибут имени *name*, дающий описание того, как следует изменить сущность, если она становится невидимой. Непрозрачная заполненная область скрывает полностью или часть сущности кривой находящегося внутри области и имеющего меньшее значение уровня вида. Заполненная область не должна скрывать сущности кривых с тем же значением уровня вида.

3. Если значение HLI «false», то созданные сущности кривой и заполненной области можно направлять в CAD, если прикладная программа запрашивает выполнение указанной передачи с помощью функции *Fix_Ent* (функция фиксирования сущности в CAD) или запрашивает их непосредственное создание в CAD в процессе построения вида. Если значение HLI «true», то сущность кривой и сущность заполненной области остаются в интерфейсе до окончания процесса удаления невидимых линий.

4. Если прикладная программа запрашивает передачу в CAD сущности, включенной в процесс удаления невидимых линий, то либо с помощью функции *Fix_Ent*, либо путем запроса на непосредственное его создание в CAD данная сущность подключается к предварительно определенному виртуальному стилю интерфейса прикладного программирования *api_predefined_vitually_sent_style*, то есть отправляется виртуально. Если сущность отправлятся виртуально, то указанный виртуальный стиль *api_predefined_vitually_sent_style* должен содержать атрибут задания имени интерфейса прикладного программирования *api_set_name*, определяющий (в формате строки) уникальное имя текущего открытого множества.

5. Процесс удаления выполняется после завершения построения вида. В данный процесс включаются только сущности, отправленные виртуально. Сущности временной базы данных не включаются.

6. Если отправленная виртуально сущность кривой частично скрыта заполненной областью, то ее видимая часть имеет текущий стиль кривой. Части виртуально отправленной сущности невидимой кривой обрабатываются в соответствии с атрибутом *name* и требованиями предварительно определенного стиля затенения визуализации интерфейса прикладного программирования *api_predefined_occlusion_style*.

Значения атрибута:	Сущности кривой и заполненной области:
no_change	отправляются без каких-либо изменений;
dashed (штриховая линия)	отправляются как невидимые линии, если данное свойство существует в используемой системе CAD. В противном случае сущности кривой, границы заполненной области и штриховка отправляются как штриховые линии;
invisible (невидимая линия)	отправляются как невидимые линии, если настоящее свойство существует в используемой системе CAD. В противном случае отправка не производится.

Если заполненная область закрыта другой поверхностью, то удаление невидимой части в соответствии с настоящим стандартом производится только на линиях, принадлежащих заполненной области (линии границ, штриховка).

Включение точек в настоящем стандарте не рассматривается.

Удаление невидимых линий не является обязательным. Ввод возможности удаления невидимых линий *hidden_line_capability* в таблицу описаний интерфейса указывает на доступность данного режима. Ввод режима использования невидимых линий *hidden_line* в таблицу статуса интерфейса указывает на возможность активизации процесса удаления для следующего вида. Значение атрибута *hidden_line* по умолчанию присваивается атрибуту *hidden_line_capability* (если указанная возможность доступна, то она всегда активизируется, если нет других указаний в прикладной программе).

Процесс удаления невидимых линий может быть активизирован только для 2D-видов.

Процесс удаления невидимых линий не должен изменять структуру, заданную прикладной программой в процессе (виртуальной) отправки. Заданная структура регистрируется атрибутом задания

имени интерфейса прикладного программирования *api_set_name* в соответствии с предварительно определенным стилем виртуальной отправки *api_predefined_virtualy_sent_style* интерфейса прикладного программирования.

5.3.6 Процесс представления

Функции интерфейса создают данные внутри баз данных систем, моделирующих продукт. Принято, что процесс визуализации указанных данных управляется системой моделирования продукта и ее пользователем. Вместе с тем прикладные программы должны в некоторой степени управлять геометрическими аспектами сущностей (например, для удовлетворения требований соответствующих стандартов технического черчения или для установления некоторого семантического отличия между сущностями), так как пользователю обычно требуется некоторое подобие видов, полученных от различных поставщиков библиотек.

Указанные цели достигаются нижеследующим способом (см. подразделы 6.2.4 и 6.2.5):

1) все стили воспроизведения определяются либо как предварительно определенные стили, либо как внешне определенные стили;

2) предварительно определенные стили соответствуют настоящему стандарту. Внешне определенные стили соответствуют как настоящему стандарту, так и любому другому стандарту серии, распространяющейся на обмен данными о деталях;

3) предварительно определенные стили или внешне определенные стили описывают визуализацию соответствующего стиля только частично. В соответствии с определенными требованиями они могут устанавливать режим задания цвета, например, как зависящий от реализации;

4) интерфейс — это инструмент пользователя CAD для задания точных значений всех атрибутов изображений, зависящих от реализации для каждого предварительно определенного или внешне определенного стиля;

5) если протокол обмена видами (на который производится ссылка в прикладной программе) не поддерживается данной реализацией интерфейса, то первый стиль, определенный для текущего элемента представления в настоящем стандарте, должен быть использован вместо неизвестного стиля. При этом сообщение об ошибке не возникает.

5.4 Структура сущностей

5.4.1 Структура группы в TDB

В используемых TDB сущности объединяются в группы, определяемые сущностью *entity_structured*. Если функция предназначена для выполнения операций в группе, то она выполняет эти операции в режиме повторения (рекурсивно) для каждой соответствующей сущности группы. Если данная функция является функцией дублирования, то ее результатом также является группа. Указанная группа принадлежит текущей открытой группе, она имеет такую же групповую структуру, как и исходная функция. Дубликат открытой группы является замкнутой группой. Если данная функция является функцией модификации, то она сохраняет групповую структуру существующих сущностей, находящихся внутри модифицированной группы.

Сущность не модифицируется, и сообщение об ошибке не возникает, если функция работает в группе, содержащей геометрические сущности, недопустимые для использования в качестве входных параметров функции. Например, если функция изменения стиля представления кривой *chg_curve_style* запущена в группе, содержащей сущности точки, твердого тела и кривых, то сущность *curve_style* должна быть модифицирована, однако сущность точки и сущность твердого тела должны оставаться неизменными в той же самой групповой структуре.

TDB сама является группой. Она называется корневой группой *root_group*. Данная группа открыта и не должна закрываться, если интерфейс инициализирован. Таким образом, всегда должна существовать открытая группа.

За исключением корневой группы каждая сущность (геометрическая или структурированная) должна принадлежать только одной группе (которая может быть корневой группой). Группы структурируются в соответствии с иерархической структурой дерева. Корнем дерева является корневая группа.

Группы могут быть:

- созданными: они принадлежат текущей открытой группе или становятся текущей открытой группой;
- открытыми повторно: все сущности, созданные в TDB после повторного открытия группы, принадлежат данной группе до ее закрытия;
- закрытыми.

Сущности, направляемые (созданные) в CAD, удаляются из групповой структуры.

Для формирования иерархической групповой структуры открытые группы управляются с помощью стека. Верхним уровнем стека является текущая открытая группа.

Если интерфейс инициализирован, то корневая группа помещается в стек. Никакая функция не может закрыть данную группу, поэтому она всегда должна оставаться в стеке.

Если группа является созданной, то: 1) она принадлежит текущей открытой группе; 2) она помещается на вершину стека. Таким образом, она становится текущей открытой группой.

Только группы, находящиеся на вершине стека, могут быть закрытыми. В настоящем случае группа удаляется из стека, и на вершине стека размещается следующая открытая группа.

Если группа открыта повторно, то она помещается на вершине стека. Это не изменяет свойства группы, которой принадлежит группа, открытая повторно.

Непосредственная модификация групповых структур обеспечивается тремя функциями. Ни одна из этих функций не может изменить содержимого стека.

Функция *Remove_Ent_Grp* удаляет сущности (геометрические или структурированные) из группы. По окончании выполнения функции сущность должна принадлежать корневой группе.

Функция *Gather_Ent_Grp* формирует новую группу, используя имеющийся перечень сущностей (геометрических или структурированных). Ни одна из этих сущностей не должна содержать текущей открытой группы. Все указанные сущности удаляются из группы, которой они принадлежат, и помещаются в новую группу. Данная группа должна принадлежать текущей открытой группе.

Функция *Add_Ent_Grp* добавляет сущности (геометрические или структурированные) в указанную группу. Сущности не должны содержать группу, к которой они добавляются. Данные сущности сначала удаляются из их группы, а затем добавляются в указанную группу.

Группы являются локальными по отношению к TDB. Группы облегчают геометрические построения. Максимальное количество групп, допускаемых реализацией интерфейса, должно быть не менее, чем количество, установленное в разделе 9 настоящего стандарта.

5.4.2 Структура сущностей, направляемых в CAD

Принято, что данные, хранящиеся в базе данных CAD, разделены на подмножества. Концептуально все данные принадлежат видам. Внутри некоторого вида геометрические данные структурированы в множества и подмножества в соответствии с иерархической структурой дерева. Структура, придаваемая данным, отправляемым используемой прикладной программой, описывается CAD следующим образом:

1) перед отправкой любых данных LMS инициализирует вид. Все данные, направляемые в CAD в промежуток времени между указанной инициализацией и окончанием работы прикладного программирования, должны принадлежать виду. Рассматриваемый вид не должен содержать других видов;

2) функция *Open_Set* открывает множество. Имя множества размещается на вершине заданного стека. Все геометрические сущности, направляемые в CAD, должны принадлежать настоящему множеству. Множество является подмножеством имеющегося содержимого вершины заданного стека или, если данный стек пуст, подмножеством вида.

Функция закрытия множества *Close_Set* применяется только для множества, которое уже находится на вершине данного стека. При вызове данной функции множество закрывается и его имя удаляется из стека. Если стек заполнен, то вершина стека содержит текущее открытое множество. Если стек пуст, то открытого множества нет. Закрытое множество не подлежит повторному открытию. Имя каждого множества должно быть уникальным внутри вида. Максимальный размер стека множества, допускаемый для TDB реализацией интерфейса, должен быть не менее, чем размер, установленный в разделе 9 настоящего стандарта.

Отображение между настоящей концептуальной структурой и структурой дерева ограниченной глубины, доступное в целевой системе CAD, производится следующим образом. Вершина структуры дерева CAD, если таковая существует, отображает структуру вида. Нижеследующие уровни, если они существуют, отображают первые уровни множеств и подмножеств структуры дерева. Если некоторое множество структуры дерева CAD является конечным (не делится на подмножества), то все сущности, принадлежащие подмножествам соответствующего концептуального множества, помещаются в данное конечное множество.

5.5 Имя геометрической или структурированной сущности

Для ссылки на любую сущность, созданную в TDB, все сущности, созданные функцией интерфейса, именуются значениями, принадлежащими некоторому абстрактному типу данных *entity_name_type*.

Значение указанного абстрактного типа данных либо равно 0, либо неизвестно. Если функция интерфейса не срабатывает и не позволяет создать некоторую сущность, то возвращаемое значение равно 0. Если сущность отправляется в CAD и доступ к сущности уже невозможен, то ее имя становится неизвестным. Неизвестное значение, как правило, возвращается функциями интерфейса, если сущность создана непосредственно в базе данных CAD. Если для функции интерфейса в качестве значений аргументов выбирается 0 или «неизвестное значение», то возвращаемое значение равно 0.

Все значения, не являющиеся нулевыми или неизвестными значениями имени сущности и возвращенные функцией интерфейса в одном сеансе, то есть с момента запуска LMS прикладной программой и до момента возвращения значения данной программой, должны быть уникальными. Имя сущности не может быть использовано повторно, даже если она направлена в CAD.

Примечание — На языке FORTRAN тип имени сущности *entity_name_type* представляется целым числом INTEGER. Нулевое значение отображается на 0. Неизвестное значение отображается на отрицательное целое. Таким образом, доступное имя сущности может отображаться только на положительное целое.

5.6 Координатная система и ее преобразование

Интерфейс имеет функции, которые могут изменять ссылочную координатную систему пространства с заданной OVC. Прикладные программы могут использовать такие четыре функции: *Ref_Sys_3_Pnt*, *Ref_Sys_2_Dir*, *Ref_Sys_Position_Relative* и *Ref_Sys_A2p*. После изменения координатной системы (либо в TDB, либо в базе данных CAD) все сущности определяются в новой ссылочной координатной системе.

Для сохранения предшествующей ссылочной координатной системы функция *Ref_Sys_A2p* создает локальную координатную систему (сущность *axis2_placement*) из текущей ссылочной координатной системы вида объекта (OVC). Указанная замена ссылочной координатной системы на данную LCS производится так, что позволяет вернуться к исходной OVC.

5.7 Состояние ошибки интерфейса

Глобальная переменная ошибки *error_variable* задается, если идентифицируется состояние ошибки в процессе выполнения функции интерфейса. Эта переменная принимает целочисленное значение из таблицы статуса интерфейса. Оно совпадает с номером ошибки, определяемым функцией спецификации. При этом в таблицу статуса интерфейса вносится запись «error_origin» с указанием имени функции, где произошла ошибка. Запись «error_text» содержит сообщение, ассоциированное с номером ошибки. Именем функции должно быть синтаксическое имя на текущем используемом языке программирования (например, на языке FORTRAN). Сообщение должно быть переводом описания ошибки (см. подраздел 5.8.1). Указанные переменные ошибки могут запрашиваться и задаваться повторно с помощью прикладных программ.

Если переменная ошибки *error_variable* определена, значит, интерфейс находится в состоянии ошибки (*error_state* = *true*). В указанном состоянии ошибки могут работать только нижеследующие функции интерфейса, установленные в приложении А настоящего стандарта.

1) функции запроса;

2) функции перезагрузки из состояния ошибки *Reset_Error_State*.

Все прочие функции интерфейса являются допустимыми, но изменить что-либо они не могут. Они возвращаются к вызывающей прикладной программе. Если прикладная программа возвращается, когда интерфейс находится в состоянии ошибки, то LMS должна:

1) закрыть все открытые множества;

2) закрыть открытый вид со значением *error_state* = *true*;

3) сделать записи в файле ошибок: *error_variable*, *error_origin*, *error_text*;

4) закрыть интерфейс.

5.8 Исправление ошибок

5.8.1 Методология исправления ошибок

Для каждой функции интерфейса имеется описание конечного количества состояний ошибки, при котором переменным ошибки присваиваются некоторые значения. Каждая реализация интерфейса должна поддерживать режим проверки ошибки. Переменные ошибки обеспечивают связь между прикладной программой и стандартным интерфейсом. Прикладная программа запрашивает значение ошибки, интерпретирует информацию об ошибке и заново устанавливает значение переменной ошибки

error_variable = *zero*, чтобы восстановить интерфейс в состоянии «отсутствие ошибки» (*error_state* = *false*). Выбираемая стратегия исправления ошибок интерфейса соответствует нижеследующей классификации ошибок:

- класс I — ошибки, приводящие к заранее известной реакции системы;
- класс II — ошибки, связанные с попыткой сохранить результаты или предшествующие операции;
- класс III — ошибки, приводящие к непредсказуемым результатам, включая разрушение CAD.

Интерфейс распознает три ситуации выявления ошибки:

- ситуация А — ошибка в функции интерфейса;
- ситуация Б — ошибка в функции, вызываемой из интерфейса (функции CAD, функции операционной системы и т. д.);
- ситуация В — ошибка вне интерфейса.

Если ошибки выявлены вне интерфейса (ситуация В), то либо прикладная программа принимает на себя управление выполнением операций, либо выполнение программы заканчивается ненормально. В последнем случае результаты являются непредсказуемыми (класс III), это может привести к разрушению CAD. Если тем не менее прикладная программа принимает на себя управление, то она может попытаться вернуться в LMS, чтобы попробовать закрыть интерфейс надлежащим образом (см. раздел 5.7). Операции, определенные в разделе 5.7, могут быть также выполнены самим интерфейсом. Это стандартная реакция на ошибки класса II.

Все ошибки, явно представленные в перечне как часть определения функции интерфейса, принадлежат к классу I. Они либо возникают внутри самого интерфейса (ситуация А), либо возникают, когда функция, вызванная из интерфейса, передает управление обратно некоторой функции интерфейса вместе с соответствующей информацией об ошибке (ситуация Б). Во всех случаях выявления ошибок класса I интерфейс задает значения переменным ошибки *error_variable*, *error_origin* и *error_text*. Если в процессе создания функции сущности имеет место сбой и сущность не может быть создана, то имя сущности, вычисленное функцией, устанавливается равным 0. Если функция интерфейса активизируется с более чем одним состоянием ошибки, то любой из соответствующих номеров ошибки может быть присвоен переменной ошибки.

Функция запроса *Inq_Error_State* позволяет исправлять ошибки с помощью прикладного программирования. Функция *Reset_Error_State* выводит интерфейс из состояния ошибки. Закрытие открытого вида при значении переменной *error_state* = *true* позволяет LMS заранее предупредить CAD о том, что вид является неправильным.

При наступлении состояния ошибки функции запроса действуют в соответствии с их функциональным описанием (см. приложение А), их действия не должны генерировать новых ошибок. Поэтому для функций запроса описаний ошибок не существует. Для сообщений о возможных трудностях в процессе выполнения функций используется особый выходной параметр: *error_indicator* (индикатор ошибок).

Каждая ошибка имеет свой индивидуальный номер:

1) номера менее 1001 не используются. Они зарезервированы для последующих обновлений стандарта;

2) номера ошибок от 1000 до 2000 зарезервированы для привязок языка программирования.

5.8.2 Сообщения об ошибках

Таблица 1 — Сообщения об ошибках входа

Номер ошибки	Описание ошибки
1	Имя сущности не определено (значение равно 0, значение неизвестно)
2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона
4	Значение меры плоского угла находится вне допустимого диапазона
5	Целочисленное значение находится вне допустимого диапазона
6	Значение строки находится вне допустимого диапазона
7	Действительное значение находится вне допустимого диапазона

Таблица 2 — Сообщения об ошибках геометрии

Номер ошибки	Описание ошибки
101	Попытка создания вырожденной сущности
102	Модуль вектора направления лежит вне установленного диапазона [EPS,MAX]
103	Расстояние между двумя точками лежит вне установленного диапазона [EPS, MAX]
104	Расстояние между двумя контурами менее EPS
105	Попытка создания вырожденного направления в процессе создания сущности
106	Попытка создания вырожденной оси axis2_placement в процессе создания сущности
107	Попытка создания вырожденной оси axis1_placement в процессе создания сущности
108	Попытка создания вырожденной базовой кривой в процессе создания сущности
109	Попытка создания вырожденного твердого тела в процессе создания сущности
110	Попытка создания точки вне параметрического диапазона сущности кривой
111	Попытка создания линии длиной вне установленного диапазона [EPS, MAX]
112	Попытка создания дуги длиной менее EPS
113	Попытка создания самопересекающейся сущности контура
114	Попытка создания твердого тела с перекрытием
115	Заданные сущности являются идентичными
116	Заданные точки линейно зависимы
117	Заданные направления параллельны
118	Заданные сущности кривой параллельны (концентрические)
119	Заданные сущности не лежат в одной плоскости
120	Заданный отрезок слишком длинный
121	Слишком большой (маленький) радиус
122	Пересечение заданных сущностей кривой отсутствует
123	Выявлено пересечение заданных контуров
124	Выявлено пересечение осей поверхности
125	Выявлено перекрытие заданных контуров
126	Ось поворота не лежит на плоской поверхности
127	Геометрическое построение нецелесообразно
128	Неустойчивый процесс вычисления конической дуги
129	Сбой процесса аппроксимации замыкания контура
130	Сбой булевой операции

Таблица 3 — Сообщения об ошибках системы

Номер ошибки	Описание ошибки
201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD
203	Функция несовместима с текущим уровнем интерфейса
204	Функция несовместима с текущим уровнем мощности
205	Превышено максимально допустимое число точек по линии
206	Превышено максимально допустимое число сущностей контура
207	Превышено максимально допустимое число внутренних границ
208	Превышено максимально допустимое число групп
209	Превышено максимально допустимое число символов строки
210	Переполнение стека группы
211	Переполнение заданного стека
212	Сущность может быть использована только внутри временной базы данных

Таблица 4 — Сообщения об ошибках структуры сущностей

Номер ошибки	Описание ошибки
301	Попытка закрыть корневую группу
302	Попытка повторно открыть открытую группу
303	Сущность является членом корневой группы
304	Сущность содержит текущую открытую группу
305	Попытка создания цикличной групповой структуры
306	Имя множества не является уникальным
307	Попытка закрыть корневое множество

Таблица 5 — Сообщения об ошибках стиля воспроизведения

Номер ошибки	Описание ошибки
401	Источник протокола обмена неизвестен
402	Идентификатор внешнего стиля неизвестен
403	Сбой при назначении стиля штриховки
404	Стиль затенения невидимых линий не подключен

Таблица 6 — Сообщения об ошибках привязки языка программирования

Номер ошибки	Описание ошибки
1001	Порядковый номер находится вне установленного диапазона
1002	Несоответствие порядкового номера и длины перечня
1003	Неправильно задана длина строки

6 Логическая модель целевой моделирующей системы

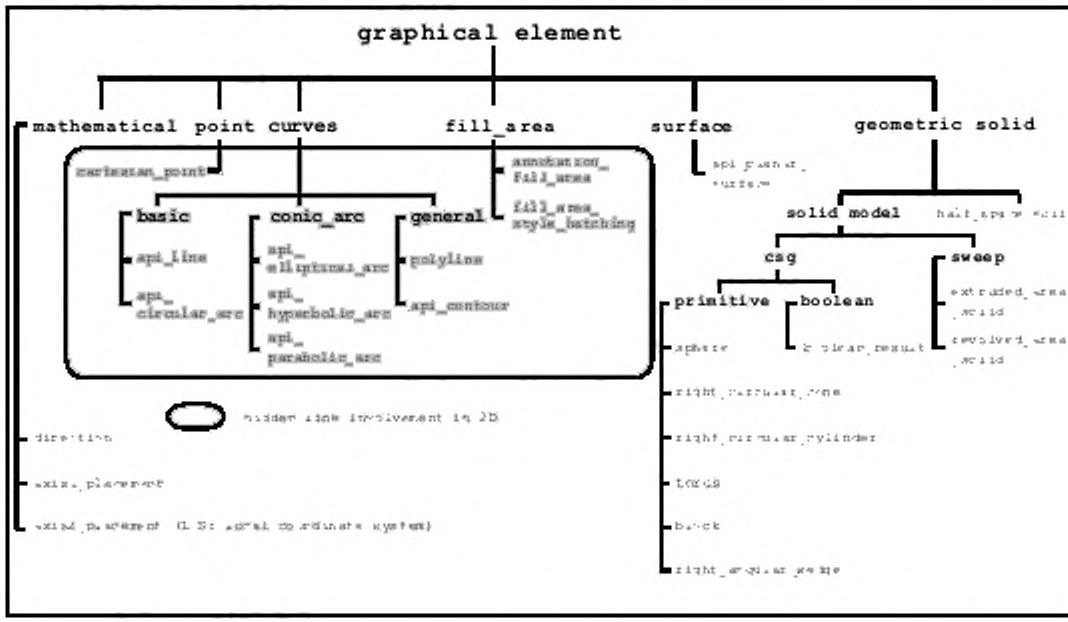
6.1 Элемент геометрического представления

Функции интерфейса создают элементы представления либо в TDB, либо в базе данных CAD.

Существует три вида элементов представления:

- элементы геометрического представления: геометрические сущности или сущности комментариев, используемые для описания формы, созданной с помощью интерфейса;
- стили: сущности, используемые для описания характеристик визуализации элементов геометрического представления;
- структурированные сущности: используются для структурирования элементов геометрического представления либо в TDB, либо в базе данных CAD.

Элементы геометрического представления, создаваемые с помощью функций интерфейса, классифицируются в соответствии с нижеследующим деревом, изображенным на рисунке 2.



Graphical element — графический элемент: mathematical — математические представления; point — точка; curves — кривые; fill_area - - заполненная область; surface — поверхность; geometric solid — геометрическое тело; cartesian_point - - декартова точка; annotation_fill_area — заполненная область комментариев, api_planar_surface -- плоская поверхность интерфейса прикладного программирования; fill_area_style_hatching -- штриховка заполненной области; basic -- базовый; conic_arc — коническая дуга; general - - основной; solid model — твердотельная модель; half_space_solid — тело в полупространстве; api_line — линия интерфейса прикладного программирования; api_elliptical_arc — дуга эллипса интерфейса прикладного программирования; polyline — полилиния; CSG — конструктивная блочная геометрия; sweep — тело, построенное очерчиванием контура, api_circular_arc — дуга окружности интерфейса прикладного программирования; api_hyperbolic_arc - - дуга гиперболы интерфейса прикладного программирования; api_contour — контур интерфейса прикладного программирования; primitive — примитив; boolean — булев; extruded_area_solid — тело, полученное вытягиванием; api_parabolic_arc— дуга параболы интерфейса прикладного программирования; sphere - - сфера; boolean_result — булев результат; revolved_area_solid — тело, полученное вращением: right_circular_cone — прямой круговой конус: hidden line involvement in 2D — включение невидимых линий в 2D-вид; direction — направление; right_circular_cylinder — прямой круговой цилиндр; axis1_placement — размещение оси 1; torus — тор; axis2_placement (LCS. local coordinate system) — размещение оси 2 (локальная координатная система); block — блок; right_angular_wedge - - прямой клин

Рисунок 2 — Элементы геометрического представления, определяемые в интерфейсе

Настоящая структура используется как для описания стилей различных сущностей, так и для описания диапазонов значений некоторых функций интерфейса.

Реализация элементов геометрического представления внутри TDB или внутри CAD не является стандартной. Тем не менее модель указанной реализации установлена в настоящем стандарте для

описания геометрического поведения каждой сущности в процессе манипуляции с ней. Настоящая модель определена с помощью абстрактной модели данных на языке EXPRESS. Данная абстрактная модель схемы *api_abstract_schema* использует подмножество групповых ресурсов, определенных в ИСО 10303 (части 41, 42, 43 и 46), содержащих описание данных модели продукта. Указанные ресурсы называются групповыми ресурсами ИСО 10303. Настоящая абстрактная модель данных не обязательна для применения в TDB или CAD.

Все сущности, созданные с помощью интерфейса, должны вести себя в соответствии с указанной абстрактной моделью данных.

Если реализация интерфейса создает сущности, удовлетворяющие требованиям некоторого протокола приложения, соответствующего ИСО 10303, описывающего групповой ресурс, то данное описание применяется для подмножества указанных ресурсов, используемых схемой *api_abstract_schema*. Любая дополнительная информация генерируется интерфейсом.

Типы и сущности схемы *api_abstract_schema*, определенные в групповом ресурсе, удовлетворяющем требованиям ИСО 10303, сохраняют свои имена, установленные указанным стандартом, даже если в их определении содержатся некоторые дополнительные ограничения допустимых подтипов. Указанные ограничения проверяются интерфейсом. Если они удовлетворяются, то созданные сущности должны соответствовать определению, данному ИСО 10303. Дополнительные правила типа «Где?» (WHERE RULES) (выражающие специфические ограничения схемы *api_abstract_schema*) идентифицируются именем с приставкой *«api_»*.

Некоторые сущности также определяются с помощью сущностей явного выделения подтипов, определенных интегрированным ресурсом ИСО 10303. Выделение подтипов используется для описания диапазонов значений некоторых функций интерфейса. Имена таких сущностей имеют приставку «*api_*». Выделение подтипов в общем случае состоит в задании ограничений для сущностей, определенных интегрированным ресурсом ИСО 10303. Указанная сущность может быть применена как экземпляр их супертипа или как экземпляр специализации указанных супертипов, определенных в некотором протоколе приложений, удовлетворяющем требованиям ИСО 10303.

Таким образом, некоторые сущности получаются путем обобщения сущностей, определенных в групповых ресурсах ИСО 10303, путем добавления новых атрибутов. Указанные сущности используются главным образом для определения структур и их визуализации. Если целевая CAD представляет собой архив (соответствующий протоколу приложений, удовлетворяющему требованиям ИСО 10303), то интерфейс должен гарантировать отображение этих сущностей на ресурс, доступный внутри указанного протокола приложений. Для отображения дается текстовое описание в определении сущности, работающей в интерфейсе прикладного программирования.

Если некоторые дополнительные ограничения налагаются групповым ресурсом ИСО 10303, представленным на языке EXPRESS, то указанные ограничения фиксируются в особом примечании. Если ограничения не зафиксированы, то определение ресурса соответствует определению группового ресурса ИСО 10303.

В интегрированных ресурсах комплекса стандартов ИСО 10303 некоторые групповые ресурсы явно ссылаются на другие групповые ресурсы, которые не активизируются и не используются для ссылок схемой *api_abstract_schema*. Их экземпляр не может появиться в совокупности, удовлетворяющей требованиям схемы *api_abstract_schema*. На эти ресурсы производятся ссылки из соответствующей схемы (представленной на языке EXPRESS) интегрированного ресурса ИСО 10303. Это необходимо для сохранения структуры данных групповых ресурсов [особенно существующих правил типа «Где?» (WHERE RULES)] при условии сохранения корректности используемой схемы. На указанные сущности ссылаются в соответствии с правилами типа «Где?» (WHERE RULES), дублированными из групповых ресурсов ИСО 10303, но не из сущностей, принадлежащих схеме *api_abstract_schema*. Таким образом, данная ссылка является только формальной.

Сущности, созданные интерфейсом, не могут быть вырожденными. Понятие вырождения не должно зависеть от какой-либо реализации интерфейса. В настоящем стандарте ограничения определены для каждой сущности, создаваемой интерфейсом. Сущность, не удовлетворяющую этим ограничениям, называют вырожденной. Если функция пытается создать вырожденную сущность, то возникает ошибка. Такая сущность не должна быть создана, формирование сообщения об ошибке обязательно.

Вырождение сущности определяется путем ссылки на абсолютное минимально допустимое значение EPS, выраженное через (текущую) единицу длины *view_length_unit* и масштабный фактор *view_scale_factor*:

$EPS = 10^{-3} \times view_length_unit \times view_scale_factor$.

Если некоторая реализация интерфейса не может создавать сущности с характерным размером EPS для заданной единицы длины *view_length_unit* с помощью прикладного программирования, то при осуществлении функции *set_OVC_length_unit*, запущенной LMS, должна возникать ошибка.

Константа *ZERO_value* определяет действительное значение, идентифицирующее нуль с помощью интерфейса на любой стадии его реализации. Если расстояние между двумя точками меньше величины *ZERO_value*, то эти две точки считаются совпадающими в данном интерфейсе.

Например, если расстояние между двумя точками *composite_curve_segment*, принадлежащими одной комбинированной кривой *composite_curve*, меньше *ZERO_value*, то интерфейс должен обеспечивать непрерывность данной комбинированной кривой *composite_curve* вне зависимости от требуемой точности целевой CAD. Величина *ZERO_value* выражается через (текущую) единицу длины *view_length_unit* и масштабный фактор *view_scale_factor*:

$ZERO_value = 10^{-6} \times view_length_unit \times view_length_scale_factor$.

Примечание — Определение этих двух значений указывает на нецелесообразность точных действительных вычислений, результаты которых могут быть неоднозначными. Полученное решение, часто используемое на практике, состоит в определении диапазона [*ZERO_value*, EPS] запрещенных действительных значений.

6.1.1 Схема API_ABSTRACT_SCHEMA

В настоящем подразделе установлены требования к схеме *api_abstract_schema*. Нижеследующая спецификация на языке EXPRESS представляет блок схемы *api_abstract_schema* и ее ссылки на внешние ресурсы, необходимые для обеспечения соответствия с определением группового ресурса, приведенные в ИСО 10303. Экземпляры указанных ссылочных сущностей не должны составлять совокупность, созданную с помощью функции интерфейса.

Спецификация на языке EXPRESS:

```
*)
SCHEMA api_abstract_schema;
 REFERENCE FROM geometry_schema
   (pcurve);
 REFERENCE FROM measure_schema
   (measure_with_unit,
descriptive_measure);
 REFERENCE FROM presentation_appearance_schema
   (surface_style_usage,
    presentation_style_by_context,
    fill_area_style_colour,
    fill_area_style_tiles,
    pre_defined_hatch_style,
    pre_defined_presentation_style,
    pre_defined_tile_style,
    externally_defined_hatch_style,
    externally_defined_curve_font,
    externally_defined_tile_style,
    curve_style_font,
    curve_style_font_and_scaling,
    text_style,
    point_style,
    symbol_style,
    approximation_tolerance);
(*
```

Примечание — Вышеуказанные ссылочные схемы определены в следующих частях ИСО 10303:
geometry_schema — в ИСО 10303-42;
measure_schema — в ИСО 10303-41;
presentation_appearance_schema — в ИСО 10303-46.

6.1.1.1 Определение констант схемы API_ABSTRACT_SCHEMA: точность геометрических представлений

В настоящем подразделе установлены поименованные константы, используемые схемой *api_abstract_schema* как ссылочные числовые границы для обеспечения точности геометрических представлений.

Спецификация на языке EXPRESS:

```
*)
CONSTANT
 EPS     : REAL :- 1.E-3;
 ZERO_VALUE : REAL :- 1.E-6;
 MAX     : REAL :- 1.E+4;
END_CONSTANT;
(*
```

Примечание — В контексте схемы *api_abstract_schema* значения EPS, *ZERO_value* и MAX выражают с помощью единиц длины *view_length_unit* и масштабного фактора *view_length_scale_factor* для меры длины *length_measure*, а также с помощью угловых единиц *view_angle_unit* для *plane_angle_measure*.

6.1.2 Определения типов схемы API_ABSTRACT_SCHEMA: основные понятия описания продукта и его поддержки

Настоящий подраздел устанавливает ресурсы группового типа (определенные в ИСО 10303-41), являющиеся частью схемы *api_abstract_schema*, соответствующей требованиям настоящего стандарта.

6.1.2.1 Идентификатор

Идентификатор (*identifier*) — это буквенно-цифровая строка для индивидуальной идентификации. Идентификатор может не иметь языкового значения.

Пример — Идентификатором может быть номер детали.

Спецификация на языке EXPRESS:

```
*)
TYPE identifier = STRING;
END_TYPE;
(*
```

6.1.2.2 Метка

Метка (*label*) — это совокупность символов для ссылки. Метка представляет собой строку, отображающую удобочитаемое название чего-либо. Метка должна иметь языковое значение.

Пример — «Smith», «Widget Inc.» и «Materials Test Laboratory» — типовые метки.

Спецификация на языке EXPRESS:

```
*)
TYPE label = STRING;
END_TYPE;
(*
```

6.1.2.3 Текст

Текст (*text*) — это буквенно-цифровая строка символов, предназначенных для чтения и восприятия человеком. Текст предназначен только для передачи информации.

Спецификация на языке EXPRESS:

```
*)
TYPE text = STRING;
END_TYPE;
(*
```

6.1.2.4 Сущность length_measure

Сущность *length_measure* устанавливает расстояния.

Спецификация на языке EXPRESS:

```
*)
TYPE length_measure = REAL;
END_TYPE;
(*
```

Примечание — В контексте схемы *api_abstract_schema* сущность *length_measure* выражается через единицы длины *view_length_unit* и масштабный фактор *view_length_scale_factor*.

6.1.2.5 Сущность plane_angle_measure

Сущность *plane_angle_measure* задает плоский угол.

Спецификация на языке EXPRESS:

```
*)
TYPE plane_angle_measure = REAL;
END_TYPE;
(*
```

Примечание — В контексте схемы *api_abstract_schema* плоский угол *plane_angle_measure* выражается с помощью угловых единиц *view_angle_unit*.

6.1.2.6 Сущность positive_length_measure

Сущность *positive_length_measure* задает положительную меру длины *length_measure*.

Спецификация на языке EXPRESS:

```
*)
TYPE positive_length_measure = length_measure;
WHERE
 WR1: SELF > 0;
END_TYPE;
(*
```

Комментарий к спецификации:

WR1: значение должно быть положительным.

6.1.2.7 Сущность positive_plane_angle_measure

Сущность *positive_plane_angle_measure* задает положительный плоский угол *plane_angle_measure*.

Спецификация на языке EXPRESS:

```
*)
TYPE positive_plane_angle_measure = plane_angle_measure;
WHERE
 WR1: SELF > 0;
END_TYPE;
(*
```

Комментарий к спецификации:

WR1: значение должно быть положительным.

6.1.2.8 Сущность parameter_value

Сущность *parameter_value* задает значение параметра в некоторой его области.

Спецификация на языке EXPRESS:

```
*)
TYPE parameter_value = REAL;
END_TYPE;
(*
```

6.1.2.9 Сущность message

Сущность *message* задает коммуникацию, адресованную системе для активизации некоторого действия. Результатом указанного действия является внешне определенный элемент *externally_defined_item*.

Примечание — Нормативные значения сообщений описаны внутри моделей соответствующих приложений.

Спецификация на языке EXPRESS:

```
*)
TYPE message = STRING;
END_TYPE;
(*
```

6.1.2.10 Ссылка

Ссылка (*reference*) обеспечивает идентификацию и запрос внешне определенного элемента *externally_defined_item*.

Спецификация на языке EXPRESS:

```
*)
TYPE source_item = SELECT (identifier, message);
END_TYPE;
(*
```

6.1.3 Определения типов схемы API_ABSTRACT_SCHEMA: геометрические и топологические представления

В настоящем подразделе установлены ресурсы группового типа, определенные в ИСО 10303-42, являющиеся частью схемы *api_abstract_schema*.

6.1.3.1 Сущность dimension_count

Сущность *dimension_count* задает положительное целое, равное размерности координатного пространства в контексте геометрического представления *geometric_representation_context*.

Спецификация на языке EXPRESS:

```
*)
TYPE dimension_count = INTEGER;
WHERE
 WR1: SELF > 0;
END_TYPE;
(*
```

Комментарий к спецификации:

WR1: целое *dimension_count* должно быть положительным.

6.1.3.2 Сущность transition_code

Сущность *transition_code* задает свойства непрерывности комбинированной кривой или поверхности. Рассматриваемая непрерывность является геометрической, а не параметрической.

Спецификация на языке EXPRESS:

```
*)
TYPE transition_code = ENUMERATION OF
 (discontinuous,
  continuous,
  cont_same_gradient,
  cont_same_gradient_same_curvature);
END_TYPE;
(*
```

Определения элементов перечислимого типа.

discontinuous: сегменты (вставки) не соединяются. Это возможно только на границе кривой (поверхности), чтобы обозначить ее незамкнутость;

continuous: сегменты (вставки) соединяются, условия на касательные к сегменту не налагаются;

cont_same_gradient: сегменты (вставки) соединяются, их касательные векторы (плоскости) параллельны и одинаково направлены в точке стыка; равенства производных не требуется;

cont_same_gradient_same_curvature: для кривых: сегменты соединены, их касательные векторы параллельны, имеют одно направление, в месте стыка кривизны равны, равенства производных не требуется. Для поверхностей: главные кривизны равны, главные направления совпадают вдоль общей границы.

Примечание — В контексте схемы *api_abstract_schema* существуют только комбинированные кривые.

6.1.3.3 Сущность preferred_surface_curve_representation

Сущность *preferred_surface_curve_representation* используется для указания предпочтительной формы представления кривой на поверхности. Кривая может быть задана в геометрическом пространстве или в параметрическом пространстве базовой поверхности.

Спецификация на языке EXPRESS:

```
*)
TYPE preferred_surface_curve_representation = ENUMERATION OF
 (curve_3d,
  pcurve_s1,
  pcurve_s2);
END_TYPE;
(*
```

Определения элементов перечислимого типа:
curve_3d: предпочтительной является кривая *curve* в трехмерном пространстве,
pcurve_s1: предпочтительной является первая *pcurve*;
pcurve_s2: предпочтительной является вторая *pcurve*.

6.1.3.4 Сущность trimming_preference

Сущность *trimming_preference* используется для указания предпочтительного способа вычленения параметрической кривой, если отрезок определяется повторно.

Спецификация на языке EXPRESS:

```
*)
TYPE trimming_preference = ENUMERATION OF
 (cartesian, parameter,
  unspecified);
END_TYPE;
(*
```

Примечание — В контексте схемы *api_abstract_schema* предпочтения вычленения кривой *trimming_preference* зависят от реализации.

Определения атрибутов:
cartesian: указывает, что предпочтительной является настройка по декартовой точке;
parameter: указывает, что предпочтительным является значение параметра;
unspecified: указывает, что предпочтение не задано.

6.1.3.5 Сущность axis2_placement

Сущность *axis2_placement* имеет тип *select*. Используется как в двумерном, так и в трехмерном декартовом пространстве. Это позволяет ссылаться на сущности, использующие указанную информацию, без определения размерности пространства.

Спецификация на языке EXPRESS:

```
*)
TYPE axis2_placement = SELECT
 (axis2_placement_2d,
  axis2_placement_3d);
END_TYPE;
(*
```

6.1.3.6 Сущность curve_on_surface

Сущность *curve_on_surface* задает одну из кривых на параметрической поверхности:
- *pcurve*;
- *surface_curve*, включая такие особые ее подтипы, как *intersection_curve* и *seam_curve*;
- *composite_curve_on_surface*.

Сущность *curve_on_surface* задается для кривых типа *select* и позволяет пользоваться ими.

Спецификация на языке EXPRESS:

```
*)
TYPE curve_on_surface = SELECT
 (pcurve,
  surface_curve,
  composite_curve_on_surface);
END_TYPE;
(*
```

6.1.3.7 Сущность pcurve_or_surface

Сущность *pcurve_or_surface* имеет тип *select*. Позволяет идентифицировать *associated surface* или *pcurve* как атрибут *composite curve on surface*.

Спецификация на языке EXPRESS:

```
*)
TYPE pcurve_or_surface = SELECT
 (pcurve,
  surface);
END_TYPE;
(*
```

6.1.3.8 Сущность trimming_select

Сущность *trimming_select* имеет тип *select*. Идентифицирует два способа вычленения параметрической кривой: 1) путем задания декартовой точки на кривой; 2) путем задания действительного значения параметра в заданном диапазоне.

Спецификация на языке EXPRESS:

```
*)
TYPE trimming_select = SELECT
 (cartesian_point,
  parameter_value);
END_TYPE;
(*
```

6.1.3.9 Сущность vector_or_direction

Сущность *vector_or_direction* используется для идентификации типа сущности, применяемой для векторных вычислений.

Спецификация на языке EXPRESS:

```
*)
TYPE vector_or_direction = SELECT
 (vector,
  direction);
END_TYPE;
(*
```

6.1.4 Определения типов схемы API_ABSTRACT_SCHEMA: геометрические модели

В настоящем подразделе установлены ресурсы группового типа для построения геометрических моделей, определенных в ИСО 10303-42. Данные ресурсы являются частью схемы *api_abstract_schema*.

6.1.4.1 Сущность boolean_operand

Сущность *boolean_operand* имеет тип *select*. Идентифицирует все типы сущностей, используемых в булевых операциях создания тел конструктивной блочной геометрии.

Спецификация на языке EXPRESS:

```
*)
TYPE boolean_operand = SELECT
 (solid_model,
  half_space_solid,
  csg_primitive,
  boolean_result);
END_TYPE;
(*
```

6.1.4.2 Сущность boolean_operator

Сущность *boolean_operator* задает три булевых оператора для определения тел конструктивной блочной геометрии.

Спецификация на языке EXPRESS:

```
*)
TYPE boolean_operator = ENUMERATION OF
 (union,
  intersection,
  difference);
END_TYPE;
(*
```

Определения атрибутов.

union: операция построения заданного регуляризованного теоретического объединения объемов, определенных двумя телами;

intersection: операция построения заданного регуляризованного теоретического пересечения объемов, определенных двумя телами;

difference: операция построения заданной регуляризованной теоретической разности объемов, определенных двумя телами.

6.1.4.3 Сущность csg_primitive

Сущность *csg_primitive* имеет тип *select*. Определяет набор сущностей конструктивной блочной геометрии (*CSG*), используемых в булевых операциях: сфера, прямой круговой конус, прямой круговой цилиндр, тор, блок, прямой клин.

Спецификация на языке EXPRESS:

```
*)
TYPE csg_primitive = SELECT
 (sphere,
  block,
  right_angular_wedge,
  torus,
  right_circular_cone,
  right_circular_cylinder);
END_TYPE;
(*
```

6.1.4.4 Сущность csg_select

Сущность *csg_select* идентифицирует типы сущностей, выбираемых в качестве корня дерева конструктивной блочной геометрии (*CSG*). В частном случае это может быть отдельная сущность конструктивной блочной геометрии.

Спецификация на языке EXPRESS:

```
*)
TYPE csg_select = SELECT
 (boolean_result,
  csg_primitive);
END_TYPE;
(*
```

6.1.4.5 Сущность geometric_set_select

Сущность *geometric_set_select* имеет тип *select*. Идентифицирует типы геометрических сущностей.

Спецификация на языке EXPRESS:

```
*)
TYPE geometric_set_select = SELECT
 (point,
  curve,
  surface);
END_TYPE;
(*
```

6.1.5 Определения типов схемы API_ABSTRACT_SCHEMA: особые типы структурирования интерфейса прикладного программирования

В настоящем подразделе установлены особые типы ресурсов интерфейса прикладного программирования, предназначенные для структурирования элементов геометрического представления, созданного с помощью функций интерфейса.

6.1.5.1 Сущность api_grouped_item

Сущность *api_grouped_item* задает описания объектов, которые могут быть элементом группы.

Спецификация на языке EXPRESS:

```
*)
TYPE api_grouped_item = SELECT
 (direction,
  vector,
  placement,
  annotation_fill_area,
```

```
  fill_area_style_hatching,
  geometric_set_select,
  solid_model,
  half_space_solid,
  csg_select,
  api_group);
END_TYPE;
(*
```

6.1.5.2 Сущность api_set_item

Сущность *api_set_item* задает описания объектов, которые могут быть частью множества *api_set*.

Спецификация на языке EXPRESS:

```
*)
TYPE api_set_item = SELECT
 (direction,
  vector,
  placement,
  annotation_fill_area,
  geometric_set_select,
  solid_model,
  half_space_solid,
  csg_select,
  api_set);
END_TYPE;
(*
```

6.1.6 Определения сущностей схемы API_ABSTRACT_SCHEMA: основные понятия описания и поддержки продукта

В настоящем подразделе установлены ресурсы групповых сущностей, определенные в ИСО 10303-41. Указанные ресурсы являются частью схемы *api_abstract_schema*.

6.1.6.1 Сущность shape_representation

Сущность *shape_representation* — особый вид представления, определяющего геометрическую форму.

Примечание 1 — В контексте схемы *api_abstract_schema* существует только одно представление формы *shape_representation*. Оно соответствует форме продукта, созданного с помощью интерфейса базы данных CAD. Это установлено особыми глобальными правилами, ассоциированными с интерфейсом прикладного программирования.

Примечание 2 — В контексте схемы *api_abstract_schema* контекстом изделий *context_of_item* должен быть контекст геометрического представления *geometric_representation_context*. Это установлено правилами типа «Где?» (WHERE RULES) интерфейса прикладного программирования.

Спецификация на языке EXPRESS:

```
*)
ENTITY shape_representation
 SUBTYPE OF (representation);
WHERE
api_WR1: 'API_ABSTRACT_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT' IN
     TYPEOF (SELF\representation.context_of_items);
END_ENTITY;
(*
```

Определения атрибутов:

SELF\representation.item: множество *representation_item*, представляющих форму продукта;

SELF\representation.context_of_item: *representation_context* в координатной системе OVC, в которой элементы связаны с формой продукта.

Комментарии к спецификации:

api_WR1: элемент контекста *context_of_item* представления формы *shape_representation* должен быть контекстом геометрического представления *geometric_representation_context*.

Ассоциированное глобальное правило: следующее глобальное правило ассоциировано с настоящей сущностью. Оно ограничивает область применения сущности или ее соотношения с другими сущностями:

unique_shape_representation: правило представления уникальной формы *unique_shape_representation* требует существования уникальной сущности *shape_representation* в совокупности сущностей схемы *api_abstract_schema*. Данная сущность соответствует форме продукта, созданного с помощью интерфейса базы данных CAD.

6.1.6.2 Сущность group

Сущность *group* идентифицирует набор элементов.

Спецификация на языке EXPRESS:

```
*)
ENTITY group;
 name     : label;
 description : text;
END_ENTITY;
(*
```

Определения атрибутов:

name: слово или группа слов, с помощью которых производится ссылка на группу;

description: текст, поясняющий природу группы.

6.1.6.3 Сущность group_assignment

Сущность *group_assignment* устанавливает связь группы с данными продукта.

Спецификация на языке EXPRESS:

```
*)
ENTITY group_assignment
 ABSTRACT SUPERTYPE;
 assigned_group : group;
END_ENTITY;
(*
```

Определение атрибута:

assigned_group: группа, ассоциированная с данными продукта.

6.1.6.4 Сущность external_source

Сущность *external_source* идентифицирует источник данных продукта, не являющийся протоколом приложений, по которому производится обмен.

Примечание — В контексте схемы *api_abstract_schema* внешние источники *external_source* представляют собой внешне определенные стили *externally_defined_style* элементов геометрического представления.

Спецификация на языке EXPRESS:

```
*)
ENTITY external_source;
 source_id : source_item;
END_ENTITY;
(*
```

Определение атрибута:

source_id: идентификация внешнего источника *external_source*.

6.1.6.5 Сущность pre_defined_item

Сущность *pre_defined_item* идентифицирует информацию, не представленную явно в заданном обмене, но определенную в протоколе приложений, по которому производится обмен.

Примечание — В контексте схемы *api_abstract_schema* некоторые особые стили сущностей определяются как предварительно определенные элементы *pre_defined_item*.

Спецификация на языке EXPRESS:

```
*)
ENTITY pre_defined_item;
```

```
 name : label;
END_ENTITY;
(*
```

Определение атрибута:

name: слово или группа слов, с помощью которых производится ссылка на предварительно определенные элементы *pre_defined_item*.

6.1.6.6 Сущность externally_defined_item

Сущность *externally_defined_item* идентифицирует информацию, не представленную явно в заданном обмене и не определенную в протоколе приложений, по которому производится обмен.

Примечание — В контексте схемы *api_abstract_schema* стили сущностей определены как внешне определенные элементы *externally_defined_item*.

Спецификация на языке EXPRESS:

```
*)
ENTITY externally_defined_item;
 item_id : source_item;
 source : external_source;
END_ENTITY;
(*
```

Определения атрибутов:

item_id: идентификатор элемента, на который производится ссылка;

source: внешний источник *external_source*, содержащий элемент, на который производится ссылка.

6.1.7 Определения сущностей схемы API_ABSTRACT_SCHEMA: структуры представлений

В настоящем подразделе установлены ресурсы групповых сущностей, определенные в ИСО 10303-43. Указанные ресурсы являются частью схемы *api_abstract_schema*.

6.1.7.1 Сущность representation_context

Сущность *representation_context* задает контекст, в котором устанавливается связь между элементами представления *representation_items*.

Два некоторых контекста представления *representation_context* определяются отдельно, если обратное не установлено особым соотношением.

Спецификация на языке EXPRESS:

```
*)
ENTITY representation_context;
 context_identifier : identifier;
 context_type     : text;
INVERSE
 representations_in_context : SET [1:?] OF representation
  FOR context_of_items;
END_ENTITY;
(*
```

Определения атрибутов.

context_identifier: идентификатор контекста представления *representation_context*;

context_type: описание типа контекста представления *representation_context*;

representations_in_context: по крайней мере одно представление должно быть ассоциировано с каждым контекстом представления *representation_context*.

6.1.7.2 Сущность representation_item

Сущность *representation_item* — это элемент данных продукта. Он является частью одного и более представлений или дополняет определения других элементов представления *representation_items*.

Сущность *representation_item* дополняет определения другого элемента представления *representation_item*, если на него производится ссылка этим элементом.

Спецификация на языке EXPRESS:

```
*)
ENTITY representation_item;
 name : label;
WHERE
 WR1  : SIZEOF(using_representations(SELF)) > 0;
 api_WR2: SIZEOF(using_representations(SELF)) = 1;
END_ENTITY;
(*
```

Примечание — В контексте схемы *api_abstract_schema* каждый элемент представления *representation_item* должен быть ассоциирован только с одним контекстом представления *representation_context*. Это устанавливается сущностью *api_WR2*.

Определение атрибута:

name: идентификатор элемента представления *representation_item*.

Комментарии к спецификации:

WR1: каждый элемент представления *representation_item* должен быть ассоциирован по крайней мере с одним контекстом представления *representation_context*;

api_WR2: на элемент представления *representation_item* производится ссылка только одним контекстом представления *representation_context*.

6.1.7.3 Сущность representation

Задает набор из одного или нескольких элементов представления *representation_items* в некотором контексте представления *representation_context*. Порядок использования представления в ИСО 10303 не рассматривается.

Соотношение между элементами представления *representation_items* и контекстом представления *representation_context* дает основу для выделения связанных сущностей из полного набора сущностей *representation_item*.

Примечание 1 — Рассмотрим контекст, в котором множество элементов геометрического представления *geometric_representation_items* используется для представления некоторой формы. Используются только элементы данного контекста. Все прочие элементы геометрического представления не включены. Это составляет основу для выделения требуемых элементов геометрического представления на множестве элементов представления. В противном случае указанный признак не включается в спецификацию элементов представления.

Элементы множества изделий и все элементы представления, на которые косвенно производится ссылка из указанного множества, связаны с элементом контекста *context_of_item* через сущность представления. Косвенная ссылка на элементы представления имеет место, если она производится с помощью нескольких промежуточных сущностей, каждая из которых имеет тип элемента представления.

Примечание 2 — Представление связывает контекст представления *representation_context* с деревом элементов представления *representation_items*. При этом каждое дерево имеет корень на одном элементе множества. Элементы представления создают узлы на дереве, а ссылка одного элемента на другой создает ветвь дерева.

Множество представлений, на которые непосредственно производится ссылка как на изделие (связанные с контекстом *context_of_item*), является представлением. Элементы представления *representation_item*, на которые косвенно производится ссылка, поддерживают определения изделий и связаны с тем же контекстом представления *representation_context*.

Примечание 3 — В представлении формы куба с помощью множества линий указанное множество линий является единственным элементом представления формы. Сущности *lines*, в свою очередь, ссылаются на сущности декартовых точек *cartesian_point* и сущности направления *direction* (поддерживающие определения линий и связанные друг с другом), а также на определения сущности *lines* в ссылочном контексте геометрического представления *geometric_representation_context*. Указанная форма вместе с тем не может быть представлена с помощью сущностей декартовых точек и направления.

Представление должно соответствовать приложению. Часто представление является неполным и не моделирует исчерпывающим образом рассматриваемое понятие.

Примечание 4 — Рассмотрим набор двумерных элементов геометрических представлений *geometric_representation_items*, используемых для представления формы обрабатываемой детали. Описание ее формы является неполным, но пригодным для некоторых приложений, например, автоматизированного черчения.

Один элемент представления *representation_item* может быть связан с несколькими контекстами представления *representation_context*. Два представления не могут быть связаны, потому что на один и тот же элемент представления производится ссылка (прямо или косвенно) из соответствующих множеств элементов.

Примечание 5 — Рассмотрим поверхность *surface*, используемую в представлении *representation* геометрической формы литьевой пресс-формы, а также геометрической формы отливаемой в ней детали. Поверхность геометрически представляется двумя отличными контекстами *geometric_representation_contexts*, одним для пресс-формы и одним для детали. Вместе с тем данные представления не связаны. Наоборот, каждое из них по отдельности задает общую поверхность. В данном случае два несвязанных представления просто делят общий элемент представления *representation_item*.

Один и тот же элемент представления *representation_item* может быть многократно связан с одним контекстом представления *representation_contex*, используемым прямо или косвенно в нескольких представлениях, каждое их которых ссылается на один и тот же контекст представления. Это не означает, что каждое представление создает новый экземпляр того же элемента представления *representation_item* в том же контексте *representation_context*. Скорее, каждое представление подтверждает наличие одного экземпляра элемента представления в заданном контексте представления для различных целей.

Примечание 6 — Рассмотрим два представления с одинаковыми элементами контекста *context_of_items*. Одно является представлением формы «куб», оно косвенно ссылается на сущность *line* при задании ребер. Второе просто ссылается на сущность *line* как на элемент представления. Но это не означает наличия двух вхождений в сущность *line* и ее дерева ссылочных элементов геометрических представлений *geometric_representation_items* в рассматриваемом контексте геометрического представления *geometric_representation_context*. Скорее, однократное использование сущности *line* в данном контексте подтверждается дважды, по одному разу в каждом представлении. Причем первое может определять форму куба в целом. Второе может определять ребро того же куба.

Спецификация на языке EXPRESS:

```
*)
ENTITY representation;
 name       : label;
 items      : SET[1:?] OF representation_item;
 context_of_items : representation_context;
END_ENTITY;
(*
```

Определения атрибутов:

name: идентификатор представления *representation*;

item: множество элементов представления *representation_item*, связанных в контексте *context_of_item*;

context_of_item: контекст представления *representation_context*, в котором элементы связываются для формирования представления некоторого понятия.

Примечание 7 — В контексте схемы *api_abstract_schema* все элементы геометрического представления, созданные с помощью функции интерфейса, закладываются вместе с OVC в контекст изделия.

6.1.7.4 Сущность representation_map

Сущность *representation_map* идентифицирует представление *representation* и элемент представления *representation_item* в нем для целей отображения. Элемент представления определяет начало координат отображения. Карта представления *representation_map* используется как источник отображения с помощью элемента отображения *mapped_item*.

Примечание 1 — Определение отображения, используемое для описания нового элемента представления *representation_item*, содержит сущность отображения представления *representation_map* и сущность элемента отображения *mapped_item*. Без данных сущностей отображение получается неполным. Они дают возможность по одному исходному представлению *representation_map.mapped_representation* получить отображение нескольких новых представлений *mapped_item*.

Спецификация на языке EXPRESS:

```
*)
ENTITY representation_map;
 mapping_origin    : representation_item;
 mapped_representation : representation;
INVERSE
 map_usage        : SET[1:?] OF mapped_item FOR mapping_source;
WHERE
 WR1: item_in_context(SELF.mapping_origin,
           SELF.mapped_representation.context_of_items);
END_ENTITY;
(*
```

Определения атрибутов:

mapping_origin: элемент представления *representation_item*, для которого производится отображение представления *mapped_representation*.

Примечание 2 — Рассмотрим декартово отображение одного геометрического представления на другое. Источником отображения *mapping_origin* может быть локальная координатная система *axis2_placement* в контексте отображения представления *mapped_representation*, определяющего положение отображения;

mapped_representation: представление, отображаемое по крайней мере на один элемент отображения *mapped_item*;

map_usage: множество одного и более элементов отображения *mapped_item*, на которые производится отображение представления *representation_map*.

Комментарий к спецификации:

WR1: источник отображения *mapping_origin* должен содержаться в контексте *representation_context* отображения представления *mapped_representation*.

6.1.7.5 Сущность mapped_item

Сущность *mapped_item* использует представление *mapping_source.mapped_representation* при участии в отображении представления *representation_map* в качестве элемента представления *representation_item*.

Примечание 1 — Элемент отображения *mapped_item* является подтипом элемента представления *representation_item*. Он активизирует некоторое представление в виде элемента представления *representation_item* одного или нескольких других представлений. Элемент отображения определяет представление с помощью других представлений.

Отображение достигается посредством оператора, неявно определенного атрибутами *mapping_source.mapping_origin* и *mapping_target*. Отображение описывается с помощью преобразования *item_defined_transformation* (дополнительно см. ИСО 10303-43, пункт 4.4.7).

Спецификация на языке EXPRESS:

```
*)
ENTITY mapped_item
 SUBTYPE OF (representation_item);
 mapping_source : representation_map;
 mapping_target : representation_item;
WHERE
 WR1: acyclic_mapped_representation(using_representations(SELF), [SELF]);
END_ENTITY;
(*
```

Определения атрибутов:

mapping_source: отображение представления *representation_map*, которое является источником элемента отображения *mapped_item*;

mapping_target: элемент преставления *representation_item*, который является целью для источника отображения *mapping_source*.

Комментарий к спецификации:

WR1: источник отображения *mapped_item* не должен самоопределяться путем участия в определении отображения представления.

Примечание 2 — Подробности процедуры отображения определяются сущностями *mapped_item* и *representation_map*.

Пример — Рассмотрим декартово отображение одного геометрического представления на другое. Источником отображения mapping_source может быть отображение представления representation_map, ссылающееся на представление representation и размещение axis_placement, заложенное в контекст геометрического представления geometric_representation_context ссылочного представления. Элемент отображения mapped_item может ссылаться на данное отображение representation_map и на второе размещение axis_placement. В этом случае элемент mapped_item может стать элементом представления representation_item, являющимся отображением, на который производится ссылка представлением, так что источник отображения representation_map.mapping_origin налагается на цель отображения mapped_item.mapping_target.

6.1.8 Определения сущностей схемы API_ABSTRACT_SCHEMA: структуры геометрических представлений

В настоящем подразделе установлены ресурсы групповых сущностей, определенные в ИСО 10303-42 для структур геометрических представлений, являющихся частью схемы *api_abstract_schema*.

6.1.8.1 Сущность geometric_representation_context

Сущность *geometric_representation_context* является контекстом представления *representation_context*, в который геометрически закладываются элементы геометрического представления *geometric_representation_item*.

Контекст геометрического представления *geometric_representation_contexti* — это особое координатное пространство, не связанное с прочими координатными пространствами, за исключением тех координатных пространств, для которых применяются особые преобразования.

Спецификация на языке EXPRESS:

```
*)
ENTITY geometric_representation_context
 SUBTYPE OF (representation_context);
 coordinate_space_dimension : dimension_count;
END_ENTITY;
(*
```

Определение атрибута:

coordinate_space_dimension: целочисленный счетчик *dimension_count* размерности пространства.

Примечание — В контексте схемы *api_abstract_schema* OVC составляет контекст геометрического представления *geometric_representation_context*, в котором заложены все элементы геометрического представления. Размерность пространства *dimension_count* может быть равна 2 или 3 в соответствии с уровнем геометрической мощности *geometrical_power_level* текущего открытого вида.

6.1.8.2 Сущность geometric_representation_item

Сущность *geometric_representation_item* — это элемент представления *representation_item*, приобретающий особый смысл в результате добавления понятий геометрического положения и/или ориентации. Указанный смысл приобретается в результате:

- использования понятия декартовой точки *cartesian_point* или направления *direction*;
- прямой ссылки на декартову точку *cartesian_point* или направление *direction*;
- косвенной ссылки на декартову точку *cartesian_point* или направление *direction*.

Примечание 1 — Косвенная ссылка на декартову точку или направление означает, что заданный элемент геометрического представления *geometric_representation_item* ссылается на декартову точку или направление посредством одного или более промежуточных атрибутов. В большинстве случаев данная информация задается в форме размещения *axis2_placement*.

Пример 1 — Рассмотрим окружность. Она принимает свое геометрическое положение и ориентацию путем ссылки на размещение axis2_placement, которое, в свою очередь, ссылается на декартову точку и несколько направлений.

Пример 2 — Множественное целостное контурное представление manifold_solid_brep является элементом геометрического представления geometric_representation_item, который (посредством нескольких слоев элементов топологических представлений topological_

representation_items) ссылается на кривые, поверхности и точки. С помощью дополнительных промежуточных сущностей кривые и поверхности ссылаются на декартовы точки и направления. Определения сущностей manifold_solid_geometry, topological_representation_item и surface (на языке EXPRESS) приведены в ИСО 10303-42.

Примечание 2 — Промежуточные сущности (тип *representation_item*) не обязаны иметь подтип *geometric_representation_item*. Возьмем рассмотренный выше пример с сущностью *manifold_solid_brep*. Промежуточным уровнем элемента представления *representation_item* может быть крупный план замкнутой оболочки *close_shell*. Данный элемент топологического представления *topological_representation_item* не требует задания контекста геометрического представления *geometric_representation_context*. Если же сущность *close_shell* — часть сущности *manifold_solid_brep*, являющейся, в свою очередь, элементом геометрического представления *geometric_representation_item*, то она закладывается в контекст геометрического представления. Определение сущности *close_shell* (на языке EXPRESS) приведено в ИСО 10303-42.

Примечание 3 — Сущность *geometric_representation_item* наследует необходимость связанности с контекстом представления *representation_context*. Правило совместимой размерности *compatible_dimension* гарантирует, что контекст представления *representation_context* является контекстом геометрического представления *geometric_representation_context*. В данном контексте настоящее соотношение геометрически закладывает элементы геометрического представления *geometric_representation_items*. Определение термина «геометрически закладывает» приведено в ИСО 10303-42.

Спецификация на языке EXPRESS:

```
*)
ENTITY geometric_representation_item
 SUPERTYPE OF (ONEOF(point, direction, vector, placement, curve,
           annotation_fill_area, surface, solid_model,
           boolean_result, sphere, right_circular_cone,
           right_circular_cylinder, torus, block,
           right_angular_wedge, half_space_solid,
           fill_area_style_hatching,
           one_direction_repeat_factor))
 SUBTYPE OF (representation_item);
DERIVE
 dim : dimension_count := dimension_of(SELF);
WHERE
 api_WR1: SIZEOF (QUERY (using_rep <* using_representations (SELF) |
      NOT ('API_ABSTARCT_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT' IN
      TYPEOF (using_rep.context_of_items)))) = 0;
END_ENTITY;
(*
```

Определение атрибута:

dim: счетчик координат *dimension_count* элемента геометрического представления *geometric_representation_item*.

Примечание 4 — Атрибут *dim* вычисляется сущностью *coordinate_space_dimension* для контекста геометрического представления *geometric_representation_context*, в котором геометрически закладывается элемент геометрического представления *geometric_representation_item*.

Примечание 5 — Элементы геометрического представления геометрически закладываются в один и более контекст геометрического представления с одинаковой размерностью пространства *coordinate_space_dimension* (правило совместимой размерности *compatible_dimension* приведено в ИСО 10303-42).

Примечание 6 — В контексте схемы *api_abstract_schema* все элементы геометрического представления являются подтипами элемента геометрического представления.

Примечание 7 — В контексте схемы *api_abstract_schema* интерфейсом создаются только точки, направления, векторы, размещения, кривые, заполненные области комментариев, поверхности, твердотельные модели, сферы, результаты булевых операций, прямые круговые конусы, прямые круговые цилиндры, торы, блоки, прямые клинья, тела в полупространстве, заполненные области штриховки и факторы повтора в одном направлении. Поэтому необходима модификация супертипа.

Комментарий к спецификации:

api_WR1: в контексте схемы *api_abstract_schema* любое представление, ссылающееся на элемент *geometric_representation_item*, должно иметь тип контекста геометрического представления *geometric_representation_context*.

6.1.9 Определения сущностей схемы API_ABSTRACT_SCHEMA: геометрические математические сущности

В настоящем подразделе установлены ресурсы групповых сущностей, определенные в ИСО 10303-42. Указанные ресурсы являются частью схемы *api_abstract_schema*.

6.1.9.1 Сущность point

Сущность *point* задает точку в некотором действительном декартовом координатном пространстве R^m, где m = 1, 2 или 3.

Спецификация на языке EXPRESS:

```
*)
ENTITY point
 ABSTRACT SUPERTYPE OF (ONEOF(cartesian_point))
 SUBTYPE OF (geometric_representation_item);
END_ENTITY;
(*
```

Примечание — В контексте схемы *api_abstract_schema* существуют только декартовы точки *cartesian_point*. Сущность *point* определяется как абстрактный супертип, все прочие подтипы, определенные в ИСО 10303-42, удаляются.

6.1.9.2 Сущность cartesian_point

Сущность *cartesian_point* задает точку координатами в прямоугольной декартовой координатной системе или в параметрическом пространстве. Сущность определяется в одномерном, двумерном или трехмерном пространстве в соответствии с количеством координат, указанным в перечне.

Примечание — В схеме *api_abstract_schema* используются только двумерные и трехмерные точки. Сущность *cartesian_point* всегда определяется в декартовой координатной системе.

Спецификация на языке EXPRESS:

```
*)
ENTITY cartesian_point
 SUBTYPE OF (point);
 coordinates : LIST [1:3] OF length_measure;
END_ENTITY;
(*
```

Определения атрибутов.

coordinates[1]: первая координата точки;

coordinates[2]: вторая координата точки (не существует в одномерном пространстве);

coordinates[3]: третья координата точки (не существует в одномерном и двумерном пространстве);

SELF\geometric_representation_item.dim: размерность пространства, в котором задана сущность *point*. Настоящий производный атрибут унаследован из супертипа элемента геометрического представления. Для декартовой точки он определен по числу координат в перечне.

6.1.9.3 Сущность direction

Сущность *direction* определяет общее направление вектора в двумерном или трехмерном пространстве. Фактические величины компонентов не оказывают влияния на заданное направление: важными являются только отношения $x : y : z$ или $x : y$.

Примечание 1 — Компоненты данной сущности не нормированы. Если требуется задание единичного вектора, то он нормируется перед использованием.

Спецификация на языке EXPRESS:

```
*)
ENTITY direction
 SUBTYPE OF (geometric_representation_item);
 direction_ratios : LIST [2:3] OF REAL;
WHERE
 WR1  : SIZEOF(QUERY(tmp <* direction_ratios | tmp <> 0.0)) > 0;
 api_WR2: NOT((ABS(direction_ratios[1]) < EPS) AND
        (ABS(direction_ratios[2]) < EPS) AND
```

```
        (ABS(direction_ratios[3]) < EPS));
  api_WR3: NOT(((direction_ratios[1] < EPS) AND
        (direction_ratios[1] > ZERO_VALUE)) OR
        ((direction_ratios[2] < EPS) AND
        (direction_ratios[2] > ZERO_VALUE)) OR
        ((direction_ratios[3] < EPS) AND
        (direction_ratios[3] > ZERO_VALUE)));
END_ENTITY;
(*
```

Примечание 2 — В контексте схемы *api_abstract_schema* вырождение направления устанавливается дополнительными правилами типа «Где?» (WHERE RULES).

Определения атрибутов.

direction_ratio[1]: компонент по оси *X*;

direction_ratio[2]: компонент по оси *Y*;

direction_ratio[3]: компонент по оси *Z* (отсутствует в двумерном координатном пространстве);

SELF\geometric_representation_item.dim: размерность направления координатного пространства. Является наследованным атрибутом супертипа элемента геометрического представления *geometric_representation_item*; для настоящей сущности определяется количеством соотношений направления *direction_ratio* в перечне.

Комментарии к спецификации:

WR1: модуль вектора направления должен быть больше 0;

api_WR2: модуль вектора направления должен быть не меньше допуска EPS;

api_WR3: значения *direction_ratio* не могут лежать между допуском EPS и нулем *ZERO_value*.

6.1.9.4 Сущность vector

Сущность *vector* определяет вектор через его направление и модуль. Значения атрибута *magnitude* задают модуль вектора.

Примечание 1 — Модуль вектора не должен вычисляться по компонентам атрибута *orientation*. Данная форма представления вектора устраняет проблемы при вычислениях.

Пример — Вектор с модулем 2,0 мм, равнонаклоненный к координатным осям, может быть представлен атрибутом направления со значениями (1.0, 1.0, 1.0).

Спецификация на языке EXPRESS:

```
*)
ENTITY vector
 SUBTYPE OF (geometric_representation_item);
 orientation : direction;
 magnitude   : length_measure;
WHERE
 WR1    : magnitude >= 0.0;
 api_WR2 : MAX >= magnitude;
 api_WR3 : magnitude >= EPS;
END_ENTITY;
(*
```

Примечание 2 — В контексте схемы *api_abstract_schema* факт вырождения сущности *vector* устанавливается дополнительным правилом «Где?» (WHERE RULES).

Определения атрибутов:

orientation: направление вектора;

magnitude: модуль вектора. Все векторы с модулем, равным 0.0, считаются равными вне зависимости от значения атрибута *orientation*;

SELF\geometric_representation_item.dim: размерность пространства, в котором определена сущность *vector*.

Комментарии к спецификации:

WR1: модуль должен быть положительным или равным 0;

api_WR2: модуль не может превышать значение MAX;

api_WR3: модуль не может быть меньше допуска EPS.

6.1.9.5 Сущность placement

Сущность *placement* задает расположение геометрического элемента с учетом координатной системы его геометрического контекста. Данная сущность задает расположение элемента и его ориентацию (в случае использования подтипов размещения оси).

Спецификация на языке EXPRESS:

```
*)
ENTITY placement
 SUPERTYPE OF (ONEOF(axis1_placement,
           axis2_placement_2d,
           axis2_placement_3d))
 SUBTYPE OF (geometric_representation_item);
 location : cartesian_point;
END_ENTITY;
(*
```

Определение атрибута:

location: геометрическое положение ссылочной точки (например, центр окружности) рассматриваемого геометрического элемента.

6.1.9.6 Сущность axis1_placement

Сущность *axis1_placement* определяет направление и расположение в трехмерном пространстве одной оси. Сущность *axis1_placement* определяется положением точки (наследованной из размещения супертипа) и направлением оси. Это либо задаваемое направление для атрибута *axis*, либо направление (0.0, 0.0, 1.0) по умолчанию. Фактическое направление оси задается производным атрибутом *z*.

Примечание — В контексте схемы *api_abstract_schema* направление оси определяется числом.

Спецификация на языке EXPRESS:

```
*)
ENTITY axis1_placement
 SUBTYPE OF (placement);
 axis : OPTIONAL direction;
DERIVE
 z  : direction := NVL(normalise(axis), direction([0.0,0.0,1.0]));
WHERE
 WR1   : SELF\geometric_representation_item.dim = 3;
 api_WR2 : EXISTS (SELF.axis) ;
END_ENTITY;
(*
```

Определения атрибутов:

SELF\placement.location: ссылочная точка на оси;

axis: направление локальной оси *Z*;

z: стандартное направление локальной оси *Z*;

SELF\geometric_representation_item.dim: размерность пространства сущности *axis1_placement* (определяемая по расположению) всегда равна 3.

Комментарии к спецификации:

WR1: размерность координатного пространства равна 3;

api_WR2: должно быть задано направление оси *axis*.

6.1.9.7 Сущность axis2_placement_2d

Сущность *axis2_placement_2d* задает расположение и ориентацию в двумерном пространстве двух взаимно перпендикулярных осей. Сущность *axis2_placement_2d* определена в терминах точки (наследованной из супертипа размещения) и оси. Сущность может быть использована для позиционирования и ориентации объекта в двумерном пространстве и задания координатной системы указанного размещения. Сущность включает точку, формирующую начало координатной системы. Вектор направления завершает определение локальной координатной системы. Атрибут *ref_direction* определяет направление оси *X*. Направление оси *Y* определяется по направлению оси *X*.

Примечание — В контексте схемы *api_abstract_schema* задается значение для ссылочного направления *ref_direction*.

Спецификация на языке EXPRESS:

```
*)
ENTITY axis2_placement_2d
 SUBTYPE OF (placement);
 ref_direction : OPTIONAL direction;
DERIVE
  p      : LIST [2:2] OF direction := build_2axes(ref_direction);
WHERE
 WR1  : SELF\geometric_representation_item.dim = 2;
 api_WR2: EXISTS(SELF.ref_direction);
END_ENTITY;
(*
```

Определения атрибутов:

SELF\placement.location: пространственное положение ссылочной точки, определяющей начало координат ассоциированной локальной координатной системы;

ref_direction: направление, используемое для определения направления локальной оси *X*. Если атрибут *ref_direction* опущен, то данное направление берется из геометрической координатной системы;

p. набор осей локальной координатной системы;

p[1]: стандартное направление оси *X*. Задается как (1.0, 0.0), если атрибут ref_direction опущен;

p[2]: стандартное направление оси *Y*. Задается ортогонально оси *p[1]*.

Комментарии к спецификации:

WR1: размерность пространства *axis2_placement_2d* равна 2;

api_WR2: должно существовать значение ref_direction.

6.1.9.8 Сущность axis2_placement_3d

Сущность *axis2_placement_3d* задает расположение и ориентацию в трехмерном пространстве двух взаимно перпендикулярных осей. Сущность *axis2_placement_3d* определена в терминах точки (наследованной из размещения супертипа) и двух ортогональных осей. Данная сущность может использоваться для позиционирования и ориентации неосесимметричных объектов в пространстве и для определения локальной координатной системы. Сущность включает точку, задающую начало координатной системы. Для завершения размещения координатной системы необходимы два направляющих вектора. Атрибут *axis* задает направление оси *Z*, атрибут *ref_direction* аппроксимирует направление оси *X*.

Примечание 1 — Пусть атрибут *z* задает направление оси *Z*, а сущность *a* аппроксимирует направление оси *X*. Существует два метода (математически идентичных, но различных в вычислительном отношении) для расчета направлений осей *X* и *Y*:

a) вектор *a* проектируется на плоскость, определенную начальной точкой *P* и вектором *z*. Если указанный вектор задает направление оси *X* по формуле $x = <a — (a \times z)\ z>$, то направление оси *Y* задается формулой $y = <z \times x>$;

b) если направление оси *Y* вычисляется по формуле $y = <z \times a>$, то направление оси *X* задается формулой $x = <y \times z>$.

Расчеты показывают, что первый метод более устойчив к вычислениям, чем второй. Поэтому он используется в настоящем стандарте.

Для конических кривых и элементарных поверхностей локальная координатная система, на которую производится ссылка с помощью параметрических уравнений, вычисляется по данным сущности *axis2_placement_3d*.

Примечание 2 — В контексте схемы *api_abstract_schema* значения задаются для атрибутов *axis* и *ref_direction*.

Спецификация на языке EXPRESS:

```
*)
ENTITY axis2_placement_3d
 SUBTYPE OF (placement);
 axis      : OPTIONAL direction;
 ref_direction : OPTIONAL direction;
DERIVE
 p      : LIST [3:3] OF direction := build_axes(axis,ref_direction);
WHERE
 WR1  : SELF\placement.location.dim = 3;
```

```
  WR2  : (NOT (EXISTS (axis))) OR (axis.dim = 3);
  WR3  : (NOT (EXISTS (ref_direction))) OR (ref_direction.dim = 3);
  WR4  : (NOT (EXISTS (axis))) OR (NOT (EXISTS (ref_direction))) OR
          (cross_product(axis,ref_direction).magnitude > 0.0);
  api_WR5: EXISTS (axis) AND EXISTS (ref_direction);
  api_WR6: cross_product(axis, ref_direction).magnitude >= EPS;
END_ENTITY;
(*
```

Примечание 3 — В контексте схемы *api_abstract_schema* сущность *api_WR6* устанавливает вырожденный случай для сущности *axis2_placement_3d*.

Определения атрибутов:

SELF\placement.location: пространственное положение опорной точки и начала координат ассоциированной локальной координатной системы;

axis: точное направление локальной оси *Z*;

ref_direction: направление, задаваемое для локальной оси *X*. При необходимости для поддержания ортогональности соответствующей оси *Y* выполняется ее корректировка. Если атрибут *axis* и/или атрибут *ref_direction* опущены, то указанные направления берутся из геометрической координатной системы;

p. оси локальной координатной системы. При необходимости направления этих осей задаются атрибутами с соответствующими значениями по умолчанию;

p[1]: стандартное направление локальной оси *X*;

p[2]: стандартное направление локальной оси *Y*;

p[3]: стандартное направление локальной оси *Z*.

Примечание 4 — Интерпретация атрибутов представлена на рисунке 3.

Комментарии к спецификации:

WR1: размерность пространства атрибута SELF\placement.location равна 3;

WR2: размерность пространства атрибута *axis* равна 3;

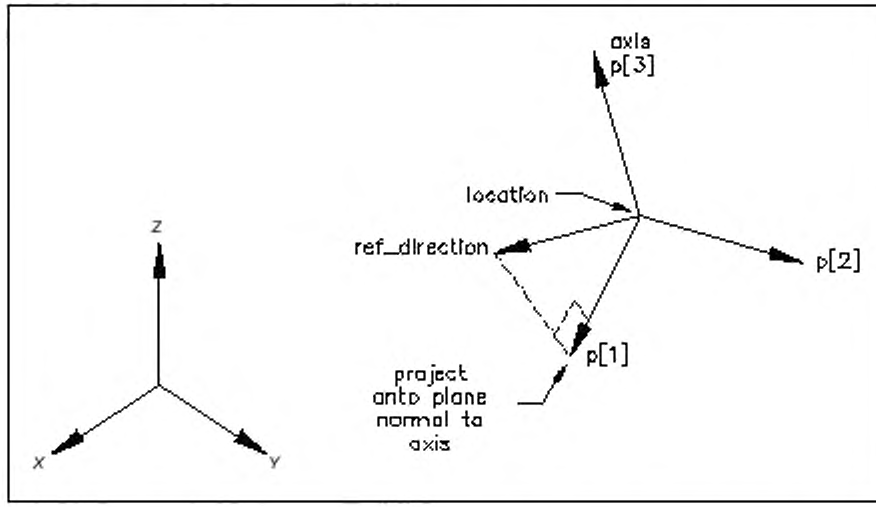
WR3: размерность пространства атрибута *ref_direction* равна 3;

WR4: оси, задавамые атрибутами *axis* и *ref_direction*, не могут быть параллельными или непараллельными (требование функции build_axes);

api_WR5: оси *axis* и ref_direction должны существовать;

api_WR6: модуль векторного произведения векторов *axis* и *direction* не превышает MAX;

api_WR7: модуль векторного произведения векторов *axis* и *direction* не может быть меньше EPS.



Axis — ось; location — размещение (начало координат); ref_direction — ссылочное направление, project onto plane normal to axis — проекция на плоскость, перпендикулярную оси

Рисунок 3 — Сущность axis2_placement_3D

6.1.10 Определения сущностей схемы API_ABSTRACT_SCHEMA: сущности геометрических кривых

В настоящем подразделе установлены ресурсы групповых сущностей для кривых, определенных в ИСО 10303-42 и являющихся частью схемы *api_abstract_schema*. Данные сущности, за исключением сущности *line*, не могут быть созданы непосредственно с помощью функций интерфейса. Они могут быть созданы только косвенно для представления особых сущностей интерфейса.

Сущность *line* может быть создана непосредственно для задания описаний других сущностей с учетом ограничений. Тем не менее сущность *line* рассматривается как математическая сущность, она ассоциирована со стилем *null_style*.

6.1.10.1 Сущность curve

Сущность *curve* интерпретируется как след точки в координатном пространстве.

Спецификация на языке EXPRESS:

```
*)
ENTITY curve
 SUPERTYPE OF (ONEOF(line, conic, surface_curve))
 SUBTYPE OF (geometric_representation_item);
END_ENTITY;
(*
```

Примечание — В контексте схемы *api_abstract_schema* интерфейсом создаются только линии, конические линии и линии поверхности кривой. Таким образом, супертип оказывается отсеченным.

Дополнительные комментарии:

IP1: кривые соединяются как дуги;

IP2: длина дуги больше 0;

api_IP3: длина дуги больше EPS.

6.1.10.2 Сущность line

Сущность *line* задает бесконечную кривую с постоянным направлением касательной. Линия определяется точкой и направлением. Положительным направлением линии является направление вектора *dir*.

Кривые параметризуются следующим образом:

$$P = pnt;$$

$$V = dir;$$

$$\lambda(u) = P + uV.$$

Параметрический диапазон: $-\infty < u < \infty$.

Примечание — В контексте схемы *api_abstract_schema* линии могут создаваться непосредственно для задания определений других сущностей с учетом ограничений. Они могут создаваться неявно как базовые кривые *basis_curve* из отрезков кривых *trimmed_curve* для построения сущностей *api_line* в интерфейсе прикладного программирования.

Спецификация на языке EXPRESS:

```
*)
ENTITY line
 SUBTYPE OF (curve);
 pnt : cartesian_point;
 dir : vector;
WHERE
 WR1 : dir.dim = pnt.dim;
END_ENTITY;
(*
```

Определения атрибутов:

pnt: расположение линии;

dir: направление линии. Модуль и единицы измерения длины вектора *dir* оказывают влияние на параметризацию линии;

SELF\geometric_representation_item.dim: размерность координатного пространства для линии. Наследованный атрибут для супертипа элемента геометрического представления.

Комментарий к спецификации:

WR1: Сущности *pnt* и *dir* являются двумерными или трехмерными.

6.1.10.3 Сущность bounded_curve

Сущность *bounded_curve* задает кривую конечной длины с неидентифицируемыми конечными точками.

Примечание — В контексте схемы *api_abstract_schema* интерфейсом создаются только полилинии, отрезанные кривые, конечные кривые на поверхности и комбинированные кривые. Таким образом, супертипы отсекаются.

Спецификация на языке EXPRESS:

```
*)
ENTITY bounded_curve
 SUPERTYPE OF (ONEOF(polyline, trimmed_curve, bounded_surface_curve,
           composite_curve))
 SUBTYPE OF (curve);
END_ENTITY;
(*
```

6.1.10.4 Сущность trimmed_curve

Сущность *trimmed_curve* определяет конечную кривую путем выбора сегмента, расположенного между двумя заданными точками ассоциированной базовой кривой. Сама базовая кривая остается неизменной. На одну базовую кривую может ссылаться несколько отрезков кривых *trimmed_curve*. Точки вычленения кривой определяются:

- по значению параметра;
- по геометрическому положению;
- обоими способами.

По крайней мере один из этих методов используют на каждом конце кривой. Атрибут положительного направления обхода контура *sense* обеспечивает недвусмысленное определение любого сегмента замкнутой кривой, например, окружности. Комбинация заданного положительного направления и двух упорядоченных конечных точек обеспечивает определение четырех конечных направленных сегментов, соединяющих две различные точки на окружности (или другой замкнутой кривой). Для обеспечения однозначности также используют свойство периодичности (цикличности) диапазона значений параметров.

Пример 1 — Угол 370° эквивалентен углу 10° (за счет периода, равного 360°).

Сущность отрезка кривой *trimmed_curve* имеет режим параметризации, наследованный из режима параметризации заданной ссылочной базовой кривой. Более точно значения параметра *s* отрезка кривой *trimmed_curve* получаются из параметра *t* базовой кривой следующим образом:

если значение атрибута sense равно «true», то $s = t - t_1$;

если значение атрибута sense равно «false», то $s = t_2 - t$.

Здесь t_1 — это значение, заданное параметром *trim_1* или значением параметра, соответствующим точке *point_1*, а t_2 — это значение, заданное параметром *trim_2* или значением параметра, соответствующим точке *point_2*. Результирующий отрезок кривой имеет значение параметра *s* в диапазоне от 0 (для первой точки вычленения) до $|t_2 - t_1|$ (для второй точки вычленения).

Примечание 1 — Если базовая кривая замкнута, то может оказаться необходимым дать приращения величинам t_1 и t_2 за счет параметрической длины для обеспечения соответствия значению флажка направления обхода кривой.

Пример 2 — Если значение sense_agreement «true» и $t_2 < t_1$, то t_2 увеличивают на период.

Пример 3 — Если значение sense_agreement «false» и $t_1 < t_2$, то t_1 увеличивают на период.

Спецификация на языке EXPRESS:

```
*)
ENTITY trimmed_curve
 SUPERTYPE OF (ONEOF (api_line, api_circular_arc, api_elliptical_arc,
            api_hyperbolic_arc, api_parabolic_arc))
 SUBTYPE OF (bounded_curve);
 basis_curve       : curve;
 trim_1         : SET[1:2] OF trimming_select;
 trim_2         : SET[1:2] OF trimming_select;
 sense_agreement     : BOOLEAN;
 master_representation : trimming_preference;
WHERE
 WR1: (HIINDEX(trim_1) = 1) XOR (TYPEOF(trim_1[1]) <> TYPEOF(trim_1[2]));
 WR2: (HIINDEX(trim_2) = 1) XOR (TYPEOF(trim_2[1]) <> TYPEOF(trim_2[2]));
END_ENTITY;
(*
```

Примечание 2 — В контексте схемы *api_abstract_schema* определяются особые подтипы для спецификации диапазонов значений или целей некоторых функций интерфейса.

Примечание 3 — В контексте схемы *api_abstract_schema* главное представление *master_representation* должно зависеть от реализации.

Примечание 4 — В контексте схемы *api_abstract_schema*, если базовая кривая *basis_curve* замкнута, то замкнутый отрезок кривой *trimmed_curve* (соответствующая полной базовой кривой) представляется с помощью параметра идентификации точек вычленения.

Пример 4 — Дуга окружности, заданная значениями атрибутов sense_agreement = false, trim_1 = 450 и trim_2 = 90, является замкнутой дугой окружности, направленной по часовой стрелке. Точки ее вычленения заданы пересечением базовой кривой basis_curve окружности circle и осью Y, определенной сущностью axis2_placement.

Определения атрибутов.

basis_curve: отрезок кривой. Для кривых с многократным представлением любое значение параметров, заданное точками вычленения *trim_1* или *trim_2*, ссылается только на главное представление базовой кривой *basis_curve*;

trim_1: первая точка вычленения, которая может быть описана либо как декартова точка (*point_1*), либо как действительное значение параметра (*parameter_1* = t_1), либо обоими способами;

trim_2: вторая точка вычленения, которая может быть описана либо как декартова точка (*point_2*), либо как действительное значение параметра (*parameter_2* = t_2), либо обоими способами;

sense_agreement: флажок, указывающий, совпадает направление отрезка кривой *trimmed_curve* с заданным направлением базовой кривой *basis_curve* или нет.

Атрибут *sense_agreement* равен «true», если обход кривой производится в направлении увеличения значения параметра;

атрибут *sense_agreement* в противном случае равен «false».

Для открытой кривой атрибут *sense_agreement* равен «false», если $t_1 > t_2$. Если $t_2 > t_1$, то атрибут *sense_agreement* равен «true». В данном случае информация о знаке направления обхода является избыточной (она существенна только для замкнутой кривой);

master_representation: используется, если параметр и точка присутствуют либо в начале, либо в конце кривой, — это указывает на предпочтительность формы. Множественное представление обеспечивает связь данных более чем одной формы, при этом данные могут быть геометрически идентичными.

Примечание 5 — Атрибут главного представления *master_representation* не гарантирует, что множественные формы действительно идентичны, он указывает предпочтительную форму. Это реализуется разработчиком данных. Все характеристики (например, параметризация, области и результаты вычислений для сущностей, имеющих множественные представления) являются производными главного представления. Использование прочих представлений — это компромисс для практических приложений.

Комментарии к спецификации:

WR1: задается одно значение точки вычленения *trim_1* или задаются две настройки различного типа (точка и параметр);

WR2: задается одно значение точки вычленения *trim_2* или задаются две настройки различного типа (точка и параметр).

Дополнительные комментарии:

IP1: если значение параметра и десятичная точка *cartesian_point* существуют для атрибутов *trim_1* или *trim_2*, то они должны быть совместимыми, то есть базовая кривая *basis_curve*, вычисленная для указанного значения параметра, должна совпадать с заданной точкой;

IP2: если десятичная точка *cartesian_point* задана атрибутом *trim_1* или атрибутом *trim_2*, то она должна лежать на базовой кривой *basis_curve*;

IP3: в случае замкнутой базовой кривой *basis_curve*, когда оба значения *parameter_1* и *parameter_2* существуют, они должны быть совместимы со значением флажка направления обхода кривой, то есть *sense* = (*parameter_1* < *parameter_2*);

IP4: если оба значения *parameter_1* и *parameter_2* существуют, то *parameter_1* <> *parameter_2*;

IP5: если значения параметра описаны атрибутами *trim_1* или *trim_2*, то указанные значения должны находиться внутри параметрического диапазона базовой кривой *basis_curve*.

6.1.10.5 Сущность composite_curve

Сущность *composite_curve* задает комбинацию различных кривых, соединенных последовательно. Отдельные сегменты такой комбинированной кривой *curve* сами могут быть комбинированными *composite_curve_segment*. Параметризация комбинированной кривой состоит в аккумулировании параметрических диапазонов сегментов ограниченной кривой *bounded_curve*. Первый сегмент параметризуется в интервале от 0 до l_1, а для *i* 3 2 соответственно *i*-й сегмент параметризуется в интервале:

$$\text{от} \quad \sum_{k\ 1}^{k\ i-1} lk \quad \text{до} \quad \sum_{k\ 1}^{k\ i} lk,$$

где l_k — параметрическая длина (то есть разность между максимальным и минимальным значением параметра) *k*-го сегмента кривой.

Примечание 1 — В контексте схемы *api_abstract_schema* комбинированная кривая *composite_curve* используется либо для представления интерфейса сущности контура *api_contour*, либо для представления граничной кривой *boundary_curve* ограниченной поверхности *curve_bounded_surface*. Оба варианта кривой должны быть плоскими, замкнутыми и несамопересекающимися *self_intersecting*.

Спецификация на языке EXPRESS:

```
*)
ENTITY composite_curve
 SUBTYPE OF (bounded_curve);
 segments    : LIST [1:?] OF composite_curve_segment;
 self_intersect : LOGICAL;
DERIVE
 n_segments   : INTEGER := SIZEOF(segments);
 closed_curve  : BOOLEAN
        := segments[n_segments].transition <> discontinuous;
WHERE
 WR1  : ((NOT closed_curve) AND (SIZEOF(QUERY(temp <* segments |
          temp.transition = discontinuous)) = 1)) OR
        ((closed_curve) AND (SIZEOF(QUERY(temp <* segments |
          temp.transition = discontinuous)) = 0));
 api_WR2: closed_curve ;
 api_WR3: NOT self_intersect ;
END_ENTITY;
(*
```

Примечание 2 — В контексте схемы *api_abstract_schema* дополнительное правило «Где?» (WHERE RULES) устанавливает требования к замкнутой комбинированной кривой composite_curve, созданной интерфейсом.

Определения атрибутов:

n_segment: число компонентов комбинированной кривой;

segments: конечные кривые сегментов, их переходы и флажки.

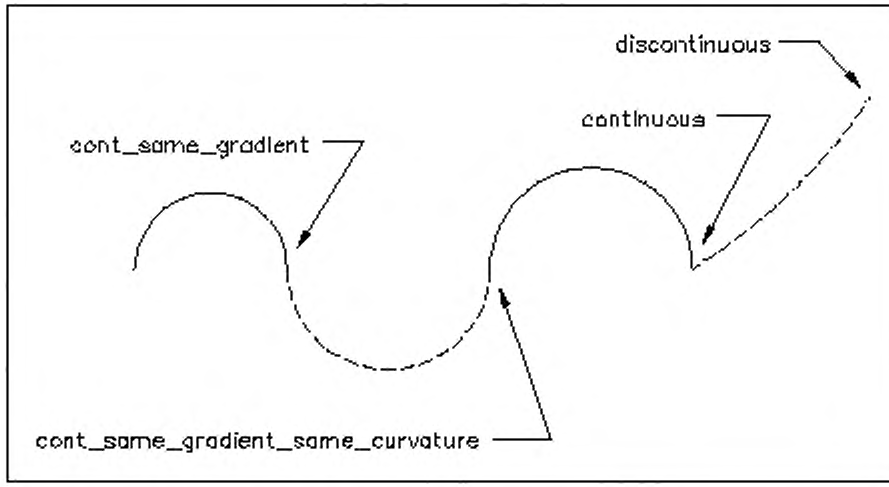
Атрибут перехода для последнего сегмента определяет переход от конца последнего сегмента к началу первого. Атрибут перехода может принимать значение «discontinuous», если кривая не замкнута (см. раздел 6.1.3.2);

self_intersect: указывает, является ли кривая самопересекающейся (справочно);

dim: размерность координатного пространства комбинированной кривой. Это наследуемый атрибут супертипа элемента геометрического представления;

closed_curve: указывает, является ли данная кривая замкнутой. Это определяется по коду перехода на последний сегмент.

Примечание 3 — Дополнительная информация об указанных атрибутах приведена на рисунке 4.



Discontinuous — разрывный; continuous — непрерывный; cont_same_gradient — стыковка сегментов с сохранением направления касательной; cont_same_gradient_same_curvature — стыковка сегментов с сохранением направления касательной и значения кривизны

Рисунок 4 — Комбинированная кривая

Комментарии к спецификации:

WR1: код перехода может быть разрывным только для последнего сегмента открытой кривой;

api_WR2: комбинированная кривая *composite_curve* должна быть замкнутой;

api_WR3: комбинированная кривая *composite_curve* не должна быть самопересекающейся.

Дополнительные комментарии:

IP1: атрибут same_sense для каждого сегмента описывает соответствие положительных направлений обхода. При переходе в направлении, указанном атрибутом *same_sense*, сегменты должны присоединяться концом к концу;

api_IP2: комбинированная кривая *composite_curve* должна быть плоской.

6.1.10.6 Сущность composite_curve_segment

Сущность *composite_curve_segment* задает конечную кривую вместе с информацией о переходе, используемой при создании комбинированной кривой *composite_curve*.

Примечание — В контексте схемы *api_abstract_schema* сегменты комбинированной кривой *composite_curve_segment* автоматически вычисляются интерфейсом при создании контура *api_contour* или плоской поверхности *api_planar_surface*. Поэтому переход не должен быть разрывным.

Спецификация на языке EXPRESS:

```
*)
ENTITY composite_curve_segment;
 transition  : transition_code;
 same_sense  : BOOLEAN;
 parent_curve : curve;
INVERSE
 using_curves : BAG[1:?] OF composite_curve FOR segments;
```

```
WHERE
 WR1  : ('API_ABSTRACT_SCHEMA.BOUNDED_CURVE' IN TYPEOF(parent_curve));
 api_WR2: (transition = continuous) OR (transition = cont_same_gradient);
END_ENTITY;
(*
```

Определения атрибутов.

transition: состояние перехода (то есть свойство геометрической непрерывности перехода из конечной точки предшествующего сегмента в начальную точку последующего сегмента) для комбинированной кривой *composite_curve*;

same_sense: индикатор, показывающий, согласуется ли направление обхода сегмента с направлением обхода первичной кривой *parent_curve*. Если значение атрибута *same_sense* «false», то точка с наивысшим значением параметра есть начальная точка сегмента;

parent_curve: конечная кривая *bounded_curve*, определяющая геометрический сегмент;

using_curve: множество комбинированных кривых *composite_curve*, использующих данный сегмент *composite_curve_segment*. Данное множество не должно быть пустым.

Комментарии к спецификации:

WR1: первичная кривая *parent_curve* должна быть ограниченной кривой *bounded_curve*;

api_WR2: переход *transition* может быть либо непрерывным, либо иметь дополнительно непрерывную касательную *cont_same_gradient*.

6.1.10.7 Сущность surface_curve

Сущность *surface_curve* задает кривую на поверхности. Рассматриваемая кривая является кривой в трехмерном пространстве *curve_3d*. Она может соответствовать кривой *pcurve* в двумерном параметрическом пространстве на поверхности. Способность данной кривой ссылаться на одну или две кривые на поверхности *pcurve_or_surface* позволяет настоящей сущности определять либо кривую на одной поверхности, либо кривую пересечения двух ассоциированных поверхностей. «Шов» на замкнутой поверхности также может быть представлен настоящей сущностью. В данном случае каждая ассоциированная геометрия *associated_geometry* представляет собой *pcurve*, лежащую на одной поверхности. Каждая *pcurve* (если таковая существует) параметризуется по направлению обхода, соответствующему кривой *curve_3d*. Кривая на поверхности параметризуется непосредственно либо как *curve_3d*, либо как *pcurve* в соответствии с атрибутом *master_representation*.

Примечание 1 — В контексте схемы *api_abstract_schema surface_curve*, автоматически вычисленные интерфейсом, при создании плоской поверхности *api_planar_surface* ссылаются на сущность плоскости.

Спецификация на языке EXPRESS:

```
*)
ENTITY surface_curve
 SUPERTYPE OF (bounded_surface_curve)
 SUBTYPE OF (curve);
 curve_3d       : curve;
 associated_geometry  : LIST[1:2] OF pcurve_or_surface;
 master_representation : preferred_surface_curve_representation;
DERIVE
 basis_surface     : SET[1:2] OF surface
           := get_basis_surface(SELF);
WHERE
 WR1   : curve_3d.dim = 3;
 WR2   : ('GEOMETRY_SCHEMA.PCURVE' IN TYPEOF(associated_geometry[1])) OR
                 (master_representation <> pcurve_s1);
 WR3   : ('GEOMETRY_SCHEMA.PCURVE' IN TYPEOF(associated_geometry[2])) OR
                 (master_representation <> pcurve_s2);
 WR4   : NOT ('GEOMETRY_SCHEMA.PCURVE' IN TYPEOF(curve_3d));
 api_WR5 : master_representation = curve_3D;
 api_WR6 : SIZEOF(SELF.associated_geometry) = 1;
 api_WR7 : 'API_ABSRACT_SCHEMA.PLANE' IN
                 TYPEOF (SELF.associated_geometry [1]);
 api_WR8 : SELF.associated_geometry[1] :=: SELF.basis_surface;
END_ENTITY;
(*
```

Определения атрибутов:

curve_3d: кривая, являющаяся трехмерным представлением кривой на поверхности *surface_curve*;

associated_geometry: перечень одной или двух *pcurve* (поверхностей), определяющих поверхности, ассоциированные с кривой на поверхности. Два элемента данного перечня указывают, что кривая имеет связь с двумя поверхностями, которые могут быть не разделены. Если кривая выбрана, то она идентифицирует поверхность и связанную с ней базовую кривую в параметрическом пространстве данной поверхности.

Примечание 2 — В контексте схемы *api_abstract_schema* ассоциированная геометрия *associated_geometry* ссылается на сущность *plane* при создании плоской поверхности *api_planar_surface* интерфейса прикладного программирования;

master_representation: указание на главное представление. Оно задает кривую, определяющую уникальную параметризацию кривой на поверхности.

Главное представление использует одно из значений атрибутов *curve_3d*, *pcurve_s1* или *pcurve_s2*. Таким образом, можно указать предпочтительность *3D-curve*, первой *pcurve* или второй *pcurve* в ассоциированном геометрическом перечне соответственно. Множественные представления способствуют установлению связи данных нескольких форм, даже если указанные данные могут быть геометрически идентичными.

Примечание 3 — Главное представление *master_representation* признает нецелесообразность обеспечения идентичности нескольких форм и позволяет выбрать предпочтительные формы. Выбор выполняет разработчик данных. Все характеристики (например, параметризация, область и результаты вычислений для сущностей, имеющих множественное представление) являются производными главного представления. Любое использование других представлений является компромиссом для практических приложений.

Примечание 4 — В контексте схемы *api_abstract_schema* главным представлением *master_representation* должна быть *curve_3d*;

basis_surface: поверхность, на которой лежит *surface_curve*. Она задается первым элементом перечня *associated_geometry*.

Примечание 5 — В контексте схемы *api_abstract_schema* данной поверхностью является плоскость *plane* для плоской поверхности *api_planar_surface* интерфейса прикладного программирования.

Комментарии к спецификации:

WR1: curve_3d определяется в трехмерном пространстве;

WR2: pcurve_s1 является главным представлением, если первый элемент перечня *associated_geometry* является *pcurve*;

WR3: pcurve_s2 является главным представлением, если второй элемент перечня *associated_geometry* является *pcurve*. При этом кривая *pcurve_s2* не должна рассматриваться, если перечень *associated_geometry* содержит только один элемент;

WR4: curve_3d не должна быть *pcurve*;

api_WR5: главное представление *master_representation* должно быть *curve_3D*;

api_WR6: перечень *associated_geometry* должен содержать только один элемент;

api_WR7: если перечень *associated_geometry* содержит уникальный элемент, то он должен быть сущностью *plane*;

api_WR8: производная сущности *basis_surface* должна быть той же сущностью, что и уникальная сущность, содержащаяся в перечне *associated_geometry*.

Дополнительные комментарии:

IP1: если *curve_3d* и *pcurve* существуют, то они должны представлять одно и то же множество математических точек (то есть они должны совпадать геометрически, но могут отличаться способом параметризации);

IP2: curve_3d и любая ассоциированная с ней *pcurve* должны быть согласованы с учетом положительного направления их обхода.

6.1.10.8 Сущность composite_curve_on_surface

Сущность *composite_curve_on_surface* задает набор сегментов кривых на поверхности. Каждый сегмент должен лежать на базовой поверхности. Это может быть:

- *surface_curve*;
- *pcurve*;
- *composite_curve_on_surface*.

Примечание 1 — Комбинированная кривая на поверхности *composite_curve_on_surface* может быть включена как атрибут *parent_curve* сегмента комбинированной кривой *composite_curve_segment*, который является подтипом ограниченной кривой *bounded_curve*.

Примечание 2 — В контексте схемы *api_abstract_schema* каждый сегмент должен быть кривой на поверхности *surface_curve*.

Необходимо обеспечить по крайней мере непрерывность взаимного положения примыкающих сегментов. Комбинированные кривые *composite_curve* параметризуются путем объединения параметрических диапазонов сегментов. Диапазон значений параметра первого сегмента от 0 до l_1 и, соответственно, диапазон значений параметра *i*-го сегмента:

$$\text{от} \quad \sum_{k=1}^{k=i-1} lk \quad \text{до} \quad \sum_{k=1}^{k=i} lk,$$

где l_k — параметрическая длина (то есть разность между максимальным и минимальным значением параметра) *k*-го сегмента кривой.

Спецификация на языке EXPRESS:

```
*)
ENTITY composite_curve_on_surface
 SUPERTYPE OF (boundary_curve)
 SUBTYPE OF (composite_curve);
DERIVE
 basis_surface : SET[0:2] OF surface := get_basis_surface(SELF);
WHERE
 WR1  : SIZEOF(basis_surface) > 0;
 WR2  : constraints_composite_curve_on_surface(SELF);
 api_WR3: SIZEOF(QUERY(temp<*SELF\composite_curve.segments |
           'GEOMETRY_SCHEMA.PCURVE' IN TYPEOF (temp.parent_curve)
           )  -- end of query
         ) = 0;
END_ENTITY;
(*
```

Определения атрибутов:

basis_surface: поверхность, на которой определена рассматриваемая комбинированная кривая;

SELF\composite_curve.n_segment: число сегментов комбинированной кривой;

SELF\composite_curve.segments: конечные кривые сегментов, их сопряжения и положительные направления обхода. Сопряжение последнего сегмента комбинированной кривой задает порядок перехода от конца последнего сегмента к началу первого. Настоящий элемент может принимать значение «discontinuous», что указывает на разрывность кривой.

Примечание 3 — В контексте схемы *api_abstract_schema* наследованное правило «Где?» (WHERE RULES) используемого интерфейса прикладного программирования гарантирует, что сопряжение (переход) не является разрывным;

SELF\composite_curve.self_intersect: указывает, является ли кривая самопересекающейся;

SELF\composite_curve.dim: размерность координатного пространства комбинированной кривой *composite_curve*;

SELF\composite_curve.closed_curve: указывает, является ли кривая замкнутой.

Комментарии к спецификации:

WR1: множество *basis_surface* должно содержать по крайней мере одну поверхность. Это гарантирует, что все сегменты относятся к кривой на одной и той же поверхности;

WR2: каждый рассматриваемый сегмент должен относиться к *pcurve* или к *surface_curve* либо к сегменту *composite_curve_on_surface*;

api_WR3: никакой из сегментов на может относиться к *pcurve*.

Дополнительный комментарий:

IP1: каждая первичная кривая *parent_curve*, на которую производится ссылка сегментом *composite_curve_on_surface*, должна быть комбинированной кривой на поверхности *curve_on_surface* и ограниченной кривой *bounded_curve*.

6.1.10.9 Сущность bounded_surface_curve

Сущность *bounded_surface_curve* задает особый подтип кривой поверхности *surface_curve*, имеющей свойства ограниченной кривой *bounded_curve*.

Спецификация на языке EXPRESS:

```
*)
ENTITY bounded_surface_curve
 SUBTYPE OF (surface_curve, bounded_curve);
WHERE
api_WR1 : ('API_ABSTRACT_SCHEMA.BOUNDED_CURVE' IN
      TYPEOF (SELF\surface_curve.curve_3d);
END_ENTITY;
(*
```

Комментарий к программе:

api_WR1: атрибут *curve_3d* супертипа *surface_curve* должен быть *bounded_curve*.

6.1.11 Определения сущностей схемы API_ABSTRACT_SCHEMA: геометрические конические сущности

В настоящем подразделе установлен групповой ресурс сущностей для *curves*, определенных в ИСО 10303-42 и являющихся частью схемы *api_abstract_schema*. Рассматриваемые сущности не могут быть созданы непосредственно с помощью функций интерфейса. Они могут быть созданы только косвенно и представляют собой особые сущности интерфейса.

6.1.11.1 Сущность conic

Сущность *conic* задает плоскую кривую пересечения плоскости с конусом.

Коническая кривая определяется в терминах ее внутренних геометрических свойств.

Сущность *conic* всегда имеет локальную координатную систему *placement*, определенную сущностью *axis2_placement*. Параметрическое представление сущности установлено в терминах настоящей локальной координатной системы *placement*.

Примечание — В контексте схемы *api_abstract_schema* конические кривые создаются только как базовые кривые для *api_circular_arc*, *api_elliptical_arc*, *api_hyperbolic_arc* и *api_parabolic_arc* в интерфейсе прикладного программирования.

Спецификация на языке EXPRESS:

```
*)
ENTITY conic
 ABSTRACT SUPERTYPE OF (ONEOF(circle, ellipse, hyperbola, parabola))
 SUBTYPE OF (curve);
 position: axis2_placement;
WHERE
 api_WR1: SIZEOF( USEDIN(SELF,
          'API_ABSTRACT_SCHEMA.TRIMMED_CURVE.BASIS_CURVE') ) = 1;
END_ENTITY;
(*
```

Определение атрибута:

position: расположение и ориентация конических кривых. Дальнейшие подробности интерпретации настоящего атрибута заданы для индивидуальных подтипов.

Комментарий к спецификации:

api_WR1: каждая коническая кривая используется как *basis_curve* одной сущностью отрезка кривой *trimmed_curve*.

6.1.11.2 Сущность circle

Сущность *circle* задает окружность по радиусу, центру и заданному положительному направлению обхода:

C = position.location;

x = position.p[1];

y = position.p[2];

z = position.p[3];

R = радиус.

Окружность задается параметром u по формуле:

$$\lambda(u) = C + R((\cos u)x + (\sin u)y).$$

Диапазон параметризации: $0° \le u \le 360°$.

В локальной координатной системе *placement*, определенной выше, уравнение окружности имеет вид $C = 0$, где:

$$C(x, y, z) = x^2 + y^2 - R^2.$$

Положительное направление обхода окружности задается сущностью *circle* в любой ее точке в направлении вектора касательной T к окружности:

$$T = (-C_y, C_x, 0).$$

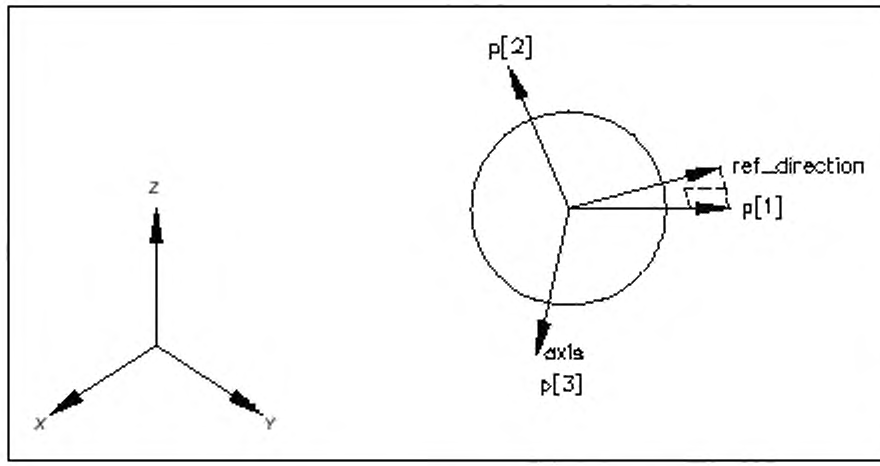
Примечание 1 — Дуга окружности определяется сущностью *trimmed_cirve* в совокупности с сущностью *circle*.

Примечание 2 — В контексте схемы *api_abstract_schema* дуга окружности *circular_arc* определяется *api_circular_arc*.

Примечание 3 — В контексте схемы *api_abstract_schema* окружность создается только интерфейсом прикладного программирования как сущность *basis_curve* для *api_circular_arc*.

Спецификация на языке EXPRESS:

```
*)
ENTITY circle
  SUBTYPE OF (conic);
  radius : positive_lenght_measure;
END_ENTITY;
(*
```



Ref_direction — ссылочное направление, axis — ось

Рисунок 5 — Окружность

Определения атрибутов:

SELF\conic.position.location: настоящий унаследованный атрибут определяет центр окружности;

radius: радиус окружности должен быть положительным.

Примечание 4 — Интерпретация атрибутов приведена на рисунке 5.

6.1.11.3 Сущность ellipse

Сущность *ellipse* задает коническую кривую, определенную двумя (большой и малой) полуосями эллипса, положением эллипса (центром или серединой отрезка, соединяющего фокусы эллипса) и ориентацией эллипса. Данные эллипса:

C = position.location;
x = position.p[1];
y = position.p[2];
z = position.p[3];
R_1 = semi_axis_1;
R_2 = semi_axis_2.

Эллипс задается параметром u по формуле:

$$\lambda(u) = C + (R_1 \cos u)x + (R_2 \sin u)y.$$

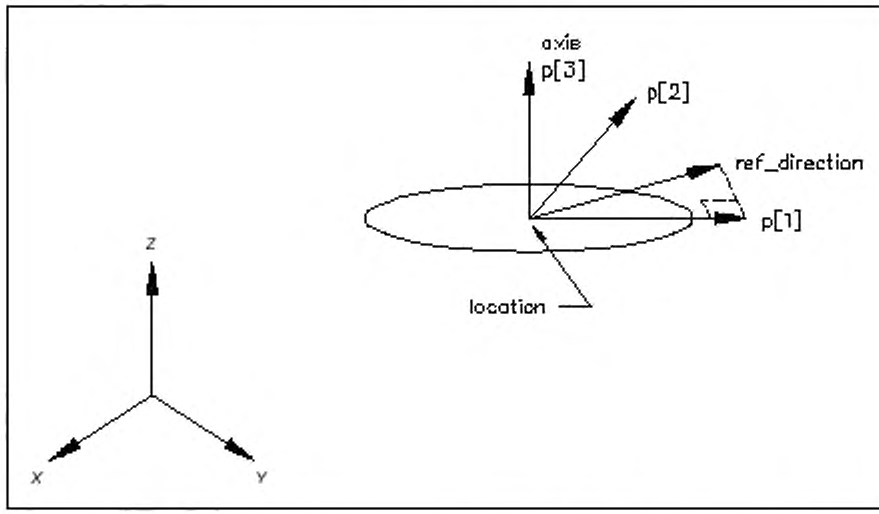
Диапазон параметризации: $0° \le u \le 360°$.

В локальной координатной системе *placement*, определенной выше, уравнение эллипса имеет вид $C = 0$, где:

$$C(x, y, z) = x^2/R_1^2 + y^2/R_2^2 - 1.$$

Положительное направление обхода эллипса в любой его точке задается касательным вектором $T = (-C_y, C_x, 0)$.

Примечание 1 — В контексте схемы *api_abstract_schema* эллипс может быть создан интерфейсом прикладного программирования только как базовая кривая для *api_elliptical_arc*.



Axis — ось; ref_direction — ссылочное направление; location — центр эллипса

Рисунок 6 — Эллипс

Спецификация на языке EXPRESS:

```
*)
ENTITY ellipse
 SUBTYPE OF (conic);
 semi_axis_1 : positive_length_measure;
 semi_axis_2 : positive_length_measure;
END_ENTITY;
(*
```

Определения атрибутов.

SELF\conic.position: атрибут *conic.position.location* задает центр эллипса, атрибут *conic.position.p[1]* задает направление первой полуоси эллипса *semi_axis_1*;

semi_axis_1: первая полуось эллипса, должна быть положительной;

semi_axis_2: вторая полуось эллипса, должна быть положительной.

Примечание 2 — Интерпретация атрибутов указана на рисунке 6.

6.1.11.4 Сущность hyperbola

Сущность *hyperbola* задает коническую кривую, определенную большим и малым радиусами гиперболы, положением гиперболы (серединой отрезка, соединяющего два фокуса гиперболы) и ориентацией гиперболы. Данные гиперболы:

C = position.location;

x = position.p[1];

y = position.p[2];

z = position.p[3];

R_1 = semi_axis;

R_2 = semi_imag_axis.

Гипербола задается параметром u по формуле.

$$X(u) = C + (R_1 \cosh u)x + (R_2 \sinh u)y.$$

Диапазон параметризации: $-\infty < u < \infty$.

В локальной координатной системе placement, определенной выше, гипербола представляется уравнением $C = 0$, где:

$$C(x, y, z) = x^2/R_1^2 - y^2/R_2^2 - 1.$$

Положительное направление обхода гиперболы в любой ее точке определяется вектором касательной T:

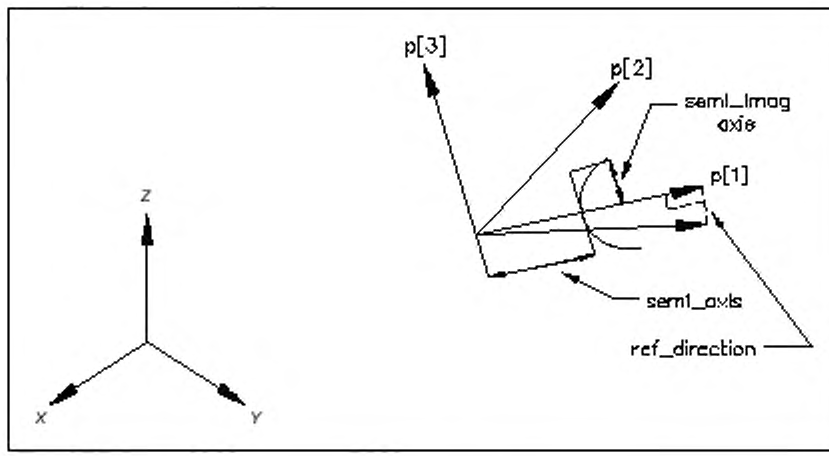
$$T = (-C_y, C_x, 0).$$

На рисунке 7 изображена только одна ветвь гиперболы для положительных значений координаты x.

Примечание 1 — В контексте схемы *api_abstract_schema* в интерфейсе прикладного программирования гипербола может быть создана только как базовая кривая для *api_hyperbolic_arc*.

Спецификация на языке EXPRESS:

```
*)
ENTITY hyperbola
 SUBTYPE OF (conic);
 semi_axis    : positive_length_measure;
 semi_imag_axis : positive_length_measure;
END_ENTITY;
(*
```



Semi_imag axis — мнимая полуось гиперболы; semi_axis — действительная полуось гиперболы; ref_direction — ссылочное направление

Рисунок 7 — Гипербола

Определения атрибутов:

SELF\conic.position: расположение и ориентация конической кривой. Атрибут *conic.position.location* задает центр гиперболы, атрибут *conic.position.p[1]* задает направление действительной полуоси гиперболы. Указана только одна ветвь гиперболы для положительных значений координаты *position.p[1]*;

semi_axis: длина действительной полуоси гиперболы. Она положительна и равна половине минимального расстояния между ветвями гиперболы;

semi_imag_axis: длина мнимой полуоси гиперболы, показана для положительных значений координаты *y*.

Примечание 2 — Атрибуты гиперболы указаны на рисунке 7.

Комментарии к спецификации:

WR1: длина действительной полуоси *semi_axis*, должна быть больше нуля.

WR2: длина мнимой полуоси *semi_imag_axis*, должна быть больше нуля.

6.1.11.5 Сущность parabola

Сущность *parabola* задает коническую кривую, определенную фокусным расстоянием, положением вершины параболы и ориентацией параболы.

Данные параболы:

C = position.location;

x = position.p[1];

y = position.p[2];

z = position.p[3];

F = focal_dist.

Парабола задается параметром *u* по формуле:

$$X(u) = C + F(u^2 x + 2uy).$$

Диапазон параметризации: $-\infty < u < \infty$.

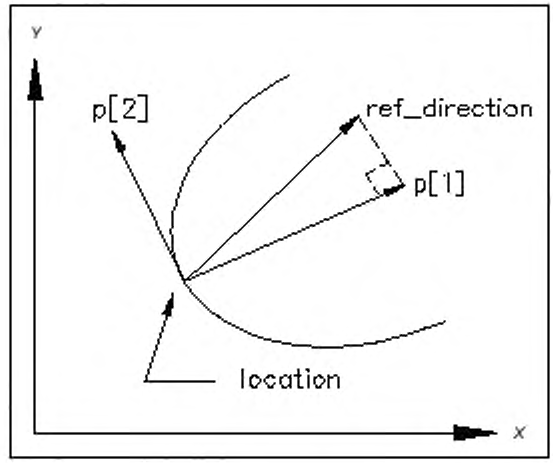
В локальной координатной системе, определенной выше, парабола представлена уравнением $C = 0$, где:

$$C(x, y, z) = 4Fx - y^2.$$

Положительное направление обхода кривой в любой ее точке задается вектором касательной *T*:

$$T = (-C_y, C_x, 0).$$

Примечание 1 — В контексте схемы *api_abstract_schema* в интерфейсе прикладного программирования парабола может быть создана только как базовая кривая для *parabolics_arc*.



Ref_direction — ссылочное направление, location — вершина параболы

Рисунок 8 — Парабола

Спецификация на языке EXPRESS:

```
*)
ENTITY parabola
 SUBTYPE OF (conic);
 focal_dist : length_measure;
WHERE
 WR1: focal_dist <> 0.0;
END_ENTITY;
(*
```

Определения атрибутов:

SELF\conic.position: расположение и ориентация кривой. Атрибут *conic.position.location* задает вершину параболы, атрибут *conic.position.p[1]* задает ось симметрии параболы;

focal_dist: расстояние фокуса от вершины параболы.

Примечание 2 — Атрибуты параболы представлены на рисунке 8.

Комментарий к спецификации:

WR1: фокусное расстояние не должно быть равно нулю.

6.1.12 Определение сущностей схемы API_ABSTRACT_SCHEMA: базовые кривые интерфейса прикладного программирования

В настоящем подразделе установлены сущности кривых интерфейса прикладного программирования, вычисляемых и генерируемых интерфейсом посредством спецификаций с учетом ограничений. Интерфейс имеет утилиты для определения характеристик указанных сущностей. Задание ориентации рассматриваемых сущностей (с помощью точек вычленения *trim_1* и *trim_2* в соответствии с соглашением о задании положительного направления обхода кривой *sense_agreement*) устраняет неоднозначность геометрических построений. Указанные сущности должны существовать в целевой моделирующей системе. Поэтому процесс моделирования для их реализации не определяется.

6.1.12.1 Сущность api_line

Сущность *api_line* задает отрезок кривой *trimmed_curve* линейного сегмента. Определяется сущностью *trimmed_cirve* в совокупности с сущностью *line*.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_line
 SUBTYPE OF (trimmed_curve) ;
WHERE
 api_WR1 : 'API_ABSTRACT_SCHEMA.LINE' IN
                TYPEOF (SELF\TRIMMED_CURVE.BASIS_CURVE);
END_ENTITY;
(*
```

Примечание 1 — Данная сущность интерфейса прикладного программирования позволяет задать диапазон значений некоторых функций интерфейса.

Примечание 2 — В контексте схемы *api_abstract_schema* главное представление *master_representation* должно зависеть от реализации.

Примечание 3 — Данная сущность может быть использована как *trimmed_curve*.

Определения атрибутов:

SELF\trimmed_curve.basis_curve: линия, в пределах которой берется отрезок;

SELF\trimmed_curve.trim_1: первая точка вычленения, описываемая либо как декартова точка *cartesian_point* (*point_1*), либо действительным значением параметра (*parameter_1* = t_1), либо обоими способами;

SELF\trimmed_curve.trim_2: вторая точка вычленения, описываемая либо как декартова точка *cartesian_point* (*point_2*), либо как действительное значение параметра (*parameter_2* = t_2), либо обоими способами;

SELF\trimmed_curve.sense_agreement: флажок, указывающий, согласуется или нет направление обхода отрезка кривой с направлением обхода базовой кривой;

master_representation: главное представление, где и параметр, и точка, присутствующие на обоих концах кривой, указывают предпочтительную форму. Множественное представление позволяет устанавливать связь данных более чем одной формы, даже если эти данные могут оказаться геометрически идентичными.

Комментарий к спецификации:

api_WR1: basis_curve для кривой *trimmed_curve* должна быть линия.

Дополнительный комментарий:

api_IP1: *api_line* должна быть не меньше допуска EPS и не больше установленного максимального значения MAX.

6.1.12.2 Сущность api_circular_arc

Сущность *api_circular_arc* задает отрезок кривой *trimmed_curve* для одного сегмента окружности. Определяется с помощью сущности *trimmed_cirve* в совокупности с сущностью *circle*.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_circular_arc
 SUBTYPE OF (trimmed_curve);
WHERE
 api_WR1 : 'API_ABSTRACT_SCHEMA.CIRCLE' IN
                  TYPEOF (SELF\TRIMMED_CURVE.BASIS_CURVE);
END_ENTITY;
(*
```

Примечание 1 — Данная сущность интерфейса прикладного программирования задает диапазон значений некоторых функций интерфейса.

Примечание 2 — В контексте схемы *api_abstract_schema* главное представление *master_representation* должно зависеть от реализации.

Примечание 3 — Данная сущность может быть применена как сущность *trimmed_curve*.

Примечание 4 — Если начальная и конечная точки отрезка дуги окружности *api_circular_arc* совпадают, то сущность *api_circular_arc* задает окружность с указанием положительного направления ее обхода в соответствии с соглашением *sense_agreement*.

Определения атрибутов:

SELF\trimmed_curve.basis_curve: окружность, из которой вычленяется дуга;

SELF\trimmed_curve.trim_1: начальная точка отрезка, описываемая либо как декартова точка *cartesian_point* (*point_1*), либо действительным значением параметра (*parameter_1* = t_1), либо обоими способами;

SELF\trimmed_curve.trim_2: конечная точка отрезка, описываемая либо как декартова точка *cartesian_point* (*point_2*), либо как действительное значение параметра (*parameter_2* = t_2), либо обоими способами;

SELF\trimmed_curve.sense_agreement: флажок, указывающий, согласуется или нет направление обхода отрезка кривой с направлением обхода базовой кривой;

master_representation: главное представление, где и параметр, и точка, присутствующие на обоих концах кривой, указывают предпочтительную форму. Множественное представление позволяет устанавливать связь данных более чем одной формы, даже если эти данные могут оказаться геометрически идентичными.

Комментарий к спецификации:

api_WR1: базовой кривой для отрезка кривой должна быть окружность.

Дополнительный комментарий:

api_IP1: длина дуги окружности *api_circular_arc* должна быть не меньше допуска EPS.

6.1.13 Определения сущностей схемы API_ABSTRACT_SCHEMA: дуги конических кривых интерфейса прикладного программирования

В настоящем подразделе установлены сущности дуг конических кривых интерфейса прикладного программирования, генерируемых с помощью функций интерфейса. Если функция, создающая дугу конической кривой, запущена, то кривая *conic* сначала создается как базовая кривая *basis_curve* дуги конической кривой, а затем указанная дуга конической кривой создается как подтип отрезка кривой *trimmed_curve*.

Сущности, определенные в настоящем разделе, задают диапазон значений некоторых функций интерфейса.

Примечание — Дуги конических кривых могут быть использованы как вычленения кривых *trimmed_curve*.

Если сущности конических кривых не существуют в целевых моделирующих системах, то моделирование выполняется интерфейсом. Настоящее моделирование производится для каждой сущности путем интерполяции. Для интерполяции в моделируемой сущности берутся две заданные конечные точки и некоторое количество внутренних точек, определенных в строке задания числа узлов интерполяции *interpolation_nodes_number* таблицы статуса интерфейса. Интерполирующие кривые должны быть непрерывными, с непрерывными касательными. Они должны сохранять касательные сущности в обеих конечных точках. Кривая, используемая для интерполяции, зависит от реализации. Это может быть, например, дуга окружности *circular_arc*. Тип кривой может быть определен внутри целевой моделирующей системы или интерфейса (например, кривые Безье). Выбор промежуточных точек интерполяции также зависит от реализации. Единственным требованием к промежуточным точкам является их равномерное (в некотором смысле) распределение.

Строка задания числа узлов интерполяции *interpolation_nodes_number* таблицы статуса интерфейса может запрашиваться прикладной программой. Число узлов интерполяции может быть меньше или равно некоторому максимальному значению *max_interpolation_nodes_number*, определенному в таблице описаний интерфейса. Указанное значение *max_interpolation_nodes_number* должно быть больше или равно 1.

6.1.13.1 Сущность api_elliptical_arc

Сущность *api_elliptical_arc* задает отрезок кривой *trimmed_curve* для сегмента эллипса *ellipse*. Определяется сущностью *trimmed_cirve* в совокупности с сущностью *ellipse*.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_elliptical_arc
 SUBTYPE OF (trimmed_curve);
WHERE
 api_WR1 : 'API_ABSTRACT_SCHEMA.ELLIPSE' IN
                  TYPEOF (SELF\TRIMMED_CURVE.BASIS_CURVE);
END_ENTITY;
(*
```

Примечание 1 — Настоящая сущность интерфейса прикладного программирования задает диапазон значений некоторых функций интерфейса.

Примечание 2 — В контексте схемы *api_abstract_schema* главное представление *master_representation* должно зависеть от реализации.

Примечание 3 — Настоящая сущность может быть применена как сущность *trimmed_curve*.

Примечание 4 — Если начальная и конечная точки *api_elliptical_arc* совпадают, то сущность *api_elliptical_arc* задает эллипс целиком с положительным направлением обхода, соответствующим соглашению *sense_agreement*.

Определения атрибутов.

SELF\trimmed_curve.basis_curve: эллипс, в пределах которого вычленяется дуга;

SELF\trimmed_curve.trim_1: начальная точка отрезка, описываемая либо как декартова точка *cartesian_point* (*point_1*), либо действительным значением параметра (*parameter_1* = t_1), либо обоими способами,

SELF\trimmed_curve.trim_2: конечная точка отрезка, описываемая либо как декартова точка *cartesian_point* (*point_2*), либо как действительное значение параметра (*parameter_2* = t_2), либо обоими способами,

SELF\trimmed_curve.sense_agreement: флажок, указывающий, согласуется или нет направление обхода отрезка кривой с направлением обхода базовой кривой;

master_representation: главное представление, где и параметр, и точка, присутствующие на обоих концах кривой, указывают предпочтительную форму. Множественное представление позволяет устанавливать связь данных более чем одной формы, даже если эти данные могут оказаться геометрически идентичными.

Комментарий к спецификации:

api_WR1: базовой кривой для отрезка кривой должен быть эллипс.

Дополнительный комментарий:

IP1: длина дуги эллипса *api_elliptical_arc* должна быть не меньше допуска EPS.

6.1.13.2 Сущность api_hyperbolic_arc

Сущность *api_hyperbolic_arc* задает отрезок кривой *trimmed_curve* сегмента гиперболы. Определяется сущностью *trimmed_cirve* в совокупности с сущностью *hyperbola*.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_hyperbolic_arc
  SUBTYPE OF (trimmed_curve);
WHERE
 api_WR1 : 'API_ABSTRACT_SCHEMA.HYPERBOLA' IN
                  TYPEOFF (SELF\TRIMMED_CURVE.BASIS_CURVE);
END_ENTITY;
(*
```

Примечание 1 — Настоящая сущность интерфейса прикладного программирования задает диапазон значений некоторой функции интерфейса.

Примечание 2 — В контексте схемы *api_abstract_schema* главное представление *master_representation* должно зависеть от реализации.

Примечание 3 — Настоящая сущность может быть применена как сущность *trimmed_curve*.

Определения атрибутов:

SELF\trimmed_curve.basis_curve: гипербола, в пределах которой вычленяется сегмент;

SELF\trimmed_curve.trim_1: начальная точка отрезка, описываемая либо как декартова точка *cartesian_point* (*point_1*), либо действительным значением параметра (*parameter_1* = t_1), либо обоими способами,

SELF\trimmed_curve.trim_2: конечная точка отрезка, описываемая либо как декартова точка *cartesian_point* (*point_2*), либо как действительное значение параметра (*parameter_2* = t_2), либо обоими способами;

SELF\trimmed_curve.sense_agreement: флажок, указывающий, согласуется или нет направление обхода отрезка кривой с направлением обхода базовой кривой;

master_representation: главное представление, в котором параметр и точка, присутствующие на обоих концах кривой, указывают предпочтительную форму. Множественное представление позволяет устанавливать связь данных более чем одной формы, даже если эти данные могут оказаться геометрически идентичными.

Комментарий к спецификации:

api_WR1: базовая кривая для отрезка кривой должна быть гиперболой.

Дополнительный комментарий:

api_IP1: длина дуги гиперболы *api_hyperbolic_arc* должна быть не меньше допуска EPS.

6.1.13.3 Сущность api_parabolic_arc

Сущность *api_parabolic_arc* задает отрезок кривой *trimmed_curve* сегмента параболы. Определяется сущностью *trimmed_cirve* в совокупности с сущностью *parabola*.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_parabolic_arc
 SUBTYPE OF (trimmed_curve);
WHERE
 api_WR1 : 'API_ABSTRACT_SCHEMA.PARABOLA' IN
                  TYPEOF (SELF\TRIMMED_CURVE.BASIS_CURVE);
END_ENTITY;
(*
```

Примечание 1 — Настоящая сущность интерфейса прикладного программирования задает диапазон значений некоторой функции интерфейса.

Примечание 2 — В контексте схемы *api_abstract_schema* главное представление *master_representation* должно зависеть от реализации.

Примечание 3 — Настоящая сущность может быть применена как сущность *trimmed_curve*.

Определения атрибутов:

SELF\trimmed_curve.basis_curve: парабола, в пределах которой вычленяется сегмент;

SELF\trimmed_curve.trim_1: начальная точка отрезка, описываемая либо как декартова точка *cartesian_point* (*point_1*), либо действительным значением параметра (*parameter_1* = t_1), либо обоими способами;

SELF\trimmed_curve.trim_2: конечная точка отрезка, описываемая либо как декартова точка *cartesian_point* (*point_2*), либо как действительное значение параметра (*parameter_2* = t_2), либо обоими способами;

SELF\trimmed_curve.sense_agreement: флажок, указывающий, согласуется или нет направление обхода отрезка кривой с направлением обхода базовой кривой;

master_representation: главное представление, где и параметр, и точка, присутствующие на обоих концах кривой, указывают предпочтительную форму. Множественное представление позволяет устанавливать связь данных более чем одной формы, даже если эти данные могут оказаться геометрически идентичными.

Комментарий к спецификации:

api_WR1: базовая кривая для отрезка кривой должна быть параболой.

Дополнительный комментарий:

IP1: длина дуги параболы *api_parabolic_arc* должна быть не меньше допуска EPS.

6.1.14 Определения сущностей схемы API_ABSTRACT_SCHEMA: сущности кривых

Настоящий подраздел устанавливает две сущности кривых, создаваемых с помощью функций интерфейса: сущность *polyline* (групповой ресурс сущностей, определенный стандартом ИСО 10303-42 и являющийся частью схемы *api_abstract_schema*) и сущность *api_contour* интерфейса прикладного программирования.

6.1.14.1 Сущность polyline

Сущность *polyline* создает ограниченную кривую *bounded_curve*, состоящую из $n - 1$ линейных сегментов, определенных n точками $P_1, P_2, ..., P_n$.

При этом i-й сегмент кривой параметризуется следующим образом:

$$\lambda(u) = P_i(i - u) + P_{i+1}(u + 1 - i), \text{ для } 1 \le i \le n - 1,$$

где $i - 1 \le u \le i$ — параметрический диапазон целочисленного параметра $0 \le u \le n - 1$.

Примечание 1 — Если сущность *polyline* не существует в CAD, то она должна моделироваться посредством соединения линий. Максимальное количество узлов полилинии, допустимое в данной реализации интерфейса, не меньше величины, установленной в разделе 9.

Примечание 2 — В контексте схемы *api_abstract_schema* длина каждого линейного сегмента не должна быть меньше EPS или больше MAX.

Спецификация на языке EXPRESS:

```
*)
ENTITY polyline
 SUBTYPE OF (bounded_curve);
 points : LIST [2:?] OF cartesian_point;
END_ENTITY;
(*
```

Определение атрибута:

points: узлы, задающие полилинии.

Дополнительный комментарий:

api_IP1: длина каждого линейного сегмента не должна быть меньше EPS или больше MAX.

6.1.14.2 Сущность api_contour

Сущность *api_contour* задает несамопересекающуюся ориентированную плоскую замкнутую комбинированную кривую composite_curve, построенную интерфейсом из базовых сущностей, дуг конических кривых и/или полилиний. Контур *api_contour* разрезает плоскость на два подмножества. Ограниченное подмножество является внутренним. Интерфейс должен гарантировать замкнутость контура. При этом сущности, получающиеся в результате геометрического преобразования контура с помощью интерфейса, остаются замкнутыми.

Сущность *api_contour* определена прикладной программой как неупорядоченный перечень сущностей кривых. Любые базовые сущности (линии *api_line*, дуги окружности *api_circular_arc*), конические дуги и сущности *polyline* могут быть использованы для определения контура *api_contour* в интерфейсе прикладного программирования при условии:

1) для любой крайней точки одной сущности существует только одна крайняя точка другой сущности в окрестности нулевого радиуса *ZERO_value*,

2) кривая, полученная путем соединения соседних крайних точек, заданных сущностью, должна быть плоской, замкнутой и несамопересекающейся.

Указанные положения прежде всего проверяются интерфейсом. При этом перечень сущностей, определенный прикладной программой, логически переупорядочивается внутри интерфейса. Первой из переупорядоченных сущностей является первая сущность исходного перечня. Второй является сущность, содержащая только одну крайнюю точку в окрестности крайней точки первого сущности, и т. д. Начальными точками первой сущности могут быть начальная и конечная точки контура.

Если оба условия выполнены, то контур *api_contour* вычисляется интерфейсом. Данный процесс выполняется в два этапа: 1) моделирование некоторых сущностей; 2) результирующая сущность корректируется, чтобы гарантировать замыкание контура *api_contour*.

1. Контуры *api_contour* определяются для построения заполненных областей комментариев *annotation_fill_area*, плоских поверхностей *api_planar_surface* и твердых тел *solid bodies*. Таким образом, некоторые сущности не могут быть использованы для представлений контура (например, когда они не поддерживаются CAD). Только сущности базовых кривых (например, вычленения прямых *api_line*, дуги окружностей *api_circular_arc*) допускаются любым интерфейсом для представления контура. Если какие-либо сущности кривых, используемые функциями генерации контура, не допускаются интерфейсом для представления контура, то указанные сущности моделируются особой установленной процедурой.

Примечание 1 — Сущности, допустимые для представления контура, определены записями *contour_entities* таблицы описаний интерфейса (см. раздел 8.1). Максимальное количество сущностей на каждый контур *api_contour* для заданной реализации интерфейса должно быть не меньше величины, описанной в разделе 9.

2. Замкнутый контур *api_contour* конструируется в следующем порядке:

1) первая кривая переупорядоченного перечня дублируется вместе с базовой кривой при условии, что это отрезок кривой;

2) первый сегмент комбинированной кривой *composite_curve_segment* строится с помощью указанной дублированной кривой как *parent_curve*. При этом значение атрибута направления обхода кривой *same_sense* равно «true»;

3) вычисляется направление *direction* касательного вектора текущего сегмента комбинированной кривой *composite_curve_segment* в его конечной точке. Вычисляется также направление касательного вектора последующего сегмента кривой (в переупорядоченном перечне кривых) в его начальной точке в окрестности конечной точки предшествующего сегмента комбинированной кривой *composite_curve_segment*. Таким образом, последующая кривая для последней кривой (переупорядоченного перечня кривых) оказывается первой кривой;

4) если оба направления *direction* параллельны, то переход *transition* текущего сегмента комбинированной кривой *composite_curve_segment* на последующий сегмент устанавливается из условия сохранения касательной сущностью *cont_same_gradient*, в противном случае она устанавливается из условия обеспечения непрерывности кривой сущностью *continuous*;

5) вплоть до окончания перечня кривых каждый *composite_curve_segment* вычисляется аналогично:

a) кривая дублируется базовой кривой при условии, что это отрезок кривой;

b) сегмент комбинированной кривой *composite_curve_segment* строится с помощью данной дублированной кривой как сущность *parent_curve*. Указанный сегмент *composite_curve_segment* называется текущим *composite_curve_segment*;

c) если первая крайняя точка дублированной кривой находится в окрестности крайней точки предшествующего сегмента комбинированной кривой *composite_curve_segment*, то значение атрибута направления обхода кривой *same_sense* текущего сегмента *composite_curve_segment* равно «true». В противном случае его значение равно «false». Данный атрибут определяет ориентацию текущего сегмента *composite_curve_segment*, а также его начало и конец;

d) если начальная точка текущего сегмента *composite_curve_segment* не совпадает с конечной точкой предшествующего сегмента *composite_curve_segment*, то *basis_curve* для указанного сегмента *composite_curve_segment* преобразуется и обеспечивает замыкание контура *api_contour*;

е) вычисляются направления касательных векторов текущего сегмента комбинированной кривой *composite_curve_segment* в его конечной точке и последующей кривой (в переупорядоченном перечне кривых) в ее конечной точке, находящейся в окрестности указанной конечной точки текущего сегмента *composite_curve_segment*. Если оба направления параллельны, то сопряжение текущего сегмента *composite_curve_segment* с последующим устанавливается сущностью *cont_same_gradient*. В противном случае сопряжение должно быть непрерывным. Таким образом, последующей для последней кривой (переупорядоченного переченя кривых) является первая кривая;

6) если конечная точка последнего сегмента комбинированной кривой *composite_curve_segment* и начальная точка первого сегмента *composite_curve_segment* совпадают, то создается контур *api_contour*, состоящий из упорядоченного перечня вычисленных сегментов *composite_curve_segment*. Если указанные две точки не совпадают и если оба кода перехода двух последних *composite_curve_segment* являются непрерывными, то последний сегмент комбинированной кривой *composite_curve_segment* корректируется, чтобы обеспечить замыкание контура. Если контур незамкнутый и код перехода для двух последних сегментов *composite_curve_segment* определен сущностью *cont_same_gradient*, то интерфейс аппроксимирует последний сегмент комбинированной кривой *composite_curve_segment* одним или двумя сегментами *composite_curve_segment*. Процесс аппроксимации, зависящий от реализации, должен гарантировать:

- замыкание контура;
- корректировку кода перехода *transition* предшествующего сегмента *composite_curve_segment*;
- равенство значения кода перехода *transition* аппроксимирующей сущности и значения кода перехода аппроксимируемого сегмента комбинированной кривой *composite_curve_segment*;
- увеличение аппроксимируемой сущности больше EPS.

Если интерфейс не может выполнять указанную аппроксимацию, то последняя сущность немного перемещается, чтобы обеспечить замыкание контура. При этом коды перехода двух последних сегментов комбинированной кривой *composite_curve_segment* изменяются соответственно. Таким образом, зацикливание переупорядоченного перечня кривых всегда предполагает следующее: предшествующей сущностью для первой сущности является последняя сущность.

Примечание 2 — Для прикладных программ можно избежать модификации какой-либо сущности в процессе построения контура. Для этого указанные программы должны использовать только допустимые сущности контура (минимально это вычленения прямых *api_line* и дуги окружности *api_circular_arc*) и гарантировать замыкание контура.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_contour
 SUBTYPE OF (composite_curve);
END_ENTITY;
(*
```

Примечание 3 — Указанная сущность интерфейса прикладного программирования используется для задания диапазона некоторых функций интерфейса.

Примечание 4 — Указанная сущность может быть применена как сущность *composite_curve*.

Определения атрибутов:

SELF\composite_curve. segments: упорядоченный перечень *composite_curve_segment*, вычисленных интерфейсом и составляющих контур *api_contour* интерфейса прикладного программирования;

SELF\composite_curve.self_intersect: атрибут типа LOGICAL, имеющий значение «false» и указывающий на то, что контур *api_contour* — несамопересекающийся;

SELF\composite_curve.n_segment: количество сущностей, равное значению, установленному прикладной программой, и плюс один, если процесс вычислений, определенный выше, требует дополнительный сегмент *composite_curve_segment* для обеспечения замыкания контура *api_contour* интерфейсом прикладного программирования;

SELF\composite_curve.closed_curve: булев атрибут, имеющий значение «true» и указывающий на замыкание контура *api_contour*.

Дополнительный комментарий:

api_IP1: внутри контура должна помещаться окружность диаметром EPS.

6.1.15 Определения сущностей схемы API_ABSTRACT_SCHEMA: заполненные области

Настоящий подраздел объявляет групповые ресурсы сущностей заполненной области, определенные стандартом ИСО 10303-46 и являющиеся частью схемы *api_abstract_schema*. Заполненные области моделируются посредством сущности заполненных областей комментариев *annotation_fill_area*. В контексте схемы *api_abstract_schema* данная сущность допустима только для 2D-видов, то есть если интерфейс открыт с уровнем геометрической мощности *geometrical_power_level*, равным 1. Сущность *annotation_fill_area* представляет собой плоскую многосвязную фигуру, внешние и внутренние границы которой являются контурами *api_contour* интерфейса прикладного программирования. Максимальное число внутренних границ, допустимое заданной реализацией интерфейса, должно быть не меньше величины, определенной в разделе 9 настоящего стандарта. Все контуры должны лежать в одной плоскости и не должны пересекаться. Все контуры внутренних границ заполненной области должны лежать внутри внешнего контура границ и не должны пересекаться. В контексте схемы *api_abstract_schema* и *annotation_fill_area* не содержат комментариев. Они играют две вспомогательные роли:

1) область может быть заштрихована. При этом штриховка определяется сущностью *annotation_fill_area_occurrence*, задающей стиль штриховки *fill_area_style_hatching* элемента представления *annotation_fill_area*;

2) область может быть просто закрашена. Область имеет фоновый цвет, который затеняет прочие сущности. При этом глобальные значения записей таблицы статуса интерфейса равны: «on» — для записи *hidden_line*, «true» — для записи *hidden_line_involved*.

6.1.15.1 Сущность annotation_fill_area

Сущность *annotation_fill_area* задает множество замкнутых кривых со штриховкой, затенением, закрашиванием или клеточным заполнением. Данная сущность задается границами, представленными непересекающимися и несамопересекающимися замкнутыми кривыми *curve*. Указанные границы задают плоские области, заполненные в соответствии со стилем *annotation_fill_area*. Правила заполнения области:

- кривая, не окруженная какими-либо другими кривыми, является границей между незаполненной областью снаружи и заполненной областью внутри.

Примечание 1 — См. рисунок 9 а);

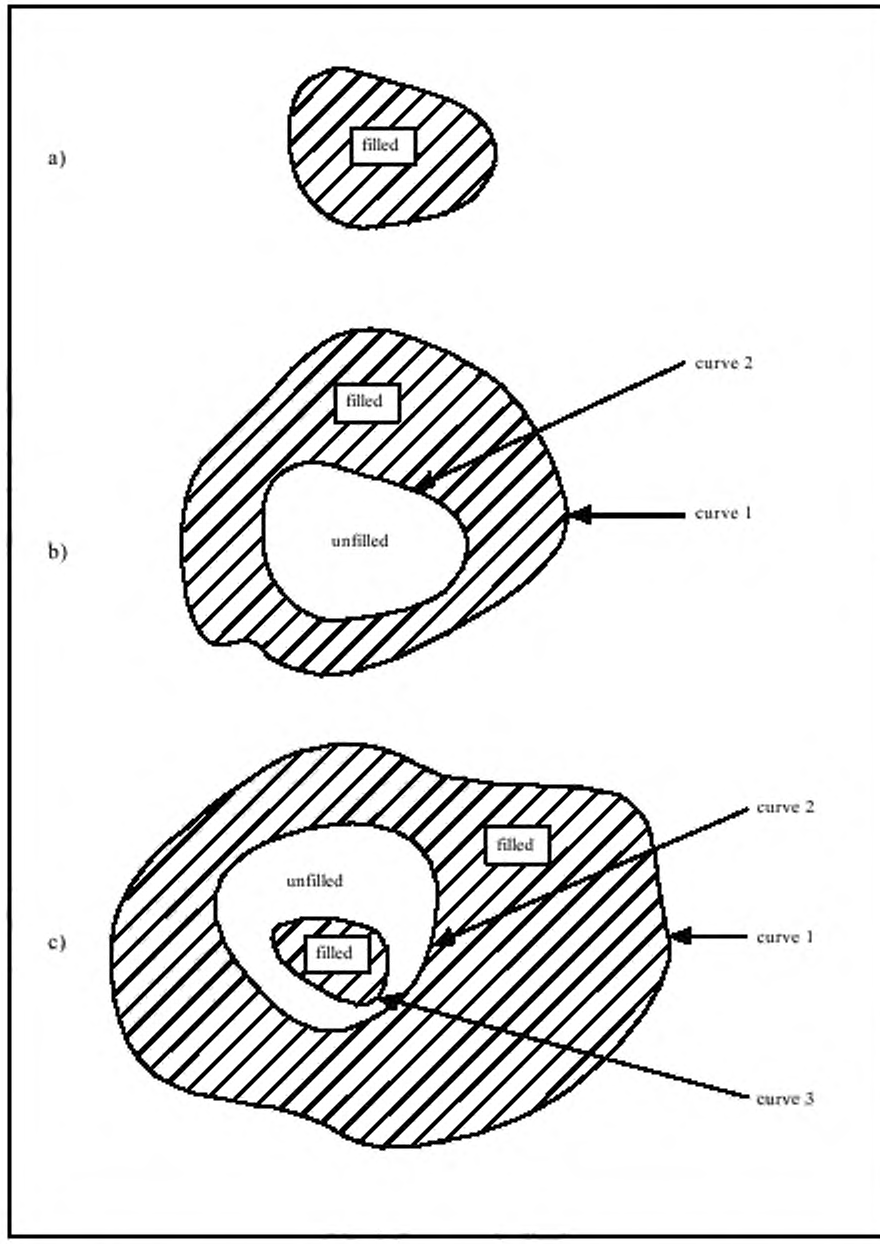
- кривая (кривая 2 на рисунке) окружает незаполненную область, если она окружена другой кривой (кривая 1 на рисунке), заполненной изнутри.

Примечание 2 — См. рисунок 9 б);

- если кривая 3 (кривая 3 на рисунке 9 а) размещается внутри кривой 2, то данная кривая окружает заполненную область.

Примечание 3 — См. рисунок 9 в);

- для каждой дополнительной кривой используется аналогичная процедура.



Filled — заполненная область, curve — кривая; unfilled — незаполненная область

Рисунок 9 — Заполнение области сущностью annotation_fill_area

Спецификация на языке EXPRESS:

```
*)
ENTITY annotation_fill_area
 SUBTYPE OF (geometric_representation_item);
 boundaries : SET [1:?] OF curve;
WHERE
api_WR1: SIZEOF(QUERY( temp <* SELF.boundaries |
           'API_ABSTRACT_SCHEMA.API_CONTOUR' IN TYPEOF (SELF) )
         ) = SIZEOF(SELF.boundaries);
END_ENTITY;
(*
```

Определение атрибута:

boundaries: множество кривых *curve*, определяющих границы заполненной области.

Комментарий к спецификации:

api_WR1: все границы должны быть контурами *api_contour*.

Дополнительные комментарии:

IP1: все кривые множества *SELF.boundaries* должны быть замкнутыми и плоскими;

IP2: если на множестве *SELF.boundaries* существует две и более кривых, то все эти кривые должны лежать в одной плоскости и не пересекаться;

IP3: оси *X* и *Y* локальной координатной системы *SELF.filling_position* должны лежать в одной плоскости с кривыми множества *SELF.boundaries [1]*;

api_IP3: если на множестве *SELF.boundaries* существует два и более контура *api_contour*, то расстояние между двумя контурами *api_contour* интерфейса прикладного программирования должно быть не меньше допуска EPS.

6.1.16 Определения сущностей схемы API_ABSTRACT_SCHEMA: сущности геометрических поверхностей

Настоящий подраздел устанавливает групповой ресурс сущностей поверхностей *surface*, определенных стандартом ИСО 10303-42 и являющихся частью схемы *api_abstract_schema*. Указанные сущности не могут быть созданы непосредственно с помощью функций интерфейса. Они могут быть созданы только косвенно и представляют особые сущности интерфейса.

6.1.16.1 Сущность surface

Сущность *surface* задает множество соединенных точек в трехмерном пространстве. Локально поверхность всегда является двумерной и односвязной. Поверхность не должна вырождаться в точку или (как частично, так и полностью) в кривую.

Примечание 1 — См. раздел 3.1 и подраздел 4.4.48 ИСО 10303-42.

Спецификация на языке EXPRESS:

```
*)
ENTITY surface
 SUPERTYPE OF (ONEOF(elementary_surface, bounded_surface))
 SUBTYPE OF (geometric_representation_item);
END_ENTITY;
(*
```

Примечание 2 — В контексте схемы *api_abstract_schema* допустимы только плоскости *plane* и плоские поверхности *api_planar_surface*. Таким образом, супертипы оказываются отсеченными.

Дополнительные комментарии:

IP1: площадь поверхности не может быть равна нулю;

IP2: поверхности соединяются по дугам.

6.1.16.2 Сущность elementary_surface

Сущность *elementary_surface* задает простую аналитическую поверхность с определенным параметрическим представлением.

Спецификация на языке EXPRESS:

```
*)
ENTITY elementary_surface
 SUPERTYPE OF (ONEOF(plane))
 SUBTYPE OF (surface);
 position : axis2_placement_3d;
END_ENTITY;
(*
```

Примечание — В контексте схемы *api_abstract_schema* элементарными поверхностями *elementary_surface* могут быть только плоскости *plane*. Таким образом, супертипы оказываются отсеченными.

Определение атрибута:

position: положение и ориентированность поверхности. Используется при параметризации поверхности.

6.1.16.3 Сущность plane

Сущность *plane* задает бесконечную поверхность с постоянной нормалью (плоскость). Поверхность определяется точкой на плоскости и направлением нормали к плоскости:

C = position.location;

x = position.p[1];

y = position.p[2];

z = position.p[3] = перпендикуляр к плоскости.

Поверхность задается параметрами *u*, *v* по формуле:

$$\lambda(u,v) = C + xu + yv$$

с диапазоном параметризации $-\infty < u, v < \infty$. В вышеуказанной параметризации единицы длины для ортов *x* и *y* зависят от контекста плоскости.

Спецификация на языке EXPRESS:

```
*)
ENTITY plane
 SUBTYPE OF (elementary_surface);
END_ENTITY;
(*
```

Определения атрибутов:

SELF\elementary_surface.position: расположение и ориентация поверхности. Атрибут унаследован из супертипа *elementary_surface*;

position.location: точка на плоскости;

position.p[3]: направление, задаваемое атрибутом *position.axis*, определяет перпендикуляр к плоскости.

6.1.16.4 Сущность bounded_surface

Сущность *bounded_surface* задает поверхность конечной площади с идентифицируемыми границами.

Спецификация на языке EXPRESS:

```
*)
ENTITY bounded_surface
 SUPERTYPE OF (ONEOF(curve_bounded_surface))
 SUBTYPE OF (surface);
END_ENTITY;
(*
```

Примечание — В контексте схемы *api_abstract_schema* сущность *bounded_surface* может быть создана только сущностью *curve_bounded_surface*. Таким образом, супертипы оказываются отсеченными.

Дополнительные комментарии:

IP1: ограниченная поверхность *bounded_surface* имеет конечную ненулевую площадь;

IP2: ограниченная поверхность *bounded_surface* имеет граничные кривые.

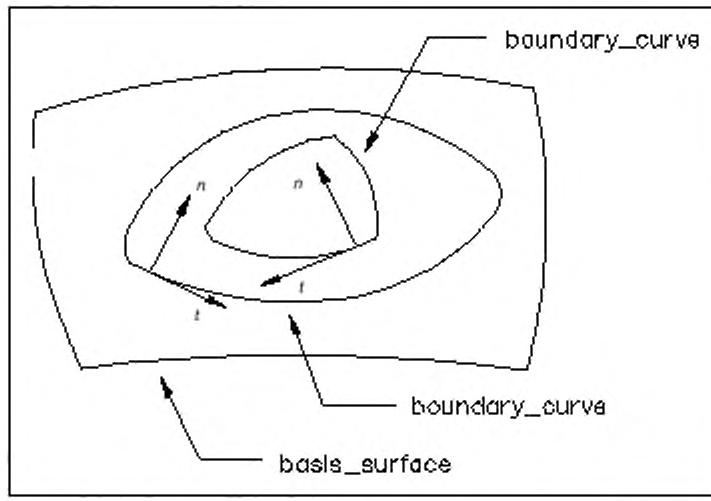
6.1.16.5 Сущность curve_bounded_surface

Сущность *curve_bounded_surface* задает параметрическую поверхность, ограниченную одной или несколькими граничными кривыми. Одна из них может быть наружной границей. Количество внутренних границ не ограничено. Наружная граница может быть определена неявно, как естественная граница поверхности. При этом значение флажка *implicit_outer* равно «true». В этом случае по крайней мере одна внутренняя граница должна быть определена. Для некоторых типов замкнутых поверхностей (например, для цилиндра) может оказаться невозможным идентифицировать какую-либо заданную границу как наружную. На базовой поверхности *basis_surface* поверхность, ограниченная кривыми *curve_bounded_surface*, представляет собой выделенный сегмент в направлении $N \times T$ из любой точки на границе, где N — нормаль к поверхности, а T — касательный вектор к граничной кривой в данной точке. Указанный сегмент выделяется дугами.

Примечание 1 — В контексте схемы *api_abstract_schema* сущность *curve_bounded_surface* может быть создана только как подтип *api_planar_surface*.

Спецификация на языке EXPRESS:

```
*)
ENTITY curve_bounded_surface
 SUBTYPE OF (bounded_surface);
 basis_surface  : surface;
 boundaries     : SET [1:?] OF boundary_curve;
 implicit_outer  : BOOLEAN;
WHERE
 WR1: NOT(implicit_outer AND
          ('API_ABSTRACT_SCHEMA.OUTER_BOUNDARY_CURVE' IN
          TYPEOF(boundaries)));
 WR2: (NOT(implicit_outer)) OR
          ('API_ABSTRACT_SCHEMA.BOUNDED_SURFACE' IN
          TYPEOF(basis_surface));
 WR3: SIZEOF(QUERY(temp <* boundaries |
          'API_ABSTRACT_SCHEMA.OUTER_BOUNDARY_CURVE' IN
          TYPEOF(temp))) <= 1;
 WR4: SIZEOF(QUERY( temp <* boundaries
          (temp\composite_curve_on_surface.basis_surface [1] :<>:
                        SELF.basis_surface))) = 0;
END_ENTITY;
(*
```



Boundary_curve — ограниченная кривая; basis_surface — базовая поверхность

Рисунок 10 — Поверхность, ограниченная кривыми

Определения атрибутов:

basis_surface: базовая поверхность, на которой выделяется ограниченная поверхность;

boundaries: граничные кривые на поверхности, отличные от неявных наружных границ (если таковые имеются). С помощью сущности *outer_boundary_curve* не более одной кривой можно идентифицировать как наружную границу;

implicit_outer: если значение логического флажка равно «true», то естественная граница поверхности используется как наружная граница.

Примечание 2 — Интерпретация указанных атрибутов представлена на рисунке 10.

Комментарии к спецификации:

WR1: если значение атрибута *implicit_outer* равно «true», то явных наружных границ нет;

WR2: наружная граница должна быть определена неявно, если базовая поверхность *basis_surface* ограничена;

WR3: перечень границ включает не более одной наружной граничной кривой;

WR4: каждая ограниченная кривая *boundary_curve* должна лежать на базовой поверхности *basis_surface*. Это подтверждается атрибутом *basis_surface* супертипа комбинированной кривой на поверхности *composite_curve_on_surface* для каждого элемента перечня границ *boundaries*.

Дополнительные комментарии:

IP1: каждая кривая множества границ *boundaries* должна быть замкнута;

IP2: никакие две кривые из множества границ *boundaries* не могут пересекаться;

IP3: не более чем одна ограниченная кривая может включать любую другую ограниченную кривую. Если наружная ограниченная кривая *outer_boundary_curve* указана, то только она может включать в себя любую другую ограниченную кривую.

6.1.16.6 Сущность boundary_curve

Сущность *boundary_curve* задает тип ограниченной кривой для определения границы поверхности.

Спецификация на языке EXPRESS:

```
*)
ENTITY boundary_curve
 SUBTYPE OF (composite_curve_on_surface);
WHERE
 WR1: SELF\composite_curve.closed_curve;
END_ENTITY;
(*
```

Комментарий к программе:

WR1: значение производного атрибута замкнутой кривой *closed_curve* супертипа *composite_curve* должно быть равно «true».

6.1.16.7 Сущность outer_boundary_curve

Сущность *outer_boundary_curve* задает подтип сущности *boundary_curve*, имеющий дополнительную семантику для определения наружной границы поверхности. Не более одной такой кривой должно быть включено в множество границ поверхности, ограниченной кривыми *curve_bounded_surface*.

Спецификация на языке EXPRESS:

```
*)
ENTITY outer_boundary_curve
 SUBTYPE OF (boundary_curve);
END_ENTITY;
(*
```

6.1.17 Определения сущностей схемы API_ABSTRACT_SCHEMA: сущности поверхностей интерфейса прикладного программирования

Настоящий подраздел описывает только сущности поверхностей, создаваемые непосредственно с помощью функций интерфейса прикладного программирования.

6.1.17.1 Сущность api_planar_surface

Сущность *api_planar_surface* задает только поверхности, создаваемые интерфейсом. Сущность *api_planar_surface* описывается сущностью *api_contour*, соответствующей внешней границе поверхности, а также перечнем контуров *api_contour*, соответствующих (возможным) внутренним границам

поверхности. Максимальное количество внутренних границ определяется реализацией интерфейса. Оно должно быть не меньше величины, установленной в разделе 9. Все указанные контуры должны лежать в одной плоскости и не должны пересекаться. Все контуры, соответствующие внутренним границам, должны принадлежать ограниченной поверхности, определенной контуром *api_contour*, соответствующим внешней границе. Ни один из указанных контуров не должен принадлежать ограниченной поверхности, определенной другим контуром *api_contour*. Это означает, что плоская поверхность *api_planar_surface* образуется дугами. Если данное условие выполняется, то плоская поверхность *api_planar_surface* вычисляется интерфейсом прикладного программирования следующим образом.

1) плоскость поверхности вычисляется по ее положению. Атрибут *position.location* задает первую точку первого сегмента комбинированной кривой *composite_curve_segment* контура *api_contour*, соответствующего внешней границе. Ось *X*, задаваемая ортогональным направлением *position.p[1]*, касается данного сегмента комбинированной кривой *composite_curve_segment*. Положительное направление обхода контура определено атрибутом *same_sense*. Ось *Z*, задаваемая ортогональным направлением *position.p[3]*, должна быть ортогональна плоскости, содержащей указанный контур *api_contour*, соответствующий внешней границе. Положительным направлением обхода контура *api_contour* является его обход против часовой стрелки по отношению к указанной ориентированной оси;

2) для каждого контура *api_contour*, определяющего плоскую поверхность *api_planar_surface*, создается экземпляр ограниченной кривой *bounded_surface_curve*. При этом каждый из них ссылается на контур *api_contour* как на кривую *curve_3d*. Атрибут *associated_geometry* указанной поверхности кривой *surface_curve* содержит только один элемент, который является плоскостью для поверхности *api_planar_surface*, описанной на этапе 1. Значение атрибута главного представления *master_representation* для поверхности кривой равно значению атрибута *curve_3d*;

3) для каждой вычисленной поверхности кривой создается экземпляр сегмента замкнутой комбинированной кривой *composite_curve_segment*. Данный сегмент:

- ссылается на поверхность кривой, которой он соответствует, как на первичную кривую *parent_curve*;
- содержит значение переменной перехода (для последнего сегмента комбинированной кривой *composite_curve_segment* контура *api_contour*, который описывается атрибутом *curve_3d* для соответствующей 3D-кривой на поверхности кривой) как значение атрибута *transition*;
- содержит атрибут *same_sense*, значение которого равно «true» для *composite_curve_segment*, соответствующего внешней границе. Данное значение гарантирует, что все прочие сегменты замкнутой кривой *composite_curve_segment* ориентированы по часовой стрелке относительно оси *Z* плоскости для поверхности *api_planar_surface* (то есть относительно ортогонального направления *position.p[3]*, см. этап 1);

4) создается экземпляр наружной ограниченной кривой *outer_boundary_curve*, сегменты которой содержат только один элемент *composite_curve_segment*, первичная кривая *parent_curve* которого ссылается (как на атрибут *curve_3d*) на контур *api_contour*, соответствующий внешней границе плоской поверхности *api_planar_surface*, создаваемой интерфейсом прикладного программирования;

5) для всех прочих сегментов замкнутой кривой создается экземпляр ограниченной кривой, сегменты которого содержат только указанные сегменты;

6) окончательно создается экземпляр поверхности *api_planar_surface*. Его базовой поверхностью *basis_surface* является поверхность *api_planar_surface*. Ее границами могут быть ограниченные кривые и наружная ограниченная кривая, вычисленная на этапах 4 и 5. Значение атрибута *implicit_outer* равно «false».

Спецификация на языке EXPRESS:

```
*)
ENTITY api_planar_surface
 SUBTYPE OF (curve_bounded_surface);
WHERE
api_WR1: 'API_ABSTRACT_SCHEMA.PLANE' IN TYPEOF (SELF.basis_surface);
api_WR2: SIZEOF(QUERY( temp <* SELF.boundaries
            'API_ABSTRACT_SCHEMA.OUTER_BOUNDARY_CURVE' IN
            TYPEOF (temp) )) =1;
api_WR3: QUERY (temp <* SELF.boundaries | SIZEOF(temp.segments) <> 1) = [];
api_WR4: SELF.implicit_outer = false;
END_ENTITY;
(*
```

Примечание 1 — Настоящая сущность интерфейса прикладного программирования задает диапазон значений некоторых функций интерфейса.

Примечание 2 — Настоящая сущность может быть применена как супертип *curve_bounded_surface*.

Определения атрибутов.

SELF\curve_bounded_surface.basis_surface: поверхность *api_planar_surface*;

SELF\curve_bounded_surface.boundaries: ограниченные кривые, соответствующие различным контурам *api_contour*, ограничивающим поверхности *api_planar_surface*, создаваемые интерфейсом прикладного программирования;

SELF\curve_bounded_surface.implicit_outer: значение «false», для которого наружная граница явно определена на множестве границ *SELF\curve_bounded_surface.boundaries*.

Комментарии к спецификации:

api_WR1: *api_planar_surface* должна лежать на плоскости;

api_WR2: существует только одна наружная ограниченная кривая на множестве границ *SELF. boundaries*;

api_WR3: каждая граница ссылается только на один элемент *composite_curve_segment*.

Дополнительный комментарий:

api_IP1: если существует несколько контуров *api_contour* в спецификации поверхности *api_planar_surface*, то расстояние между двумя указанными контурами *api_contour* должно быть не меньше допуска EPS.

6.1.18 Определения сущностей API_ABSTRACT_SCHEMA: сущности геометрических тел

Настоящий подраздел описывает групповой ресурс сущностей для булевых результатов *boolean_ result* и трехмерных моделей, являющихся частью схемы *api_abstract_schema* и определенных стандартом ИСО 10303-42.

6.1.18.1 Сущность solid_model

Сущность *solid_model* задает полное представление номинальной формы продукта. При этом соединяются все точки внутри тела. Все точки подразделяются на находящиеся внутри тела, вне тела или на его границе.

Существует несколько различных типов представления твердотельной модели.

Спецификация на языке EXPRESS:

```
*)
ENTITY solid_model
 SUPERTYPE OF (ONEOF( csg_solid, swept_area_solid))
 SUBTYPE OF (geometric_representation_item);
END_ENTITY;
(*
```

Примечание — В контексте схемы *api_abstract_schema* могут существовать только сущности *csg_solid* и *swep_area_solid*. Таким образом, супертипы оказываются модифицированными.

6.1.18.2 Сущность csg_solid

Сущность *csg_solid* — тело, представленое моделью конструктивной блочной геометрии (CSG), составлено из простейших тел, объединенных с помощью регуляризованных булевых операций. Допустимыми булевыми операциями являются пересечение, объединение и вычитание. В частном случае CSG-тело может состоять из одной простейшей CSG-сущности.

Регуляризованным подмножеством некоторого пространства является его внутреннее замыкание (настоящую фразу следует интерпретировать в обычном топологическом смысле). Для булевых результатов *boolean_result* регуляризация позволяет удалить зависшие кромки и прочие аномалии рассматриваемых операций.

Полное определение CSG-тела требует наличия геометрической и структурной информации.

Геометрическая информация задается твердотельной моделью. Как правило, это простейшие тела, например цилиндры, клинья, и экструдированные тела. Твердотельная модель может быть также копией тела solid_replica (преобразованным телом) или телом в полупространстве *half_space_solid*.

Структурная информация задается деревом (ациклическим направленным графом) *boolean_ result* и телом конструктивной блочной геометрии *csg_solid*, указывающим последовательность построения. Конечные узлы графов являются геометрическими сущностями и прочими телами. Каждое тело

конструктивной блочной геометрии *csg_solid* имеет только один ассоциированный с ним булев результат, являющийся корневым деревом (для рассматриваемого дерева в последующем могут быть получены другие булевы результаты или операнды). Значение сущности *csg_solid* заключается в том, что тело, определенное с помощью ассоциированного дерева, идентифицируется как самостоятельный объект и таким образом устанавливается его отличие от прочих булевых результатов, представляющих промежуточные результаты процесса геометрического построения.

Примечание — В контексте схемы *api_abstract_schema* функции интерфейса позволяют построить тело конструктивной блочной геометрии *csg_solid* с помощью булевых результатов. Сущность *csg_solid* соединяется дугами, однако дуги не могут использоваться для получения булева результата.

Спецификация на языке EXPRESS:

```
*)
ENTITY csg_solid
 SUBTYPE OF (solid_model);
 tree_root_expression : csg_select;
END_ENTITY;
(*
```

Определение атрибута:

tree_root_expression: булево выражение для сущностей и регуляризованных операторов, описывающих тело. Корень дерева булевых выражений представлен здесь как сущность *boolean_result* или сущность *csg_primitive*.

6.1.18.3 Сущность boolean_result

Сущность *boolean_result* определяет результат регуляризованной операции создания нового тела из двух заданных тел. Такими операциями являются регуляризованное объединение, регуляризованное пересечение и регуляризованное вычитание. При выполнении булевых операций тело рассматривается как регуляризованное множество точек.

Окончательный булев результат зависит от операции и двух операндов. При вычитании важным является порядок следования операндов. Оператор может быть либо объединением, либо пересечением, либо вычитанием.

Объединением двух тел является новое тело, содержащее все точки, принадлежащие первому операнду *first_operand*, второму операнду *second_operand* или сразу обоим операндам.

Пересечением двух тел является новое тело, которое представляет собой регуляризованное множество всех точек, принадлежащих одновременно первому и второму операндам.

Разностью двух тел является регуляризованное множество всех точек, содержащихся в первом операнде, но отсутствующих во втором операнде.

Пример — Если, например, первый операнд является блоком, а второй операнд — цилиндром подходящего размера и расположения, то булев результат операции вычитания — блок с круговым отверстием.

Спецификация на языке EXPRESS:

```
*)
ENTITY boolean_result
 SUBTYPE OF (geometric_representation_item);
 operator    : boolean_operator;
 first_operand : boolean_operand;
 second_operand : boolean_operand;
END_ENTITY;
(*
```

Определения атрибутов:

operator: булев оператор, используемый в операции для получения результата;

first_operand: первый операнд, используемый для выполнения булевой операции;

second_operand: второй операнд, используемый для выполнения булевой операции.

6.1.18.4 Сущность csg_primitive

Настоящий подраздел содержит определения всех простейших тел (сущностей) конструктивной блочной геометрии. Такими сущностями являются сфера, прямой круговой конус, прямой круговой цилиндр, тор, блок и прямой клин. Определения типов представлены ниже.

6.1.18.4.1 Сущность sphere

Сущность *sphere* задает сущность конструктивной блочной геометрии сферической формы, определенную положением центра сферы и радиусом сферы.

Спецификация на языке EXPRESS:

```
*)
ENTITY sphere
 SUBTYPE OF (geometric_representation_item);
 radius : positive_length_measure;
 centre : point;
END_ENTITY;
(*
```

Определения атрибутов:

radius: радиус сферы;

centre: центр сферы.

6.1.18.4.2 Сущность right_circular_cone

Сущность *right_circular_cone* задает сущность конструктивной блочной геометрии в форме прямого кругового конуса. Конус может быть усеченным. Он определяется осью, положением точки на оси, углом полураствора конуса и расстоянием (измеренным вдоль оси в отрицательном направлении) из точки на оси до основания конуса. Кроме того, задается радиус (ненулевой), определяющий размер и расположение верхнего основания усеченного конуса.

Спецификация на языке EXPRESS:

```
*)
ENTITY right_circular_cone
 SUBTYPE OF (geometric_representation_item);
 position  : axis1_placement;
 height    : positive_length_measure;
 radius    : length_measure;
 semi_angle : plane_angle_measure;
WHERE
 WR1: radius >= 0.0;
END_ENTITY;
(*
```

Определения атрибутов:

position: положение точки на оси конуса, направление оси;

position.location: точка на оси конуса, точка на одном из плоских круговых оснований или, если радиус равен нулю, точка в вершине конуса;

position.axis: направление центральной оси симметрии конуса;

height: расстояние между плоскими круговыми основаниями конуса (если их радиус больше нуля); расстояние от основания до вершины, если радиус равен нулю,

radius: радиус конуса в точке на оси конуса (*position.location*). Если радиус равен нулю, то конус имеет вершину в данной точке. Если радиус больше нуля, то конус является усеченным;

semi_angle: угол полураствора конуса. Это угол между осью и образующей конической поверхностью.

Комментарий к спецификации:

WR1: радиус не может быть отрицательным.

Дополнительный комментарий:

IP1: угол полураствора конуса *semi_angle*, лежит в интервале между 0° и 90°.

6.1.18.4.3 Сущность right_circular_cylinder

Сущность *right_circular_cylinder* задает простейшее тело конструктивной блочной геометрии в форме цилиндра конечной высоты. Оно определяется точкой в центре одного плоского кругового основания, направлением оси, высотой и радиусом. Основания перпендикулярны оси. Основание представляет собой круг заданного радиуса. Высотой называется расстояние от центра первой круговой грани до центра второй круговой грани, измеренное вдоль оси в положительном направлении.

Спецификация на языке EXPRESS:

```
*)
ENTITY right_circular_cylinder
 SUBTYPE OF (geometric_representation_item);
 position : axis1_placement;
 height   : positive_length_measure;
 radius   : positive_length_measure;
END_ENTITY;
(*
```

Определения атрибутов:

position: положение точки на оси цилиндра, направление оси;

position.location: точка на оси цилиндра, находящаяся в центре одной плоской круговой грани;

position.axis: направление центральной оси симметрии цилиндра;

height: расстояние между плоскими круговыми гранями цилиндра;

radius: радиус цилиндра.

6.1.18.4.4 Сущность torus

Сущность *torus* в конструктивной блочной геометрии задает тело, построенное путем движения одной окружности (образующей) вдоль другой окружности (направляющей) большего радиуса. Направляющая задается расположением и направлением (см. сущность *axis1_placement*).

Спецификация на языке EXPRESS:

```
*)
ENTITY torus
 SUBTYPE OF (geometric_representation_item);
 position    : axis1_placement;
 major_radius : positive_length_measure;
 minor_radius : positive_length_measure;
WHERE
 WR1: major_radius > minor_radius;
END_ENTITY;
(*
```

Определения атрибутов:

position: расположение центральной точки на оси, направление оси. Задает положение центра и ориентацию плоскости направляющей;

major_radius: большой радиус направляющей;

minor_radius: малый радиус направляющей.

Комментарий к программе:

WR1: *major_radius* должен быть больше *minor_radius*.

6.1.18.4.5 Сущность block

Сущность *block* задает прямоугольный параллелепипед, определенный расположением и локальной координатной системой. Блок задается положительными отрезками *x*, *y* и *z*, отложенными вдоль осей локальной координатной системы. Блок имеет вершину в начале локальной координатной системы.

Спецификация на языке EXPRESS:

```
*)
ENTITY block
 SUBTYPE OF (geometric_representation_item);
 position : axis2_placement_3d;
 x    : positive_length_measure;
 y    : positive_length_measure;
 z    : positive_length_measure;
END_ENTITY;
(*
```

Определения атрибутов:

position: расположение и ориентация системы осей сущности. Блок имеет вершину в заданном положении *position.location*, его кромки расположены вдоль локальных осей и имеют положительное направление;

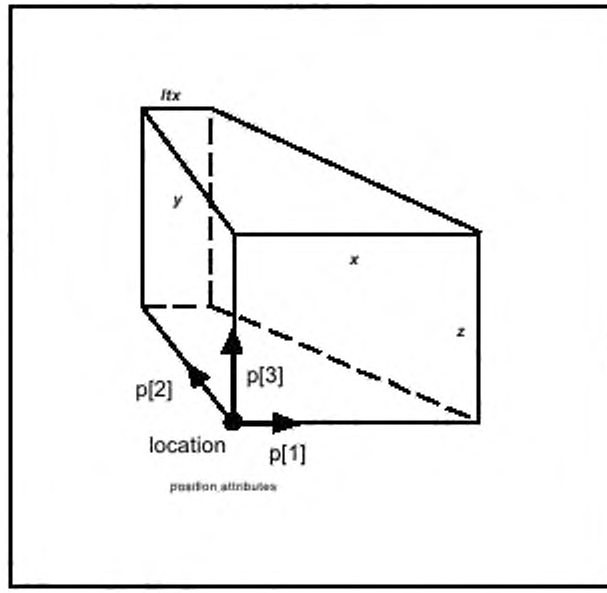
x: размер блока вдоль локальной оси *X* (position.p[1]);
y: размер блока вдоль локальной оси *Y* (position.p[2]);
z: размер блока вдоль локальной оси *Z* (position.p[3]).

6.1.18.4.6 Сущность right_angular_wedge

Сущность *right_angular_wedge* получается в результате пересечения блока с плоскостью, перпендикулярной одной из его граней. Прямой клин определяется своим расположением и локальной координатной системой. Его треугольная/трапецеидальная грань лежит в плоскости, определенной локальными осями *X* и *Y*. Данная грань определена положительными отрезками *X* и *Y* (отложенными вдоль локальных осей *X* и *Y*), отрезком *LTX* (не равным нулю, параллельным оси *X* и находящимся на расстоянии *Y* от начала локальной координатной системы) и отрезком, соединяющим концы сегментов *X* и *LTX*. Оставшаяся часть клина задается положительным отрезком *Z*, отложенным вдоль локальной оси *Z*, определяющим расстояние, на которое производится экструдирование трапеции или треугольника. Если *LTX* равно 0, то клин имеет пять граней. В противном случае клин имеет шесть граней (см. рисунок 11).

Спецификация на языке EXPRESS:

```
*)
ENTITY right_angular_wedge
 SUBTYPE OF (geometric_representation_item);
 position : axis2_placement_3d;
 x    : positive_length_measure;
 y    : positive_length_measure;
 z    : positive_length_measure;
 ltx   : length_measure;
WHERE
 WR1: ((0.0 <= ltx) AND (ltx < x));
END_ENTITY;
(*
```



Location — начало локальной координатной системы;
position attributes — атрибуты положения

Рисунок 11 — Прямой клин right_angular_wedge и его атрибуты

Определения атрибутов:

position: расположение и ориентация локальной системы осей сущности. Клин имеет одну вершину, определяемую сущностью *position.location*, ребра клина направлены вдоль осей локальной координатной системы в положительном направлении;

x: размер клина вдоль локальной оси *X*;

y: размер клина вдоль локальной оси *Y*;

z: размер клина вдоль локальной оси *Z*;

ltx: отрезок, отложенный в положительном направлении по оси *X* меньшей грани клина.

Комментарий к программе:

WR1: отрезок *ltx* должен быть неотрицательным и меньше *x*.

6.1.18.5 Сущность swept_area_solid

Сущность *swept_area_solid* формирует тело путем очерчивания плоской ограниченной поверхности по некоторой траектории. Положение в пространстве тела, полученного очерчиванием, зависит от исходного положения очерченной области *swept_area*, являющейся гранью результирующего тела *swept_area_solid*. Исключение составляет частный случай, когда очерчивание производится по замкнутой окружности на угол, равный 360°. Такое тело определяется сущностью *revolved_area_solid*.

Спецификация на языке EXPRESS:

```
*)
ENTITY swept_area_solid
 SUPERTYPE OF (ONEOF(revolved_area_solid, extruded_area_solid))
 SUBTYPE OF (solid_model);
 swept_area : curve_bounded_surface;
WHERE
 api_WR1: 'API_ABSTRACT_SCHEMA.PLANE' IN
      TYPEOF (swept_area.basis_surface);
END_ENTITY;
(*
```

Определение атрибута:

swept_area: поверхность, ограниченная кривой *curve_bounded_surface* и определяющая очерченную область. Конфигурация данной области задается атрибутом *boundaries* для ссылочной поверхности, ограниченной кривой *curve_bounded_surface*.

Комментарий к программе:

api_WR1: очерченная область *swept_area* должна быть плоской. Атрибутом базовой поверхности для поверхности, ограниченной кривой, является плоскость.

6.1.18.6 Сущность extruded_area_solid

Сущность *extruded_area_solid* задает тело, полученное путем очерчивания (вытягиванием, экструдированием) ограниченной плоской поверхности вдоль прямой. Направление поступательного движения очерченной области задается вектором направления. Глубина переноса вдоль прямой задается атрибутом *depth*. Плоская очерченная область может иметь пустоты, которые преобразуются в объемные отверстия.

Спецификация на языке EXPRESS:

```
*)
ENTITY extruded_area_solid
 SUBTYPE OF (swept_area_solid);
 extruded_direction : direction;
 depth       : positive_length_measure;
WHERE
 WR1: dot_product(
    (SELF\swept_area_solid.swept_area.basis_surface\
    elementary_surface.position.p[3]), extruded_direction) <> 0.0;

END_ENTITY;
(*
```

Определения атрибутов:

SELF\swept_area_solid.swept_area: ограниченная поверхность, очерчивающая экструдированное тело;

extruded_direction: направление очерчивания;

depth: глубина очерчивания.

Комментарий к программе:

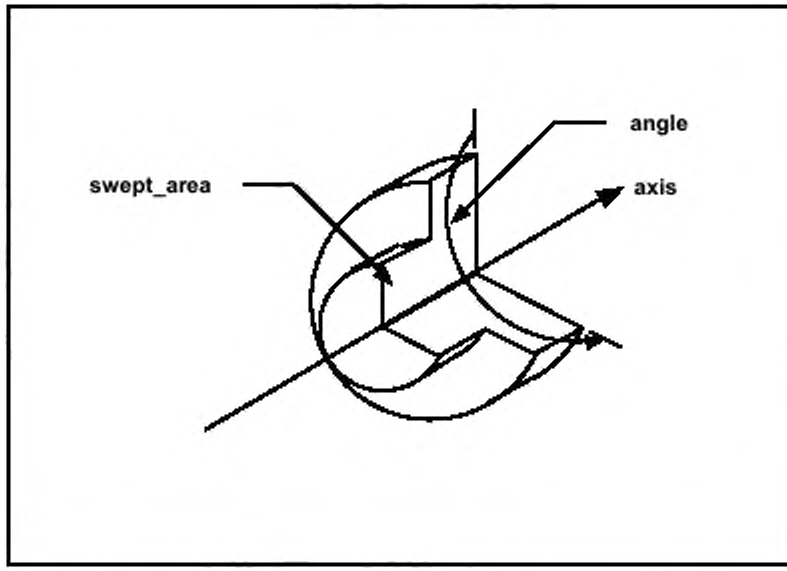
WR1: направление экструдирования не может быть перпендикулярно очерченной плоскости.

6.1.18.7 Сущность revolved_area_solid

Сущность *revolved_area_solid* задает тело путем очерчивания (вращением) плоской ограниченной поверхности вокруг оси (рисунок 12). Данная ось лежит в плоскости очерченной поверхности. Ось не должна пересекать внутренность данной поверхности. Ограниченная поверхность может иметь пустоты, которые преобразуются в объемные отверстия. Вращение производится по часовой стрелке, если смотреть на него вдоль оси в положительном направлении. В частности, если точка *A* лежит на оси, *d* — направление оси и *C* — дуга на результирующей поверхности [полученная очерчиванием произвольной точки *p*, лежащей на границе очерченной области (грани)], то данная дуга *C* является следом точки *p*, остающимся в направлении вектора *d* (*p* — *A*) в результате вращения области.

Спецификация на языке EXPRESS:

```
*)
ENTITY revolved_area_solid
 SUBTYPE OF (swept_area_solid);
 axis   : axis1_placement;
 angle   : plane_angle_measure;
DERIVE
 axis_line : line := line(axis.location, axis.z);
END_ENTITY;
(*
```



Angle — угол поворота; swept_area — очерченная область; axis — ось

Рисунок 12 — Тело, полученное вращением области

Определения атрибутов:

SELF\swept_area_solid.swept_area: поверхность, ограниченная кривой *curve_bounded_surface*, очерчивающая тело в результате вращения;

axis: ось вращения;

angle: угол поворота. Измеряется от исходного положения очерчивающей плоскости;

axis_line: линия оси вращения.

Дополнительные комментарии:

IP1: линия оси вращения *axis_line* должна лежать в очерченной плоскости атрибута *swept_face* супертипа *swept_face_solid*;

IP2: линия оси вращения не должна пересекать внутренность очерченной области;

IP3: угол поворота должен лежать между 0° и 360°.

6.1.18.8 Сущность half_space_solid

Сущность *half_space_solid* задает тело в полупространстве, которое является регулярным подмножеством области, лежащей с одной стороны неограниченной поверхности. Часть поверхности, находящаяся в указанном полупространстве, определяется перпендикуляром к поверхности и значением флажка перпендикуляра (соглашением о выборе требуемой половины пространства по направлению перпендикуляра). Если значение флажка «true», то выбирается полупространство с перпендикуляром, направленным наружу. Если значение флажка «false», то выбирается полупространство с перпендикуляром, направленным внутрь.

Для корректно заданного тела в полупространстве *half_space_solid* указанная поверхность делит пространство на два полупространства. Поверхность разделения может быть многосвязной. В этом случае перпендикуляры всех частей поверхности разделения должны быть направлены в одну сторону.

Примечание 1 — Тело в полупространстве *half_space_solid* не является подтипом сущности твердотельной модели *solid_model*. Сущность *half_space_solid* используется только как операнд булева выражения *boolean_expression*.

Примечание 2 — В контексте схемы *api_abstract_schema* допустимы только тела в полупространстве с плоскостью в качестве *базовой поверхности*.

Спецификация на языке EXPRESS:

```
*)
ENTITY half_space_solid
 SUBTYPE OF(geometric_representation_item);
 base_surface  : surface;
 agreement_flag : BOOLEAN;
WHERE
 api_WR1: 'API_ABSTRACT_SCHEMA.PLANE' IN TYPEOF (base_surface);
END_ENTITY;
(*
```

Определения атрибутов:

base_surface: поверхность, разделяющая пространство на два полупространства;

agreement_flag: значение флажка перпендикуляра равно «true», если перпендикуляр к базовой поверхности направлен от рассматриваемого тела в полупространстве.

Комментарий к спецификации:

api_WR1: поверхность, определяющая тело в полупространстве, должна быть плоскостью.

6.1.19 Определения сущностей схемы API_ABSTRACT_SCHEMA: сущности структурирования интерфейса прикладного программирования

Настоящий подраздел описывает ресурсы структурирования элементов геометрических представлений с помощью функций интерфейса прикладного программирования.

Спецификации и поведение структурированных сущностей, а также структурирование геометрических данных для получения указанных множеств описаны в разделе 5.4 настоящего стандарта.

6.1.19.1 Сущность api_group

Сущность *api_group* формирует группу, создаваемую в TDB.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_group
 SUBTYPE OF(group);
WHERE
 api_WR1: ( (SELF\group.name = 'TDB') AND (USEDIN(SELF,'') = []) )
      OR
       ( (LENGTH(SELF\group.name) = 0 ) AND
```

```
        (SIZEOF(USEDIN(SELF,'API_ABSTRACT_SCHEMA'+
                  'API_GROUP_ASSIGNMENT.ITEMS')
           ) = 1)
      );
 api_WR2: tree_api_group_structure(SELF);
END_ENTITY;
(*
```

Комментарии к спецификации:

api_WR1: сущности *api_group* присвоено имя «TDB», и ссылки других сущностей на него нет, либо именем сущности является пустая строка, и она принадлежит только одной группе *api_group*;

api_WR2: группы *api_groups* структурированы в дерево.

6.1.19.2 Сущность api_group_assignment

Сущность *api_group_assignment* назначает точки, кривые, поверхности, векторы, направления, локальные координатные системы, заполненные области, тела и группы для некоторой группы *api_group*.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_group_assignment
 SUBTYPE OF(group_assignment);
 items : SET [0:?] OF api_grouped_item;
WHERE
 api_WR1: 'API_ABSTRACT_SCHEMA.API_GROUP' IN
             TYPEOF (SELF\ group_assignment.assigned_group);
END_ENTITY;
(*
```

Определение атрибута:

item: элементы *api_grouped_item*, назначенные для некоторой группы *api_group*.

Комментарий к спецификации:

api_WR1: сущность *api_group_assignment* производит назначения для группы *api_group* интерфейса прикладного программирования.

6.1.19.3 Сущность api_set

Сущность *api_set* создает группу для базы данных CAD.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_set
 SUBTYPE OF (group);
WHERE
 api_WR1: ( (SELF\group.name = 'VIEW') AND (USEDIN(SELF,'') = []) )
      OR
      ( (SELF\group.name <> 'VIEW') AND
       (SIZEOF(USEDIN(SELF,'API_ABSTRACT_SCHEMA'+
                  'API_GROUP_ASSIGNMENT.ITEMS')
           ) = 1)
      );
 api_WR2: tree_api_set_structure (SELF);
END_ENTITY;
(*
```

Комментарии к спецификации:

api_WR1: либо сущности *api_set* присвоено имя «VIEW» и никакие другие сущности на него не ссылаются, либо ее имя отлично от «VIEW» и она принадлежит только одной сущности *api_set*;

api_WR2: сущности *api_set* структурированы в дерево.

6.1.19.4 Сущность api_set_assignment

Сущность *api_set_assignment* назначает направления, векторы, локальные координатные системы, точки, кривые, поверхности, заполненные области, тела в полупространстве, результаты булевых операций, тела и множества интерфейса прикладного программирования *api_set* для сущности *api_set*.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_set_assignment
 SUBTYPE OF(group_assignment);
 items : SET [0:?] OF api_set_item;
WHERE
 api_WR1: 'API_ABSTRACT_SCHEMA.API_SET' IN
      TYPEOF (SELF\group_assignment.assigned_group);
END_ENTITY;
(*
```

Определение атрибута:

item: элементы множества интерфейса прикладного программирования *api_set_item*, назначенные для сущности *api_set*.

Комментарии к спецификации:

api_WR1: назначения *api_set_assignment* используются только сущностью *api_set*.

6.2 Визуализация элементов геометрического представления

Все элементы представления *representation_items*, явно созданные с помощью интерфейса (как в TDB, так и в базе данных CAD), задают стиль воспроизведения. Математическим сущностям назначается нулевой стиль *null_style*. Воспроизведение элементов представления зависит от реализации. Если создание одного элемента представления требует неявного создания другого элемента представления (например, если кривая неявно создана как базовая кривая *basis_curve* для отрезка кривой *trimmed_curve*), то указанная сущность также ассоциируется с *null_style*.

Назначение стиля производится интерфейсом в процессе создания каждого *representation_item*. Назначение стиля производится путем создания экземпляра стилизованного элемента *styled_item* (для каждого *representation_item*), ссылающегося на данный *representation_item* вместе с сущностью назначения стиля *presentation_style_assignment*. Сущность *presentation_style_assignment* задает набор различных стилей воспроизведения (например, стиль точки, стиль кривой, стиль текста). Стилизация нестилизованных *representation_items* позволяет получить новые *representation_items* с заданным стилем воспроизведения. *Presentation_style_assignment* стилизованных элементов *styled_items* оказывает влияние на изображение ссылочных *representation_items* так же, как и на изображения всех *representation_items*, на которые прямо или косвенно производится ссылка указанным элементом. Изменениям подвержены только нестилизованные *representation_items*. Это означает, что стилизация ранее стилизованного *representation_item* не имеет смысла. Стилизация частично стилизованного *representation_item* изменяет только изображения его нестилизованных частей. Стилизация произвольного *representation_item* изменяет его изображение целиком. Могут быть представлены только стилизованные *representation_items*. Фактическое представление элементов зависит от удаления невидимых линий (см. пример визуализации трубопровода в среде CAD в 5.3.5).

В контексте схемы *api_abstract_schema* стилизованный элемент *styled_item* должен ссылаться только на одно назначение стиля представления *presentation_style_assignment*. В среде CAD сущность назначения стиля представления *presentation_style_assignment* должна содержать (для всех сущностей, кроме сущности заполнения области комментариев *annotation_fill_area*) только один стиль *presentation_style*. Таким стилем является текущее значение сущности *presentation_style*, соответствующее созданному *representation_item* в таблице статуса интерфейса.

Presentation_style_assignment для заполненной области комментариев *annotation_fill_area* всегда содержит одну сущность *presentation_style_select*, указывающую на необходимость заполнения определенной области цветом фона. Настоящий стиль определен сущностью *api_externally_defined_fill_area_style* интерфейса прикладного программирования. *Presentation_style_assignment* заполненной области комментариев *annotation_fill_area* может содержать любое количество стилей *fill_area_style*, причем каждый из них делает ссылку на сущность стиля штриховки *fill_area_style_hatching*.

Если текущий вид является двумерным и процесс удаления невидимых линий активизирован, то сущность *presentation_style_assignment* (соответствующая точкам кривых и заполненным областям *fill_area*) может также содержать два других стиля в интерфейсе прикладного программирования. Сущность *api_pre_defined_occlusion_style* используется для удаления невидимых линий вместе с его виртуальной высотой в виртуальном 3D-пространстве. Сущность *api_pre_defined_virtual_sent_style* описывает средства, используемые TDB для удаления невидимых линий.

Именем сущности, возвращаемой функцией интерфейса (создающей элемент представления), является имя стилизованного элемента *styled_item*, который ссылается на указанный элемент представления. Если сущность остается в TDB, то стиль воспроизведения может быть изменен впоследствии с помощью функции внесения изменений *Chg_...* (см. приложение А, раздел А.10.3).

Поставщик детали может осуществлять только логическое управление визуализацией *representation_item*, созданного с помощью функции интерфейса. Поэтому все стили, кроме стилей штриховки заполненных областей *fill_area_style_hatching*, определяются сущностью внешне определенных стилей *externally_defined_style*. Указанные *externally_defined_style* определены в настоящем стандарте, а также в других частях рассматриваемой серии стандартов, описывающих протоколы обмена видами. Часть стандарта, содержащая описания внешне определенного стиля *externally_defined_style*, определяет внешний источник для данного внешне определенного элемента *externally_defined_item*. Если некоторая реализация интерфейса использует протокол обмена видами (на который производится ссылка из программы поставщика детали для описания внешне определенного стиля *externally_defined_style*, используемого для некоторого элемента представления *representation_item*), то первый стиль, определенный для настоящего элемента представления *representation_item* в настоящем стандарте, используется вместо неизвестного стиля. Сообщение об ошибке при этом не формируется.

6.2.1 Определения типов схемы API_ABSTRACT_SCHEMA: визуальное представление

Настоящий подраздел описывает ресурс группового типа, определенный стандартом ИСО 10303-46 и являющийся частью схемы *api_abstract_schema*.

6.2.1.1 Сущность presentation_style_select

Сущность *presentation_style_select* используется сущностью *presentation_style_assignment*, чтобы ассоциировать стиль с элементом представления *representation_item*. Для каждого вида стилизуемых элементов представления предусмотрен свой стиль.

Спецификация на языке EXPRESS:

```
*)
TYPE presentation_style_select = SELECT
 (pre_defined_presentation_style,
  point_style,
  curve_style,
  surface_style_usage,
  symbol_style,
  fill_area_style,
  text_style,
  approximation_tolerance,
  externally_defined_style,
  null_style);
END_TYPE;
(*
```

6.2.1.2 Сущность null_style

Сущность *null_style* указывает, что рассматриваемому элементу никакой стиль не назначен (нулевой стиль). Стили элемента представления описаны внутри его определения. Если определение элемента не содержит каких-либо описаний стиля, данный элемент не может иметь представления.

Примечание 1 — В контексте схемы *api_abstract_schema* сущность *null_style* назначается для всех элементов представления, неявно созданных с помощью интерфейса прикладного программирования, чтобы обеспечить представление явно созданных сущностей.

Пример 1 — При создании отрезка кривой (например, дуги окружности api_circular_arc) интерфейсом прикладного программирования ее базовая кривая (например, окружность) является неявно созданной.

Спецификация на языке EXPRESS:

```
*)
TYPE null_style = ENUMERATION OF
 (null);
END_TYPE;
(*
```

Определение элементов перечислимого типа:

null: элемент представления, стиль которого представляется с помощью стилей, содержащихся в его определении (если таковые существуют).

Примечание 2 — В контексте схемы *api_abstract_schema* сущность *null_style* назначается для всех *representation_items*, неявно созданных с помощью интерфейса и позволяющих представить явно созданную сущность.

Пример 2 — При создании отрезка кривой (например, дуги окружности api_circular_arc) интерфейсом прикладного программирования ее базовая кривая (например, окружность) является неявно созданной.

6.2.1.3 Сущность size_select

Сущность *size_select* используется для описания размера маркеров и толщины кривых.

Спецификация на языке EXPRESS:

```
*)
TYPE size_select = SELECT
 (positive_length_measure,
  measure_with_unit,
  descriptive_measure,
  pre_defined_size);
END_TYPE;
(*
```

6.2.1.4 Сущность curve_font_or_scaled_curve_font_select

Сущность *curve_font_or_scaled_curve_font_select* использует сущность *curve_style_font_selector*, определяющую выбор типа линии для описания кривой и его масштабирования *curve_style_font_and_scaling*.

Спецификация на языке EXPRESS:

```
*)
TYPE curve_font_or_scaled_curve_font_select = SELECT
 (curve_style_font_select,
  curve_style_font_and_scaling);
END_TYPE;
(*
```

6.2.1.5 Сущность curve_style_font_select

Сущность *curve_style_font_select* использует сущность *curve_style_font* для предварительного определения типа линии *pre_defined_curve_font* или сущность *externally_defined_curve_font* для внешнего определения типа линии. Используется при описании немасштабируемых типов представления линий.

Спецификация на языке EXPRESS:

```
*)
TYPE curve_style_font_select = SELECT
 (curve_style_font,
  pre_defined_curve_font,
  externally_defined_curve_font);
END_TYPE;
(*
```

6.2.1.6 Сущность fill_style_select

Сущность *fill_style_select* выбирает стиль заполнения области.

Спецификация на языке EXPRESS:

```
*)
TYPE fill_style_select = SELECT
 (fill_area_style_colour,
  pre_defined_tile_style,
  externally_defined_tile_style,
  fill_area_style_tiles,
  pre_defined_hatch_style,
  externally_defined_hatch_style,
  fill_area_style_hatching);
END_TYPE;
(*
```

6.2.2 Определения типов схемы API_ABSTRACT_SCHEMA: типы визуального представления интерфейса прикладного программирования

Настоящий подраздел объявляет типы визуального представления схемы *api_abstract_schema* в интерфейсе прикладного программирования.

6.2.2.1 Сущность virtual_height_ratio

Сущность *virtual_height_ratio* задает действительное значение, определяющее виртуальную высоту элемента геометрического представления *geometric_representation_item*, который геометрически закладывается в двумерный контекст геометрического представления *geometric_representation_context*. Сущность *virtual_height_ratio* используется для предварительного затенения и удаления невидимых линий.

Спецификация на языке EXPRESS:

```
*)
TYPE virtual_height_ratio = REAL;
END_TYPE;
(*
```

6.2.3 Определения сущностей схемы API_ABSTRACT_SCHEMA: визуальное представление

Настоящий подраздел объявляет групповой ресурс сущностей, определенных ИСО 10303-46 и являющихся частью схемы *api_abstract_schema*.

6.2.3.1 Сущность styled_item

Сущность *styled_item* задает элемент представления *representation_item* с ассоциированным стилем воспроизведения.

Спецификация на языке EXPRESS:

```
*)
ENTITY styled_item
 SUBTYPE OF (representation_item);

 styles : SET [1:?] OF presentation_style_assignment;
 item   : representation_item;
WHERE
 WR1   : (SIZEOF(SELF.styles) = 1)
           XOR
         (SIZEOF(QUERY( pres_style <* SELF.styles |
          NOT ('PRESENTATION_APPEARANCE_SCHEMA.' +
            'PRESENTATION_STYLE_BY_CONTEXT' IN
          TYPEOF(pres_style))
          )) = 0);
 api_WR2: api_legal_style_number (SELF);
END_ENTITY;
(*
```

Определения атрибутов:

styles: стили, назначенные для элемента представления;

item: элемент представления, для которого назначен стиль.

Комментарии к спецификации:

WR1: множество стилей должно поддерживать только один стиль. В противном случае все члены указанного множества должны быть сущностями *presentation_style_by_context*.

Примечание — Сущность гарантирует отсутствие конфликта стилей. Описание более одного стиля возможно, если задан контекст для применения каждого стиля;

api_WR2: функция *api_legal_style_number* проверяет количество стилей, косвенно назначенных для элемента представления.

6.2.3.2 Сущность presentation_style_assignment

Сущность *presentation_style_assignment* задает множество стилей, назначенных для элемента представления *representation_item*.

Спецификация на языке EXPRESS:

```
*)
ENTITY presentation_style_assignment;
 styles : SET [1:?] OF presentation_style_select;
WHERE
 WR1: SIZEOF(QUERY(style1 <* SELF.styles |
     NOT (SIZEOF(QUERY (style2 <* (SELF.styles - style1) |
      NOT ((TYPEOF (style1) <> TYPEOF(style2)) OR
      (SIZEOF(['PRESENTATION_APPEARANCE_SCHEMA.'+
           'SURFACE_STYLE_USAGE',
           'API_ABSTRACT_SCHEMA.'+
           'EXTERNALLY_DEFINED_STYLE'] *
                  TYPEOF(style1)) = 1)))) = 0 ))) = 0;
 WR2: SIZEOF(QUERY (style1 <* SELF.styles
     'PRESENTATION_APPEARANCE_SCHEMA.SURFACE_STYLE_USAGE' IN
                        TYPEOF(style1))) <= 2;
END_ENTITY;
(*
```

Определение атрибута:

styles: множество стилей воспроизведения, назначенных для элемента представления.

Комментарии к спецификации:

WR1: один и тот же стиль не должен появиться больше одного раза в заданном множестве стилей за исключением внешне определенного стиля *externally_defined_style* или в случае использования стиля поверхности *surface_style_usage*;

WR1: сущность *surface_style_usage* не должна использоваться чаще, чем дважды в множестве стилей.

Дополнительные комментарии:

IP1: внешне определенные стили не должны конфликтовать с прочими стилями в одной сущности *presentation_style_assignment*, включая прочие внешне определенные стили.

Примечание — Один стиль конфликтует с другим, если они различны, но относятся к одной и той же характеристике, например, к цвету или толщине линий. Например, один стиль может задать голубой цвет, а другой — зеленый, причем оба применены к одной и той же сущности;

IP2: каждый тип стиля должен быть уникальным.

Пример — Если для сущности line задан стиль кривой, то он должен проявиться. Если для сущности line заданы сразу стиль кривой и стиль точки, то могут проявиться и кривая, и соответствующие декартовы точки.

IP3: если существуют два экземпляра стиля поверхности *surface_style_usage*, то каждый должен задавать описание стиля для противоположной стороны стилизуемой поверхности.

6.2.3.3 Сущность externally_defined_style

Сущность *externally_defined_style* задает внешнюю ссылку на стиль воспроизведения.

Примечание — В контексте схемы *api_abstract_schema* внешний источник *external_source* должен быть частью ИСО 13584.

Спецификация на языке EXPRESS:

```
*)
ENTITY externally_defined_style
 SUBTYPE OF (externally_defined_item);
WHERE
api_WR1 : (SELF\externally_defined_item.source.source_id LIKE
                              'ISO_13584_31')
     OR
     (SELF\externally_defined_item.source.source_id LIKE
                          'ISO_13584'+'-1'+'##');
END_ENTITY;
(*
```

Определения атрибутов:

SELF\externally_defined_item.source: название части ИСО 13854, содержащей определение стиля;

SELF\externally_defined_item.item_id: используемый идентификатор стиля.

Комментарий к спецификации:

api_WR1: источником внешне определенного стиля *externally_defined_style* должен быть либо настоящий стандарт, либо части данного стандарта с описаниями протоколов обмена видами.

6.2.3.4 Сущность curve_style

Сущность *curve_style* описывает визуализацию кривой.

Примечание — В контексте схемы *api_abstract_schema* сущность *curve_style* используется только для задания стиля линий штриховки. Данный стиль кривой использует предварительно заданный тип описания кривой *pre_defined_curve_font*, предварительно заданный размер *pre_defined_size* и цвет *colour* (цвет кривой *curve_colour* зависит от реализации и определяется приложением).

Спецификация на языке EXPRESS:

```
*)
ENTITY curve_style;
 name      : label;
 curve_font  : curve_font_or_scaled_curve_font_select;
 curve_width : size_select;
 curve_colour : colour;
WHERE
 api_WR1 : USEDIN(SELF,'API_ABSTRACT_SCHEMA.'+
            'FILL_AREA_STYLE_HATCHING.'+
            'HATCH_LINE_APPEARANCE') <> [];
END_ENTITY;
(*
```

Определения атрибутов:

name: слово или группа слов, с помощью которых производится ссылка на стиль кривой;

curve_font: для представления типов линии используются следующие сущности: тип стиля кривой *curve_style_font*, масштабированный тип стиля кривой *curve_style_font*, предварительно определенный тип кривой *pre_defined_curve_font*, масштабированный предварительно определенный тип кривой *pre_defined_curve_font*, внешне определенный тип кривой *externally_defined_curve_font* или масштабированный внешне определенный тип кривой *externally_defined_curve_font*;

curve_width: толщина видимой части представленной кривой в единицах площади;

curve_colour: цвет видимой части кривой.

Комментарий к спецификации:

api_WR1: сущность *curve_style* используется для определения типа линий *hatch_line_appearance* штриховки заполненной области *fill_area_style_hatching*.

6.2.3.5 Сущность fill_area_style

Сущность *fill_area_style* заполняет видимые сегменты кривой, области комментариев, поверхности с мозаикой и штриховкой.

Примечание — В контексте схемы *api_abstract_schema сущность fill_area_style* используется для штриховки заполненной области.

Спецификация на языке EXPRESS:

```
*)
ENTITY fill_area_style;
 name     : label;
 fill_styles : SET [1:?] OF fill_style_select;
WHERE
 WR1   : SIZEOF(QUERY(fill_style <* SELF.fill_styles |
          'PRESENTATION_APPEARANCE_SCHEMA.'+
          'FILL_AREA_STYLE_COLOUR' IN
          TYPEOF(fill_style)
          )) <= 1;
 api_WR2: QUERY(fill_style <* SELF.fill_styles |
       NOT ( 'API_ABSTRACT_SCHEMA.FILL_AREA_STYLE_HATCHING' IN
         TYPEOF(fill_style)) ) = [];
END_ENTITY;
(*
```

Определения атрибутов:

name: слово или группа слов, используемых для ссылки на стиль заполненной области;

fill_styles: множество стилей заполненной области, используемых для представления видимых сегментов кривых, представления областей комментариев и представления поверхностей.

Комментарии к спецификации:

WR1: на множестве стилей *fill_style* существует не больше одного цвета заполнения *fill_area_style_colour*;

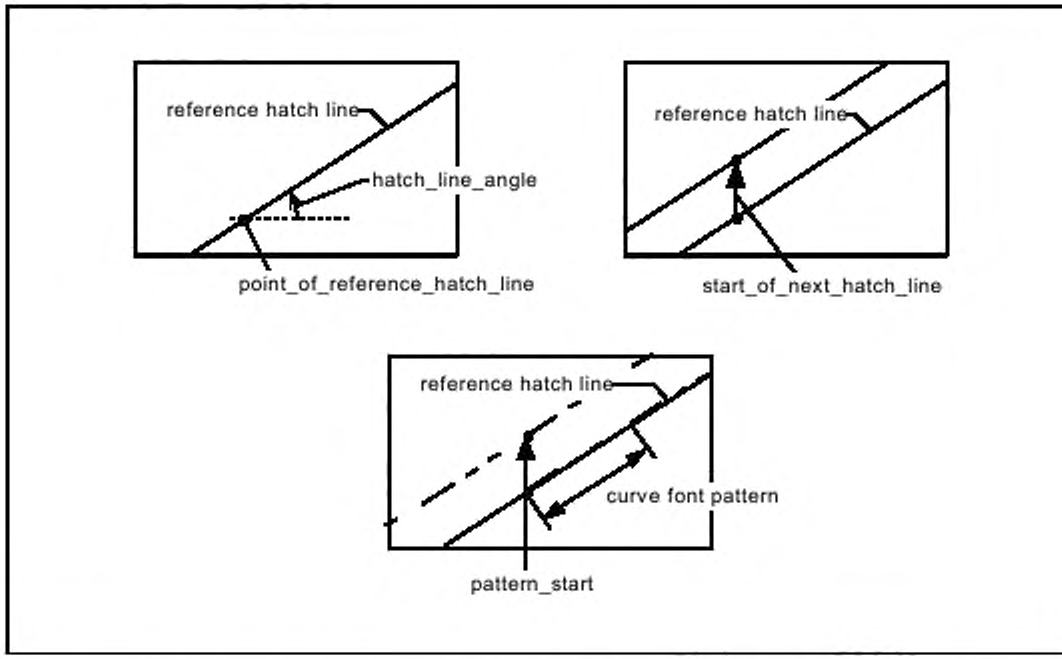
api_WR2: *fill_style_select* должен быть стилем штриховки *fill_area_style_hatching*.

6.2.3.6 Сущность fill_area_style_hatching

Сущность *fill_area_style_hatching* задает стилизованный шаблон кривой для штриховки видимого сегмента кривой, заполненной области комментариев и поверхности.

Примечание 1 — В контексте схемы *api_abstract_schema* сущность *fill_area_style_hatching* используется только для стилизации заполненной области.

Примечание 2 — В контексте схемы *api_abstract_schema* сущность *hatch_line_appearance* задается предварительно определенными элементами.



Reference hatch line — ссылочная линия штриховки; hatch_line_angle — угол штриховки, point_of_reference_hatch_line — точка ссылочной линии штриховки, start_of_next_hatch_line — начало следующей линии штриховки; curve font pattern — шаблон типа кривой, pattern_start — начало шаблона

Рисунок 13 — Стиль штриховки заполненной области

Спецификация на языке EXPRESS:

```
*)
ENTITY fill_area_style_hatching
 SUBTYPE OF (geometric_representation_item);
 hatch_line_appearance     : curve_style;
 start_of_next_hatch_line   : one_direction_repeat_factor;
 point_of_reference_hatch_line : cartesian_point;
 pattern_start          : cartesian_point;
 hatch_line_angle        : plane_angle_measure;
WHERE
 api_WR1: 'API_ABSTRACT_SCHEMA.'+
      'API_PRE_DEFINED_HATCH_CURVE_FONT' IN
              TYPEOF (SELF.HATCH_LINE_APPEARANCE.CURVE_FONT);
 api_WR2: 'API_ABSTRACT_SCHEMA.'+
      'API_PRE_DEFINED_HATCH_WIDTH' IN
              TYPEOF (SELF.HATCH_LINE_APPEARANCE.CURVE_WIDTH);
 api_WR3: 'API_ABSTRACT_SCHEMA.'+
      'API_PRE_DEFINED_HATCH_COLOUR' IN
             TYPEOF (SELF.HATCH_LINE_APPEARANCE.CURVE_COLOUR);
END_ENTITY;
(*
```

Определения атрибутов:

hatch_line_appearance: стиль линий штриховки. Любой шаблон стиля кривой должен строиться от начала каждой линии штриховки. Начало ссылочной линии штриховки задается сущностью *pattern_start*. Начало любой другой линии штриховки задается многократным использованием сущности *start_of_next_hatching_line* для начала шаблона;

start_of_next_hatching_line: вектор смещения для соседних линий штриховки;

point_of_reference_hatching_line: начальная точка отображения стиля штриховки *fill_area_style_hatching* на кривую, заполненную область комментариев или поверхность;

pattern_start: начальная точка для задания стиля кривой ссылочной линии штриховки *reference_hatching_line*;

hatch_line_angle: угол штриховки.

Примечание 3 — На рисунке 13 представлен стиль штриховки области *fill_area_style_hatching*.

Комментарии к спецификации:

api_WR1: тип линий внешней штриховки *hatch_line_appearance* предварительно определяется сущностью *api_pre_defined_hatch_curve_font* интерфейса прикладного программирования;

api_WR2: толщина линий внешней штриховки *hatch_line_appearance* предварительно определяется сущностью *api_pre_defined_hatching_curve_width* интерфейса прикладного программирования;

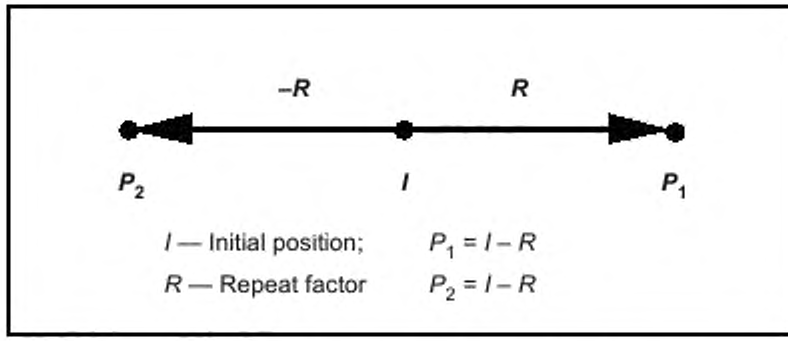
api_WR3: цвет линий внешней штриховки *hatch_line_appearance* предварительно определяется сущностью *api_pre_defined_hatching_colour* интерфейса прикладного программирования.

6.2.3.7 Сущность one_direction_repeat_factor

Сущность one_direction_repeat_factor задает однонаправленный вектор повторения, используемый сущностью *fill_area_style_hatching* для повторного задания начальной точки последующей линии штриховки относительно предыдущей линии штриховки. Если задано начальное положение *l* какой-либо линии штриховки, то однонаправленный вектор повторения *R*, заданный сущностью *one_direction_repeat_factor*, определяет два новых положения в соответствии с выражением:

$$l + k \times R, \text{ где } k = -1, 1.$$

Примечание — На рисунке 14 представлены положения вектора, определенные сущностью *one_direction_repeat_factor*.



Initial position — начальное положение; Repeat factor — фактор повторения

Рисунок 14 — Однонаправленный вектор повторения

Спецификация на языке EXPRESS:

```
*)
ENTITY one_direction_repeat_factor
 SUBTYPE OF (geometric_representation_item);
 repeat_factor : vector;
END_ENTITY;
(*
```

Определение атрибута:
repeat_factor: вектор повторения, определяющий взаимное положение линий штриховки.

6.2.3.8 Сущность colour
Сущность *colour* задает базовое свойство изображения элемента по закону отражения света.
Спецификация на языке EXPRESS:

```
*)
ENTITY colour;
END_ENTITY;
(*
```

6.2.3.9 Сущность pre_defined_size
Сущность *pre_defined_size* определяет размеры маркеров приложения.

Примечание 1 — Описание порядка использования настоящей сущности содержится в ресурсах приложения или в протоколах приложения.

Примечание 2 — В контексте схемы *api_abstract_schema* сущность *pre_defined_size* определяет толщину линий штриховки.

Спецификация на языке EXPRESS:

```
*)
ENTITY pre_defined_size
 SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

6.2.3.10 Сущность pre_defined_curve_font
Сущность *pre_defined_curve_font* определяет типы кривой *curve_fonts* для заданного приложения.

Примечание 1 — Описание порядка использования настоящей сущности см. в ресурсах приложения или протоколах приложения.

Примечание 2 — В контексте схемы *api_abstract_schema* сущность *pre_defined_curve_font* предварительно задает тип линий штриховки.

Спецификация на языке EXPRESS:

```
*)
ENTITY pre_defined_curve_font
 SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

6.2.3.11 Сущность pre_defined_colour

Сущность *pre_defined_colour* определяет цвет для заданного приложения.

Примечание 1 — Описание порядка использования настоящей сущности содержится в ресурсах приложения или протоколах приложения. Сущность *pre_defined_colour* позволяет ресурсам или протоколам приложения задавать значения цвета или значения компонентов цвета.

Примечание 2 — В контексте схемы *api_abstract_schema* сущность *pre_defined_colour* используется для предварительного определения цвета линий штриховки.

Спецификация на языке EXPRESS:

```
*)
ENTITY pre_defined_colour
 SUBTYPE OF (pre_defined_item, colour);
END_ENTITY;
(*
```

6.2.3.12 Сущность annotation_occurrence

Сущность *annotation_occurrence* задает экземпляры комментариев путем объединения двумерной геометрии (элементов комментариев) со стилем ее воспроизведения. При этом используются следующие сущности: *area_dependent_annotation_representation*, *view_dependent_annotation_representation*, *curve_style_curve_pattern*, *fill_area_style_tile_curve_with_style* и *fill_area_style_tile_coloured_region*. Дополнительную информацию об указанных сущностях см. в ИСО 10303-46.

Примечание 1 — В контексте схемы *api_abstract_schema* сущность *annotation_occurrence* используется только для воспроизведения области комментариев annotation_fill_area, для которой назначен стиль штриховки fill_area_style_hatching.

Примечание 2 — В контексте схемы *api_abstract_schema* интерфейсом наследуется только экземпляр заполненной области комментариев *annotation_fill_area_occurrence*. Таким образом, супертип оказывается отсеченным.

Спецификация на языке EXPRESS:

```
*)
ENTITY annotation_occurrence
 SUPERTYPE OF (annotation_fill_area_occurrence)
 SUBTYPE OF (styled_item);
WHERE
 WR1: 'API_ABSTRACT_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM' IN
                              TYPEOF (SELF);
END_ENTITY;
(*
```

Комментарий к спецификации:

WR1: экземпляр комментариев *annotation_occurrence* должен быть элементом геометрического представления *geometric_representation_item*.

6.2.3.13 Сущность annotation_fill_area_occurrence

Сущность *annotation_fill_area_occurrence* назначает стиль заполнения области комментариев *annotation_fill_area*, включая спецификацию точки, используемой в качестве начальной для рассматриваемой области заполнения.

Примечание 1 — В контексте схемы *api_abstract_schema annotation_fill_area_occurrence* используется для назначения стиля штриховки *fill_area_style_hatching* указанной области *annotation_fill_area*. При этом для указанного назначения исходная точка *fill_area_style_hatching.point_of_reference_hatching_line* преобразуется в целевую точку *fill_style_target*.

Примечание 2 — В контексте схемы *api_abstract_schema* (где *annotation_fill_area* и *fill_area_style_hatching* могут быть определены только в двумерном контексте геометрического представления *geometric_representation_context*) ось *X* контекста геометрического представления стиля штриховки области *fill_area_style_hatching* должна быть неявно отображена на оси *X* в контексте геометрического представления, которому принадлежит целевая точка.

Спецификация на языке EXPRESS:

```
*)
ENTITY annotation_fill_area_occurrence
 SUBTYPE OF (annotation_occurrence);
 fill_style_target : point;
WHERE
 WR1   : 'API_ABSTRACT_SCHEMA.ANNOTATION_FILL_AREA' IN
                          TYPEOF (SELF.item);
 api_WR2 : SIZEOF(QUERY(psa <*
             SELF\annotation_occurrence\styled_item.styles |
       SIZEOF(QUERY(pss <* psa.styles
        (NOT ('API_ABSTRACT_SCHEMA.FILL_AREA_STYLE' IN
                              TYPEOF(pss)))
       AND
       (SIZEOF(QUERY(fss <* pss.fill_styles
        (NOT ('API_ABSTRACT_SCHEMA.FILL_AREA_STYLE_HATCHING)
        ))) -0))) - 0) - 0)) - 0;
 api_WR3 : SIZEOF(QUERY(psa <*
             SELF\annotation_occurrence\styled_item.styles |
       SIZEOF(QUERY(pss <* psa.styles
       NOT pss.point_of_reference_hatch_line =
                    SELF.fill_style_target))- 0)) - 0;
END_ENTITY;
(*
```

Определение атрибута:

fill_style_target: точка отсчета для расположения элементов стиля области *fill_area_style*, назначенных для заполненной области комментариев.

Комментарии к спецификации:

WR1: стилизованным элементом должна быть заполненная область комментариев;

api_WR2: типом сущности *fill_style* на множестве *fill_style_select* (в процессе заполнения области комментариев) может быть только тип, определяемый сущностью *fill_area_style_hatching*;

api_WR3: fill_style_target должна ссылаться на точку ссылочной линии штриховки *point_of_reference_hatching_line* при заданном стиле штриховки *fill_area_style_hatching* на множестве *fill_style*, назначенном для элемента представления *annotation_fill_area*.

6.2.4 Определения сущностей схемы API_ABSTRACT_SCHEMA: внешне определенные стили визуального представления

Данный подраздел описывает внешне определенные стили, представленные в настоящем стандарте. На указанные стили можно ссылаться согласно прикладной программе при определении логики визуализации геометрических сущностей и комментариев.

Указанные стили определены только логически. Это облегчает настройку интерфейса его конечным пользователем.

Указанные стили определяются как внешне определенные стили *externally_defined_style*. Это позволяет расширить множество доступных стилей в соответствии с протоколами обмена видами, определенными в ИСО 13584.

Все стили, определенные в настоящей части стандарта, могут быть применены. Если какой-либо из внешне определенных стилей, установленный в некотором протоколе обмена видами, не применен в некоторой реализации интерфейса, то тогда следует использовать первый по порядку стиль, определенный для данного вида элемента в настоящем стандарте. При этом сообщение об ошибке не формируется.

6.2.4.1 Сущность api_externally_defined_point_style

Сущность *api_externally_defined_point_style* задает визуальное представление точек.

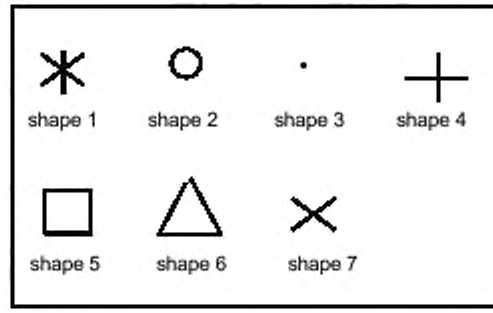
Нижеследующие стили точек определены в настоящем стандарте. Размер и цвет точек зависит от реализации. Значения стиля по умолчанию обеспечиваются интерфейсом пользователя. Для этого обеспечиваются возможности настройки интерфейса конечным пользователем.

Примечание 1 — Прочие внешне назначенные стили точек *api_externally_defined_point_style* могут быть определены протоколом обмена видами в соответствии с ИСО 13584. Если какая-либо реализация интерфейса не поддерживается указанным протоколом обмена видами, то используется первый по порядку стиль, определенный в настоящем стандарте. При этом сообщение об ошибке не формируется.

Таблица 7 — Внешне определенные стили точек

Название элемента item_id	Геометрическая форма	Цвет, размер
asterisk_point	Таблица 8, стиль 1	Зависит от реализации
circle_point	Таблица 8, стиль 2	Зависит от реализации
dot_point	Таблица 8, стиль 3	Зависит от реализации
plus_point	Таблица 8, стиль 4	Зависит от реализации
square_point	Таблица 8, стиль 5	Зависит от реализации
triangle_point	Таблица 8, стиль 6	Зависит от реализации
x_point	Таблица 8, стиль 7	Зависит от реализации
virtual_point	Не определена	Не определены

Таблица 8 — Внешне определенные стили точек



Shape – стиль

Спецификация на языке EXPRESS:

```
*)
ENTITY api_externally_defined_point_style
 SUBTYPE OF (externally_defined_style);
END_ENTITY;
(*
```

Примечание 2 — Настоящая сущность может быть применена как внешне определенный стиль *externally_defined_style*.

Определения атрибутов:

SELF\externally_defined_item.source: название части настоящего стандарта, содержащей определение стиля;

SELF_externally_defined_item.item_id: идентификатор стиля.

6.2.4.2 Сущность api_externally_defined_curve_style

Сущность *api_externally_defined_curve_style* задает визуальное представление кривой.

Нижеследующие стили кривых определены в настоящем стандарте. Размер, цвет и точный шаблон указанных стилей зависит от реализации. Значение по умолчанию задается интерфейсом пользователя. При этом должна быть обеспечена возможность настройки интерфейса конечного пользователя.

Примечание 1 — Прочие внешне определенные стили кривых *api_externally_defined_curve_style* могут быть определены протоколом обмена видами в соответствии с настоящим стандартом. Если какая-либо реализация интерфейса не поддерживает указанный протокол обмена видами, то используется первый по порядку стиль, определенный в настоящем стандарте. При этом сообщение об ошибке не формируется.

Таблица 9 — Внешне определенные стили кривых

Название элемента item_id	Описание	Вид на чертеже	Цвет, толщина линий и шаблон
plain_solid_line_thick	Сплошная толстая		Зависит от реализации
plain_solid_line_middle	Сплошная средней толщины		Зависит от реализации
plain_solid_line_thin	Сплошная тонкая		Зависит от реализации
plain_dashed_line_thick	Штриховая толстая		Зависит от реализации
plain_dashed_line_thin	Штриховая тонкая		Зависит от реализации
alternate_long_dash_dot_line_thick	Штрихпунктирная толстая		Зависит от реализации
alternate_long_dash_dot_line_thin	Штрихпунктирная тонкая		Зависит от реализации
alternate_long_dash_double_dot_line_thin	Штрихпунктирная (с двумя точками) тонкая		Зависит от реализации
alternate_long_dash_dot_line_thin_thicken_parts	Штрихпунктирная тонкая со сплошными толстыми участками в начале, в конце и при изменении направления		Зависит от реализации
virtual_line	Для вспомогательных построений	Не определен	Не определены

Спецификация на языке EXPRESS:

```
*)
ENTITY api_externally_defined_curve_style
 SUBTYPE OF (externally_defined_style);
END_ENTITY;
(*
```

Примечание 2 — Настоящая сущность может быть использована как внешне определенный стиль *externally_defined_style*.

Определения атрибутов:

SELF\externally_defined_item.source: название части настоящего стандарта, содержащей определение указанного стиля;

SELF\externally_defined_item.item_id: идентификатор используемого стиля.

6.2.4.3 Сущность api_externally_defined_fill_area_style

Сущность *api_externally_defined_fill_area_style* задает визуальное представление заполненной области. Нижеследующие стили заполнения области определены в настоящем стандарте. Цвет непрозрачного заполнения области *opaque_fill_area* зависит от реализации. Он должен соответствовать цвету фона.

Примечание 1 — Прочие внешне определенные стили заполнения области *api_externally_defined_fill_area_style* могут быть определены протоколом обмена видами в соответствии с настоящим стандартом. Если какая-либо реализация интерфейса не поддерживается настоящим протоколом обмена видами, то используется первый по порядку стиль, определенный в настоящем стандарте. При этом сообщение об ошибке не формируется.

Таблица 10 — Внешне определенные стили заполненной области

Название item_id	Описание	Цвет
opaque_fill_area	Область заполняется цветом фона. Она может загораживать (затирать) другие сущности. Область может быть заштрихована	Зависит от реализации
empty_fill_area	Область бесцветная. Она не может загораживать (затирать) другие сущности. Область может быть заштрихована	Бесцветный

Спецификация на языке EXPRESS:

```
*)
ENTITY api_externally_defined_fill_area_style
 SUBTYPE OF (externally_defined_style);
END_ENTITY;
(*
```

Примечание 2 — Настоящая сущность может быть применена как внешне определенный стиль *externally_defined_style*.

Определения атрибутов:

SELF\externally_defined_item.source: название части настоящего стандарта, содержащей определение рассматриваемого стиля;

SELF\externally_defined_item.item_id: идентификатор используемого стиля.

6.2.4.4 Сущность api_externally_defined_surface_style

Сущность *api_externally_defined_surface_style* задает визуальное представление поверхности.

Нижеследующие стили поверхностей определены в настоящем стандарте. Все визуальные представления поверхностей зависят от реализации. Значения по умолчанию должны быть обеспечены интерфейсом пользователя. Также должна быть обеспечена возможность настройки интерфейса конечным пользователем.

Примечание 1 — Прочие внешне определенные стили поверхностей *api_externally_defined_surface_style* для интерфейса прикладного программирования могут быть определены протоколом обмена видами в соответствии с настоящим стандартом. Если какая-либо реализация интерфейса не поддерживается настоящим протоколом обмена видами, то используется первый по порядку стиль, определенный в настоящем стандарте. При этом сообщение об ошибке не формируется.

Таблица 11 — Внешне определенные стили поверхности

Название item_id	Описание	Изображение
solid_surface	Представляется на экране монитора в текущем стиле, определенном приемной системой для визуализации поверхности	Зависит от реализации

Спецификация на языке EXPRESS:

```
*)
ENTITY api_externally_defined_surface_style
 SUBTYPE OF (externally_defined_style);
END_ENTITY;
(*
```

Примечание 2 — Настоящая сущность может быть применена как внешне определенный стиль *externally_defined_style*.

Определения атрибутов:

SELF\externally_defined_item.source: название части настоящего стандарта, содержащей определение рассматриваемого стиля;

SELF\externally_defined_item.item_id: идентификатор используемого стиля.

6.2.5 Определения сущностей схемы API_ABSTRACT_SCHEMA: предварительно определенные стили визуального представления

Настоящий подраздел описывает предварительно определенные стили, описанные в настоящем стандарте. На них могут производиться ссылки из прикладных программ, задающих логику визуализации геометрических сущностей и комментариев. Свойства предварительно определенных стилей заданы для линий штриховки, а также для режимов затенения двумерных областей.

6.2.5.1 Сущность api_pre_defined_hatching_width

Сущность *api_pre_defined_hatching_width* задает описание логической толщины линий штриховки.

Толщина линий предварительно определена в настоящем стандарте. Конкретная толщина линии зависит от применения. Значение по умолчанию обеспечивается интерфейсом пользователя. Необходимо обеспечить возможность настройки интерфейса конечного пользователя.

Таблица 12 — Предварительно определенная толщина линий штриховки

Название	Описание	Толщина линии
thin_hatching_line	Тонкие линии. Используются для штриховки областей проекта изделия из металла	Зависит от реализации
middle_thick_hatching_line	Линии средней толщины. Используются для особых целей	Зависит от реализации
thick_hatching_line	Толстые линии. Используются для особых целей	Зависит от реализации

Спецификация на языке EXPRESS:

```
*)
ENTITY api_pre_defined_hatch_width
 SUBTYPE OF (pre_defined_size);
WHERE
 api_WR1: SELF\pre_defined_item.name IN ['thin_hatch_line',
                    'middle_thick_hatch_line',
                    'thick_hatch_line'];
END_ENTITY;
(*
```

Примечание — Если штриховка создана в базе данных CAD, то толщина линий штриховки может быть представлена положительной мерой длины *positive_length_measure* в соответствии со значением, генерируемым реализацией интерфейса.

Определение атрибута:

SELF\pre_defined_item.name: метка используемой толщины линии штриховки.

Комментарий к спецификации:

api_WR1: имя определено в настоящем стандарте.

6.2.5.2 Сущность api_pre_defined_hatch_curve_font

Сущность *api_pre_defined_hatch_curve_font* задает описание типов линий штриховки.

Нижеследующие типы предварительно определены в настоящем стандарте.

Таблица 13 — Сегменты линий и отступы для предварительно определенных типов штриховки

Название	Сегмент (мм)	Отступ (мм)	Сегмент (мм)	Отступ (мм)	Сегмент (мм)	Отступ (мм)
continuous						
dashed	4.0	1.5	1.0	1.0		
chain	7.0	1.0	1.0	1.0	1.0	1.0
chain_double_dash	7.0	1.0				
dotted	1.0	1.0				

Спецификация на языке EXPRESS:

```
*)
ENTITY api_pre_defined_hatsh_curve_font
 SUBTYPE OF (pre_defined_hatsh_curve_font);
WHERE
 api_WR1: SELF\pre_defined_item.name IN ['continious',
                      'dashed',
                      'chain',
                      'chain_double_dash',
                      'dotted'];
END_ENTITY;
(*
```

Определение атрибута:

SELF\pre_defined_item.name: метка используемого типа линии штриховки.

Комментарий к спецификации:

api_WR1: имя определено в настоящем стандарте.

6.2.5.3 Сущность api_pre_defined_hatch_colour

Сущность *api_pre_defined_hatch_colour* задает логический цвет линии штриховки.

Цвет предварительно определен в настоящем стандарте. Конкретное значение цвета зависит от реализации. Значение по умолчанию обеспечивается интерфейсом пользователя. Должна быть обеспечена возможность настройки интерфейса конечным пользователем.

Таблица 14 — Предварительно определенный цвет штриховки

Название	Описание	Цвет
hatch_line_colour	Цвет штриховки	Зависит от реализации

Спецификация на языке EXPRESS:

```
*)
ENTITY api_pre_defined_hatch_colour
 SUBTYPE OF (pre_defined_colour);
WHERE
 api_WR1: SELF\pre_defined_item.name ='hatch_line_colour';
END_ENTITY;
(*
```

Примечание — Если штриховка создана в базе данных CAD, то цвет штриховки может быть определен значением цвета в соответствии с настройкой реализации интерфейса.

Определение атрибута:

SELF\pre_defined_item.имя: метка используемого цвета.

Комментарий к спецификации:

api_WR1: имя определено в настоящем стандарте.

6.2.5.4 Сущность api_pre_defined_occlusion_style

Сущность *api_pre_defined_occlusion_style* указывает, что стилизованный элемент *styled_item* должен быть включен в глобальный процесс удаления невидимых линий. Сущность определяет виртуальную высоту *styled_item* в виртуальном 3D-пространстве.

Примечание 1 — Если виртуальная высота не ассоциирована с текущим стилем, то он не включен в процесс удаления невидимых линий.

Примечание 2 — Если CAD обеспечивает ресурсы предварительного затенения в 2D-пространстве, то настоящий стиль должен отображаться на указанные ресурсы и процесс удаления невидимых линий реализуется CAD.

Примечание 3 — Если представление формы *shape_representation* создано в репозитории (соответствующем ИСО 10303 и требованиям прикладной программы, обеспечивающей выполнение соотношений предварительного затенения), то процесс удаления невидимых линий должен отображать настоящий стиль поверх предварительного затенения. Если прикладная программа, удовлетворяющая требованиям ИСО 10303, не обе-

спечивает выполнения соотношений предварительного затенения, то невидимые линии должны быть вычислены интерфейсом. При этом только видимые сущности (на языке EXPRESS) должны быть созданы в репозитории в соответствии с ИСО 10303 и требованиями прикладной программы.

Таблица 15 — Предварительно определенный стиль невидимых линий

Название	Описание	Цвет
hidden_line_no_changed	Невидимая линия не изменяется	Зависит от реализации
hidden_line_dashed	Невидимая линия является штриховой	Зависит от реализации
hidden_line_invisible	Линия невидима	Зависит от реализации

Спецификация на языке EXPRESS:

```
*)
ENTITY api_pre_defined_occlusion_style
 SUBTYPE OF (pre_defined_presentation_style);
 view_level: virtual_height_ratio;
WHERE
 api_WR1: SELF\pre_defined_item.name IN ['hidden_line_no_changed',
                    'hidden_line_dashed',
                    'hidden_line_invisible'];
END_ENTITY;
(*
```

Определения атрибутов:

SELF\pre_defined_item.имя: метка используемого стиля невидимых линий;

view_level. виртуальная высота стилизованного элемента *styled_item* в виртуальном 3D-пространстве.

Комментарий к спецификации:

api_WR1: предварительно определенное имя настоящего стиля выбирается из списка атрибутов: *hidden_line_no_changed*, *hidden_line_dashed* или *hidden_line_invisible*.

6.2.5.5 Сущность api_pre_defined_vitually_sent_style

Сущность *api_pre_defined_vitually_sent_style* указывает, что стилизованный элемент *styled_item* (остающийся в TDB для реализации глобального процесса удаления невидимых линий) теперь недоступен для прикладной программы. По окончании процесса удаления невидимых линий данный элемент отправляется в CAD.

Примечание 1 — Предварительно определенный стиль представления *pre_defined_presentation_style* может быть использован только для сущностей, находящихся в TDB.

Примечание 2 — Если сущность отправлена виртуально, то она должна быть удалена из структуры *api_group*. Она должна принадлежать корневой группе, которая не может использоваться какой-либо функцией управления группой.

Спецификация на языке EXPRESS:

```
*)
ENTITY api_pre_defined_virtually_sent_style
 SUBTYPE OF (pre_defined_presentation_style);
 api_set_name: STRING;
WHERE
 api_WR1: SELF\pre_defined_item.name ='virtually_sent';
END_ENTITY;
(*
```

Определение атрибута:

api_set_name: имя множества *api_set* интерфейса прикладного программирования, которое было открытым во время виртуальной отправки сущности.

Комментарий к спецификации:

api_WR1: предварительно определенное имя стиля, отправляемого виртуально.

Дополнительные комментарии:

api_IP1: данный стиль назначается только сущностям, содержащимся в TDB;

api_IP2: сущность *api_set_name* должна соответствовать имени множества *api_set* интерфейса прикладного программирования.

6.3 Определения функций схемы API_ABSTRACT_SCHEMA

6.3.1 Определения функций схемы API_ABSTRACT_SCHEMA: геометрические и топологические представления

Настоящий подраздел объявляет функции, определенные ИСО 10303-42 и являющиеся частью схемы *api_abstract_schema*.

6.3.1.1 Функция dimension_of

Функция *dimension_of* отражает целочисленное значение размерности *dimension_count* контекста геометрического представления *geometric_representation_context*, в котором геометрически закладываются входные элементы геометрического представления *geometric_representation_items*.

С учетом ограничений, налагаемых глобальным правилом совместимости размерностей *compatible_dimension*, данное значение равно размерности координатного пространства *coordinate_space_dimension* для входных элементов геометрического представления. Правило совместимости размерностей установлено ИСО 10303-42, раздел 4.5.1.

Спецификация на языке EXPRESS:

```
*)
FUNCTION dimension_of(item : geometric_representation_item) :
 dimension_count;
 LOCAL
  x : SET OF representation;
  y : representation_context;
 END_LOCAL;
 -- Find the set of representation in which the item is used.
 x := using_representations(item);
 -- Determine the dimension_count of the
 -- geometric_representation_context. Note that the
 -- RULE compatible_dimension ensures that the context_of_items
 -- is of type geometric_representation_context and has
 -- the same dimension_count for all values of x.
 y := x[1].context_of_items;
 RETURN (y\geometric_representation_context.coordinate_space_dimension);
END_FUNCTION;
(*
```

Определение аргумента:

item: входные элементы геометрического представления, для которых определена размерность пространства.

6.3.1.2 Функция associated_surface

Функция *associated_surface* определяет уникальную поверхность, ассоциированную с типом *pcurve_or_surface*. Она необходима для выполнения действий с кривыми на поверхности и их подтипами.

Спецификация на языке EXPRESS:

```
*)
FUNCTION associated_surface(arg : pcurve_or_surface) : surface;
 LOCAL
  surf : surface;
 END_LOCAL;

 IF 'GEOMETRY_SCHEMA.PCURVE' IN TYPEOF(arg) THEN
  surf := arg.basis_surface;
 ELSE
  surf := arg;
 END_IF;
 RETURN(surf);
END_FUNCTION;
(*
```

Определения аргументов:

arg: входная кривая или поверхность, для которых необходимо определение ассоциированной первичной поверхности;

surf: выходная первичная поверхность, ассоциированная с атрибутом *arg*.

6.3.1.3 Функция base_axis

Функция *base_axis* отображает три нормированных ортогональных направления: *u[1]*, *u[2]* и *u[3]*. В трехмерном пространстве при полных исходных данных ортогональное направление *u[3]* ориентировано в направлении оси *axis3*, ортогональное направление *u[1]* — в направлении проекции оси *axis1* на плоскость, перпендикулярную к *u[3]*, ортогональное направление *u[2]* перпендикулярно обоим ортогональным направлениям *u[1]* и *u[3]*, при этом данное направление совпадает с положительным направлением оси *axis2*. В двумерном пространстве: ортогональное направление *u[1]* ориентировано вдоль оси *axis1*, ортогональное направление *u[2]* перпендикулярно ортогональному направлению *u[1]*. При этом данное направление совпадает с направлением оси *axis2*. Для неполных исходных данных недостающие значения определяются по умолчанию по некоторым формулам.

Спецификация на языке EXPRESS:

```
*)
FUNCTION base_axis(dim : INTEGER; axis1, axis2, axis3 : direction) :
                        LIST [2:3] OF direction;
 LOCAL
  vec  : direction;
  u    : LIST [2:3] OF direction;
  factor : REAL;
 END_LOCAL;
 IF (dim = 3) THEN
  u[3] := NVL(axis3, direction([0.0,0.0,1.0]));
  u[1] := first_proj_axis(u[3],axis1);
  u[2] := second_proj_axis(u[3],u[1],axis2);
 ELSE
  u[3] := ?;
  IF EXISTS(axis1) THEN
   u[1] := normalise(axis1);
   u[2] := orthogonal_complement(u[1]);
   IF EXISTS(axis2) THEN
    factor := dot_product(axis2,u[2]);
    IF (factor < 0.0) THEN
     u[2].direction_ratios[1] := -u[2].direction_ratios[1];
     u[2].direction_ratios[2] := -u[2].direction_ratios[2];
    END_IF;
   END_IF;
  ELSE
   IF EXISTS(axis2) THEN
    u[2] := normalise(axis2);
    u[1] := orthogonal_complement(u[2]);
    u[1].direction_ratios[1] := -u[1].direction_ratios[1];
    u[1].direction_ratios[2] := -u[1].direction_ratios[2];
   ELSE
    u[1].direction_ratios[1] := 1.0;
    u[1].direction_ratios[2] := 0.0;
    u[2].direction_ratios[1] := 0.0;
    u[2].direction_ratios[2] := 1.0;
   END_IF;
  END_IF;
 END_IF;
 RETURN(u);
END_FUNCTION;
(*
```

Определения аргументов:

dim: (вход) целочисленное значение размерности пространства, в котором задаются стандартные ортогональные направления;

axis1: (вход) направление, используемое как первая аппроксимация направления выходной оси *u[1]*;

axis2: (вход) направление, используемое для определения положительного ортогонального направления *u[2]*;

axis3: (вход) ортогональное направление *u[3]* в том случае, если размерность dim = 3. Направление равно «null», если dim = 2;

u: (выход) перечень dim (2 или 3) взаимно перпендикулярных направлений.

6.3.1.4 Функция build_2axes

Функция *build_2axes* отображает два нормированных ортогональных направления. Ортогональное направление *u[1]* определяется сущностью *ref_direction*, ортогональное направление *u[2]* перпендикулярно *u[1]*. Если исходные данные являются неполными, то сущности *ref_direction* по умолчанию присваивается значение (1.0, 0.0).

Спецификация на языке EXPRESS:

```
*)
FUNCTION build_2axes(ref_direction : direction) : LIST [2:2] OF direction;
 LOCAL
  u : LIST[2:2] OF direction;
 END_LOCAL;
 u[1] := NVL(normalise(ref_direction), direction([1.0,0.0]));
 u[2] := orthogonal_complement(u[1]);
 RETURN(u);
END_FUNCTION;
(*
```

Определения аргументов:

ref_direction: (вход) ссылочное направление в двумерном пространстве, по умолчанию равно [1.0, 0.0];

u: (выход) перечень двух взаимно перпендикулярных направлений, орт u[1] параллелен сущности ref_direction.

6.3.1.5 Функция build_axes

Функция *build_axes* возвращает три нормированных ортогональных направления. Ортогональное направление *u[3]* имеет направление *axis*, *u[1]* направлен по проекции вектора *ref_direction* на плоскость, перпендикулярную *u[3]*. Ортогональное направление *u[2]* — векторное произведение *u[3]* и *u[1]*. Если исходные данные неполные, то берутся значения по умолчанию.

Спецификация на языке EXPRESS:

```
*)
FUNCTION build_axes(axis, ref_direction : direction) :
                    LIST [3:3] OF direction;
 LOCAL
  u : LIST[3:3] OF direction;
 END_LOCAL;

 u[3] := NVL(normalise(axis), direction([0.0,0.0,1.0]));
 u[1] := first_proj_axis(u[3],ref_direction);
 u[2] := normalise(cross_product(u[3],u[1])).orientation;
 RETURN(u);
END_FUNCTION;
(*
```

Определения аргументов:

axis: (вход) ожидаемое направление *u[3]*, по умолчанию равно [0.0, 0.0, 1.0];

ref_direction: (вход) направление *u[1]*;

u: (выход) перечень трех взаимно ортогональных направлений в 3D-пространстве.

6.3.1.6 Функция orthogonal_complement

Функция *orthogonal_complement* возвращает направление, являющееся ортогональным дополнением к исходному направлению. Исходное направление должно быть двумерным. Результатом функции является вектор того же типа, перпендикулярный исходному вектору.

Спецификация на языке EXPRESS:

```
*)
FUNCTION orthogonal_complement(vec : direction) : direction;
 LOCAL
  result : direction;
 END_LOCAL;

 IF (vec.dim <> 2) OR NOT EXISTS (vec) THEN
  RETURN(?);
 ELSE
  result.direction_ratios[1] := -vec.direction_ratios[2];
  result.direction_ratios[2] := vec.direction_ratios[1];
  RETURN(result);
 END_IF;
END_FUNCTION;
(*
```

Определения аргументов функции:

vec: (вход) направление в 2D-пространстве;

result: (выход) направление, ортогональное аргументу *vec*.

6.3.1.7 Функция first_proj_axis

Функция *first_proj_axis* представляет направление в трехмерном пространстве. Если вход определен полностью, то результатом является проекция вектора *arg* на плоскость, нормальную оси *z_axis*. Если вектор *arg* задан по умолчанию, то результатом является проекция ортогонального направления [1, 0, 0] на данную плоскость. Исключением является ситуация, когда направление *z_axis* равно [1, 0, 0] и вектор *arg* по умолчанию равен [0, 1, 0]. Сбой возникает, если направление данного вектора совпадает с исходным направлением оси *z_axis*.

Спецификация на языке EXPRESS:

```
*)
FUNCTION first_proj_axis(z_axis, arg : direction) : direction;
 LOCAL
  x_axis : direction;
  v    : direction;
  z    : direction;
  x_vec : vector;
 END_LOCAL;
 IF NOT EXISTS(z_axis) OR (NOT EXISTS(arg)) OR (arg.dim <> 3) THEN
   x_axis := ?;
 ELSE
  z_axis := normalise(z_axis);
  IF NOT EXISTS(arg) THEN
   IF (z_axis <> direction([1.0,0.0,0.0])) THEN
    v := direction([1.0,0.0,0.0]);
   ELSE
    v := direction([0.0,1.0,0.0]);
   END_IF;
  ELSE
   IF ((cross_product(arg,z).magnitude) = 0.0) THEN
    RETURN (?);
   ELSE
    v := normalise(arg);
   END_IF;
  END_IF;
  x_vec := scalar_times_vector(dot_product(v, z), z_axis);
  x_axis := vector_difference(v, x_vec).orientation;
  x_axis := normalise(x_axis);
 END_IF;
 RETURN(x_axis);
END_FUNCTION;
(*
```

Определения аргументов:

z_axis: (вход) направление, определяющее локальную ось *Z*;

arg: (вход) направление, не параллельное оси *z_axis*;

x_axis: (выход) направление, совпадающее с направлением проекции вектора *arg* на плоскость, перпендикулярную оси *z_axis*.

6.3.1.8 Функция second_proj_axis

Функция *second_proj_axis* возвращает значение нормированного вектора, который одновременно является проекцией вектора *arg* на плоскость, перпендикулярную оси *z_axis*, и на плоскость, перпендикулярную оси *x_axis*. Если вектор *arg* равен «null», то функция возвращает проекцию вектора (0, 1, 0) на ось *z_axis*.

Спецификация на языке EXPRESS:

```
*)
FUNCTION second_proj_axis(z_axis, x_axis, arg: direction) : direction;
 LOCAL
  y_axis : vector;
  v   : direction;
  temp  : vector;
 END_LOCAL;

 IF NOT EXISTS(arg) THEN
  v := direction([0.0,1.0,0.0]);
 ELSE
  v := arg;
 END_IF;

 temp  := scalar_times_vector(dot_product(v, z_axis), z_axis);
 y_axis := vector_difference(v, temp);
 temp  := scalar_times_vector(dot_product(v, x_axis), x_axis);
 y_axis := vector_difference(y_axis, temp);
 y_axis := normalise(y_axis);
 RETURN(y_axis.orientation);
END_FUNCTION;
(*
```

Определения аргументов:

z_axis: (вход) направление, задающее локальную ось *Z*;

x_axis: (вход) направление, не параллельное оси *z_axis*;

arg: (вход) направление, используемое как первая аппроксимация направления оси *y_axis*;

y_axis.orientation: (выход) направление, определяемое сначала путем проецирования вектора *arg* на плоскость, перпендикулярную оси *z_axis*, а затем проецирования полученного результата на плоскость, перпендикулярную оси *x_axis*.

6.3.1.9 Функция cross_product

Функция *cross_product* возвращает векторное произведение двух входных направлений. Входные направления должны быть трехмерными и нормированными перед вычислением произведения. Результатом всегда является безразмерный вектор. Если входные направления параллельны или непараллельны, то функция возвращает вектор с модулем, равным «null», и атрибутом *orientation*, равным *arg1*.

Спецификация на языке EXPRESS:

```
*)
FUNCTION cross_product(arg1, arg2 : direction) : vector;

 LOCAL
  mag  : REAL;
  res  : direction;
  v1,v2 : LIST[3:3] OF REAL;
  result : vector;
 END_LOCAL;
 IF ( NOT EXISTS (arg1) OR (arg1.dim - 2)) OR
```

```
    ( NOT EXISTS (arg2) OR (arg2.dim = 2)) THEN
   RETURN(?);
  ELSE
   BEGIN
    v1 := normalise(arg1).direction_ratios;
    v2 := normalise(arg2).direction_ratios;
    res.direction_ratios[1] := (v1[2]*v2[3] - v1[3]*v2[2]);
    res.direction_ratios[2] := (v1[3]*v2[1] - v1[1]*v2[3]);
    res.direction_ratios[3] := (v1[1]*v2[2] - v1[2]*v2[1]);
    mag := 0.0;
    REPEAT i := 1 TO 3;
     mag := mag + res.direction_ratios[i]*res.direction_ratios[i];
    END_REPEAT;
    IF (mag > 0.0) THEN
     result.orientation := res;
     result.magnitude := SQRT(mag);
    ELSE
     result.orientation := arg1;
     result.magnitude := 0.0;
    END_IF;
    RETURN(result);
   END;
  END_IF;
END_FUNCTION;
(*
```

Определения аргументов:

arg1: (вход) направление, определяющее первый операнд векторного произведения;

arg2: (вход) направление, определяющее второй операнд векторного произведения;

result: (выход) произведение векторов *arg1* и *arg2*.

6.3.1.10 Функция dot_product

Функция *dot_product* возвращает скалярное или точечное (.) произведение двух направлений. Входными аргументами могут быть направления, заданные в двумерном и трехмерном пространствах. Входные аргументы нормируются перед началом вычислений. Скалярное произведение не определено, если исходные направления имеют различную размерность или не определены.

Спецификация на языке EXPRESS:

```
*)
FUNCTION dot_product(arg1, arg2 : direction) : REAL;
 LOCAL
  scalar : REAL;
  vec1, vec2: direction;
  ndim : INTEGER;
 END_LOCAL;
 IF NOT EXISTS (arg1) OR NOT EXISTS (arg2) THEN
  scalar := ?;
  (* When function is called with invalid data
                        a NULL result is returned
*)
 ELSE
  IF (arg1.dim <> arg2.dim) THEN
   scalar := ?;
   (* When function is called with invalid data
                        a NULL result is returned
*)
   ELSE
    BEGIN
     vec1  := normalise(arg1);
     vec2  := normalise(arg2);
     ndim  := arg1.dim;
     scalar := 0.0;
```

```
    REPEAT i := 1 TO ndim;
     scalar := scalar +
           vec1.direction_ratios[i]*vec2.direction_ratios[i];
    END_REPEAT;
   END;
   RETURN (scalar);
  END_IF;
 END_IF;
END_FUNCTION;
(*
```

Определения аргументов функции:

arg1. (вход) направление, определяющее первый операнд точечного (скалярного) произведения;

arg2: (вход) направление, определяющее второй операнд точечного произведения;

result: (выход) скаляр, равный точечному произведению векторов *arg1* и *arg2*.

6.3.1.11 Функция normalize

Функция *normalize* возвращает вектор, компоненты которого нормированы так, что сумма их квадратов равна 1.0. Тип результата функции (направление или вектор в тех же единицах) совпадает с типом аргумента функции. Если аргумент функции не определен или имеет нулевой модуль, то выходной вектор также не определен.

Спецификация на языке EXPRESS:

```
*)
FUNCTION normalise(arg : vector_or_direction) : vector_or_direction;
 LOCAL
  ndim  : INTEGER;
  v   : direction;
  result : vector_or_direction;
  vec  : vector;
  mag  : REAL;
 END_LOCAL;

 IF NOT EXISTS (arg) THEN
  result := ?;
  (* When function is called with invalid data
                       a NULL result is returned *)
 ELSE
  ndim := arg.dim;
  IF 'API_ABSTRACT_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
   BEGIN
    vec := arg;
    v := arg.orientation;
    IF arg.magnitude = 0.0 THEN
     RETURN(?);
    ELSE
     vec.magnitude := 1.0;
    END_IF;
   END;
  ELSE
   v := arg;
  END_IF;
  mag := 0.0;
  REPEAT i := 1 TO ndim;
   mag := mag + v.direction_ratios[i]*v.direction_ratios[i];
  END_REPEAT;
  IF mag > 0.0 THEN
   mag := SQRT(mag);
   REPEAT i := 1 TO ndim;
    v.direction_ratios[i] := v.direction_ratios[i]/mag;
   END_REPEAT;
   IF 'API_ABSTRACT_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
```

```
    vec.orientation := v;
    result := vec;
   ELSE
    result := v;
   END_IF;
  ELSE
   RETURN(?);
  END_IF;
  RETURN (result);
 END_IF;
END_FUNCTION;
(*
```

Определения аргументов функции:

arg: (вход) нормированный вектор (направление);

result: (выход) вектор (направление) единичной длины, параллельный вектору *arg1*.

6.3.1.12 Функция scalar_times_vector

Функция *scalar_times_vector* возвращает вектор, полученный умножением исходного вектора на число. На вход функции подаются скаляр (число) и вектор, который может быть либо направлением, либо собственно вектором. На выходе функции получается вектор, вычисленный в тех же единицах, что и входной вектор. Если на входе указывается направление, то результат вычислений будет безразмерным. Если входной аргумент не определен, то значение функции также не определено. Если входной скаляр отрицателен, то ориентация вектора изменяется на противоположную.

Спецификация на языке EXPRESS:

```
*)
FUNCTION scalar_times_vector(scalar : REAL; vec : vector_or_direction)
                    : vector;
 LOCAL
  v   : direction;
  mag  : REAL;
  result : vector;
 END_LOCAL;

 IF NOT EXISTS (scalar) OR NOT EXISTS (vec) THEN
  result := ?;
  (* When function is called with invalid data
                        a NULL result is returned
*)
 ELSE
  IF 'API_ABSTRACT_SCHEMA.VECTOR' IN TYPEOF (vec) THEN
   v  := vec.orientation;
   mag := scalar * vec.magnitude;
  ELSE
   v  := vec;
   mag := scalar;
  END_IF;
  IF (mag < 0.0 ) THEN
   REPEAT i := 1 TO SIZEOF(v.direction_ratios);
    v.direction_ratios[i] := -v.direction_ratios[i];
   END_REPEAT;
   mag := -mag;
  END_IF;
  result.orientation := normalise(v);
  result.magnitude  := mag;
 END_IF;
  RETURN (result);
END_FUNCTION;
(*
```

Определения аргументов функции:

scalar: (вход) действительный числовой множитель;

vec: (вход) умножаемый вектор (направление);

result: (выход) вектор, являющийся результатом умножения *scalar* на *vec*.

6.3.1.13 Функция vector_sum

Функция *vector_sum* возвращает сумму векторов к входным аргументам. При этом направление рассматривается как единичный вектор. Входные аргументы должны иметь одинаковую размерность, они могут быть либо направлениями, либо векторами. Если оба входных аргумента — векторы, то они должны быть выражены в одинаковых единицах. Если оба входных вектора равны нулю, то на выходе получается вектор нулевого модуля с направлением *arg1*. Если оба входных аргумента — направления, то результат получается безразмерным.

Спецификация на языке EXPRESS:

```
*)
FUNCTION vector_sum(arg1, arg2 : vector_or_direction) : vector;
 LOCAL
  result     : vector;
  res, vec1, vec2 : direction;
  mag, mag1, mag2 : REAL;
  ndim       : INTEGER;
 END_LOCAL;

 IF ((NOT EXISTS(arg1)) OR (NOT EXISTS(arg2))) OR (arg1.dim <> arg2.dim)
   THEN
  result := ?;
  (* When function is called with invalid data
                        a NULL result is returned
*)
 ELSE
  BEGIN
   IF 'API_ABSTRACT_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
    mag1 := arg1.magnitude;
    vec1 := arg1.orientation;
   ELSE
    mag1 := 1.0;
    vec1 := arg1;
   END_IF;
   IF 'API_ABSTRACT_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN
    mag2 := arg2.magnitude;
    vec2 := arg2.orientation;
   ELSE
    mag2 := 1.0;
    vec2 := arg2;
   END_IF;
   vec1 := normalise (vec1);
   vec2 := normalise (vec2);
   ndim := SIZEOF(vec1.direction_ratios);
   mag := 0.0;
   REPEAT i := 1 TO ndim;
    res.direction_ratios[i] := mag1*vec1.direction_ratios[i] +
                 mag2*vec2.direction_ratios[i];
    mag := mag + (res.direction_ratios[i]*res.direction_ratios[i]);
   END_REPEAT;
   IF (mag > 0.0 ) THEN
    result.magnitude := SQRT(mag);
    result.orientation := res;
   ELSE
    result.magnitude := 0.0;
    result.orientation := vec1;
   END_IF;
  END;
 END_IF;
 RETURN (result);
END_FUNCTION;
(*
```

Определения аргументов:
arg1. (вход) направление, задающее первое слагаемое векторной суммы;
arg2: (вход) направление, задающее второе слагаемое векторной суммы;
result: (выход) вектор, являющийся суммой векторов *arg1* и *arg2*.

6.3.1.14 Функция vector_difference

Функция *vector_difference* вычисляет вектор, который является разностью векторов (*arg1* — *arg2*). Входные аргументы должны иметь одинаковую размерность, они могут быть либо направлениями, либо векторами. Если оба входных аргумента — векторы, то они должны быть выражены в одинаковых единицах. Если оба входных вектора равны нулю, то на выходе получается вектор нулевого модуля с направлением *arg1*. Если оба входных аргумента — направления, то результат получается безразмерным.

Спецификация на языке EXPRESS:

```
*)
FUNCTION vector_difference(arg1, arg2 : vector_or_direction) : vector;
 LOCAL
  result     : vector;
  res, vec1, vec2 : direction;
  mag, mag1, mag2 : REAL;
  ndim       : INTEGER;
 END_LOCAL;

 IF ((NOT EXISTS (arg1)) OR (NOT EXISTS (arg2))) OR (arg1.dim <> arg2.dim)
   THEN
  result := ?;
  (* When function is called with invalid data
                        a NULL result is returned
*)
 ELSE
  BEGIN
   IF 'API_ABSTRACT_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
    mag1 := arg1.magnitude;
    vec1 := arg1.orientation;
   ELSE
    mag1 := 1.0;
    vec1 := arg1;
   END_IF;
   IF 'API_ABSTRACT_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN
    mag2 := arg2.magnitude;
    vec2 := arg2.orientation;
   ELSE
    mag2 := 1.0;
    vec2 := arg2;
   END_IF;
   vec1 := normalise (vec1);
   vec2 := normalise (vec2);
   ndim := SIZEOF(vec1.direction_ratios);
   REPEAT i := 1 TO ndim;
    res.direction_ratios[i] := mag1*vec1.direction_ratios[i] -
                 mag2*vec2.direction_ratios[i];
    mag := mag + (res.direction_ratios[i]*res.direction_ratios[i]);
   END_REPEAT;
   IF (mag > 0.0 ) THEN
    result.magnitude := SQRT(mag);
    result.orientation := res;
   ELSE
    result.magnitude := 0.0;
    result.orientation := vec1;
   END_IF;
  END;
 END_IF;
 RETURN (result);
END_FUNCTION;
(*
```

Определения аргументов:

arg1. (вход) направление, определяющее уменьшаемое операции вычитания векторов;

arg2: (вход) направление, определяющее результат операции вычитания векторов;

result: (выход) вектор, являющийся разностью векторов *arg1* и *arg2*.

6.3.1.15 Функция constraints_composite_curve_on_surface

Функция *constraints_composite_curve_on_surface* проверяет условие, что все кривые, на которые производится ссылка сегментами *composite_curve_on_surface*, также являются кривыми на поверхности, включая *composite_curve_on_surface*, которые могут быть ограниченными кривыми *bounded_curve*.

Спецификация на языке EXPRESS:

```
*)
FUNCTION constraints_composite_curve_on_surface
       (c : composite_curve_on_surface)  : BOOLEAN;
 LOCAL
  n_segments : INTEGER := SIZEOF(c.segments);
 END_LOCAL;

 REPEAT k := 1 TO n_segments;
  IF (NOT('GEOMETRY_SCHEMA.PCURVE' IN
     TYPEOF (c\composite_curve.segments[k].parent_curve))) AND
    (NOT('API_ABSTRACT_SCHEMA.SURFACE_CURVE' IN
     TYPEOF (c\composite_curve.segments[k].parent_curve))) AND
    (NOT('API_ABSTRACT_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN
     TYPEOF (c\composite_curve.segments[k].parent_curve))) THEN
   RETURN (FALSE);
  END_IF;
 END_REPEAT;
 RETURN (TRUE);
END_FUNCTION;
(*
```

Определение аргумента:

c: (вход) проверяемая комбинированная кривая на поверхности.

6.3.1.16 Функция get_basis_surface

Функция *get_basis_surface* определяет базовую поверхность кривой как множество поверхностей. Если данная кривая не является сущностью *curve_on_surface*, то результатом будет пустое множество.

Спецификация на языке EXPRESS:

```
*)
FUNCTION get_basis_surface(c : curve_on_surface) : SET[0:2] OF surface;
 LOCAL
  surfs : SET[0:2] OF surface;
  n     : INTEGER;
 END_LOCAL;

 surfs := [];
 IF 'GEOMETRY_SCHEMA.PCURVE' IN TYPEOF (c) THEN
  surfs := [c\pcurve.basis_surface];
 ELSE
  IF 'API_ABSTRACT_SCHEMA.SURFACE_CURVE' IN TYPEOF (c) THEN
   n := SIZEOF(c\surface_curve.associated_geometry);
   REPEAT i := 1 TO n;
   surfs := surfs +
        associated_surface(c\surface_curve.associated_geometry[i]);
   END_REPEAT;
  END_IF;
 END_IF;
 IF 'API_ABSTRACT_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF (c) THEN
  (* For a composite_curve_on_surface the basis_surface is the
   intersection of the basis_surface of all the segments.
*)
```

```
    n := SIZEOF(c\composite_curve_on_surface.segments);
    surfs := get_basis_surface(c\composite_curve_on_surface.segments[1].
                               parent_curve);
    IF n > 1 THEN
     REPEAT i := 2 TO n;
      surfs := surfs *
               get_basis_surface(c\composite_curve_on_surface.
                         segments[1].parent_curve);
     END_REPEAT;
    END_IF;
   END_IF;
   RETURN(surfs);
  END_FUNCTION;
  (*
```

Определения аргументов:

c: (вход) кривая, для которой определяется базовая поверхность;

surf: (выход) множество, содержащее базовую поверхность или поверхность, на которой лежит кривая *c*.

6.3.1.17 Функция list_to_array

Функция *list_to_array* преобразует групповой перечень в регулярный массив с предварительно определенными границами. Если границы массива несовместимы с количеством элементов исходного перечня, то получается нулевой результат. Настоящая функция используется для создания регулярных массивов контрольных точек и совокупности последовательности сложнопрофильных кривых (b-spline).

Спецификация на языке EXPRESS:

```
*)
FUNCTION list_to_array(lis : LIST [0:?] OF GENERIC : T;
           low,u : INTEGER) : ARRAY[low:u] OF GENERIC : T;
 LOCAL
  n   : INTEGER;
  res : ARRAY [low:u] OF GENERIC : T;
 END_LOCAL;

 n := SIZEOF(lis);
 IF (n <> (u-low +1)) THEN
  RETURN(?);
 ELSE
  REPEAT i := 1 TO n;
   res[low+i-1] := lis[i];
  END_REPEAT;
  RETURN(res);
 END_IF;
END_FUNCTION;
(*
```

Определения аргументов функции:

lis: (вход) преобразуемый исходный перечень;

low: (вход) целое число, равное нижнему индексу выходного массива;

u: (вход) целое число, равное верхнему индексу;

res: (выход) регулярный массив, сгенерированный из исходных данных.

6.3.1.18 Функция make_array_of_array

Функция *make_array_of_array* создает регулярный массив из нескольких массивов, полученных из перечня перечней. Сначала функция проверяет совместимость размерности массива с размерами перечней. Необходимо, чтобы все подперечни содержали одинаковое количество элементов. Получается нулевой результат, если исходные данные функции несовместимы с размерностью массива. Функция используется для создания регулярных массивов контрольных точек и весовых коэффициентов

для поверхности, представленной совокупностью последовательности сложнопрофильных кривых (b-spline).

Спецификация на языке EXPRESS:

```
*)
FUNCTION make_array_of_array(lis : LIST[1:?] OF LIST [1:?] OF GENERIC : T;
                low1, u1, low2, u2 : INTEGER):
        ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
 LOCAL
  n1,n2 : INTEGER;
  res  : ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
  res1 : LIST[1:?] OF ARRAY [low2:u2] OF GENERIC : T;
 END_LOCAL;

 (* Check input dimensions for consistency
*)
 n1 := SIZEOF(lis);
 n2 := SIZEOF(lis[1]);

 IF (n1 <> (u1 -low1 + 1)) AND (n2 <> (u2 - low2 + 1)) THEN
  RETURN(?);
 END_IF;

 REPEAT i := 1 TO n1;
  IF (SIZEOF(lis[i]) <> n2) THEN
   RETURN(?);
  END_IF;
 END_REPEAT;

 (* Build a list of sub-arrays
*)
 REPEAT i := 1 TO n1;
  RESL[i] := list_to_array(lis[i],low2,u2);
 END_REPEAT;

 res := list_to_array(res1,low1,u1);
 RETURN(res);
END_FUNCTION;
(*
```

Определения аргументов функции:

lis: (вход) перечень преобразуемых перечней;

low1: (вход) целое число, описывающее требуемый нижний индекс первого выходного массива,

u1: (вход) целое число, описывающее верхний индекс первого выходного массива;

low2: (вход) целое число, описывающее требуемый нижний индекс второго выходного массива;

u2: (вход) целое число, описывающее верхний индекс второго выходного массива;

res: (выход) регулярный массив, составленный из массивов с заданными размерностями, сгенерированными из исходных данных после проверки соответствия указанных размерностей.

6.3.2 Определения функций схемы API_ABSTRACT_SCHEMA: поддержка ресурсов

Настоящий подраздел описывает функции, определенные ИСО 10303-41 и являющиеся частью схемы *api_abstract_schema*.

6.3.2.1 Функция bag_to_set

Функция *bag_to_set* преобразует элементы BAG (множество с повторяющимися элементами) в элементы SET.

Пример — Функция может использоваться для преобразования элементов BAG, полученных с помощью функции USEDIN, в элементы SET. Полученные значения могут быть назначены переменной типа SET.

Спецификация на языке EXPRESS:

```
*)
FUNCTION bag_to_set(the_bag : BAG OF GENERIC : intype)
              : SET OF GENERIC : intype;

 LOCAL
  the_set : SET OF GENERIC : intype := [];
  i     : INTEGER;
 END_LOCAL;

 IF SIZEOF (the_bag) > 0 THEN
  REPEAT i := 1 TO HIINDEX (the_bag);
   the_set := the_set + the_bag [i];
  END_REPEAT;
 END_IF;

 RETURN (the_set);

END_FUNCTION;
(*
```

Определение аргументов функции:

the_bag: элементы BAG, преобразуемые в элементы SET.

6.3.3 Определения функций схемы API_ABSTRACT_SCHEMA: структуры представлений

Настоящий подраздел описывает функции, определенные ИСО 10303-43 и являющиеся частью схемы *api_abstract_schema*.

6.3.3.1 Функция acyclic_mapped_representation

Функция *acyclic_mapped_representation* определяет, является ли заданный элемент отображения *mapped_item* самоопределяющимся (путем отображения представления, в котором используется указанный элемент). Функция имеет расширение, позволяющее рекурсивно отображать представления целиком *mapped_representation*, а также их элементы *mapped_representation.item* для любого элемента отображения или элемента представления (со ссылкой на элемент отображения). Указанная ссылка может оказаться циклической. Функция возвращает значение «true», если входной элемент представления не приводит к самоопределению. В противном случае функция возвращает значение «false». Функция имеет булев тип.

Настоящая функция задает ограничения сущности *mapped_item*.

Спецификация на языке EXPRESS:

```
*)
FUNCTION acyclic_mapped_representation
 (parent_set  : SET OF representation;
  children_set : SET OF representation_item) : BOOLEAN;
 LOCAL
  x,y : SET OF representation_item;
  i,j : INTEGER;
 END_LOCAL;
 -- Determine the subset of children_set that are mapped_items.
 x := QUERY(z <* children_set | 'API_ABSTRACT_SCHEMA.MAPPED_ITEM'
    IN TYPEOF(z));
 -- Determine that the subset has elements.
 IF SIZEOF(x) > 0 THEN
  -- Check each element of the set.
  REPEAT i := 1 TO HIINDEX(x);
   -- If the selected element maps a representation in the
   -- parent_set, return false.
   IF x[i]\mapped_item.mapping_source.mapped_representation
    IN parent_set THEN
    RETURN (FALSE);
   END_IF;
```

```
    -- Recursively check the items of the mapped_rep.
    IF NOT acyclic_mapped_representation
     (parent_set +
      x[i]\mapped_item.mapping_source.mapped_representation,
      x[i]\mapped_item.mapping_source.mapped_representation.items) THEN
      RETURN (FALSE);
    END_IF;
   END_REPEAT;
  END_IF;
  -- Determine the subset of children_set that are not mapped_items.
  x := children_set - x;
  -- Determine that the subset has elements.
  IF SIZEOF(x) > 0 THEN
   -- For each element of the set:
   REPEAT i := 1 TO HIINDEX(x);
    -- Determine the set of representation_items referenced.
    y := QUERY(z <* bag_to_set( USEDIN(x[i], '') ) |
       'API_ABSTRACT_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));
    -- Recursively check these in case they might be an offending
    -- mapped_item. Return false for any errors encountered.
    IF NOT acyclic_mapped_representation(parent_set, y) THEN
     RETURN (FALSE);
    END_IF;
   END_REPEAT;
  END_IF;
  -- Return true when all elements are checked and
  -- no error conditions found.
  RETURN (TRUE);
 END_FUNCTION;
 (*
```

Определения аргументов:

parent_set: множество представлений, использующих элементы отображения. Указанное множество составляет входные данные функции. При первом обращении к программе это множество, содержащее проверяемый элемент отображения, который используется и модифицируется при последующих вызовах функции;

children_set: множество элементов представления, которые могут быть элементами отображения. На них прямо или косвенно могут производиться ссылки элементами представления *parent_set*. Указанное множество образует входные данные функции. При первом обращении к функции его элементы представляют собой проверяемые элементы отображения, которые модифицируются при последующих вызовах функции.

6.3.3.2 Функция item_in_context

Функция *item_in_context* устанавливает, связан ли элемент представления *representation_item* с контекстом представления *representation_context*. Функция возвращает значение «true», если:

- аргумент *item* связан представлением с входным аргументом *cntxt*;

- аргумент *item* связан своим представлением определения *definitional_representation* с входным аргументом *cntxt*.

В противном случае функция *item_in_context* возвращает значение «false». Функция имеет булев тип.

Элемент представления связан с контекстом представления, если на него производится ссылка:

1) на множестве элементов представления, где аргумент *cntxt* является элементом контекста;

2) на множестве элементов представления определения *definitional_representation_item*, где аргумент *cntxt* является элементом контекста;

3) элементом представления, который описывается сущностью *item_in_context* в контексте *cntxt*.

Примечание 1 — Третьим условием является рекурсивная проверка, допускающая связь элемента представления и контекста представления, как ветвей дерева связанных элементов представления. Корнем дерева является сущность, связанная с контекстом представления, если удовлетворяются первое или второе условия.

Примечание 2 — Функция *item_in_context* устанавливает наличие связи элемента с особым контекстом представления. Соотношение элемента и прочих контекстов представления не рассматривается.

Спецификация на языке EXPRESS:

```
*)
FUNCTION item_in_context
  (item : representation_item;
   cntxt : representation_context) : BOOLEAN;

  LOCAL
   i : INTEGER;
   y : BAG OF representation_item;
  END_LOCAL;
  -- If there is one or more representation using both the item
  -- and cntxt return true.
  IF SIZEOF(USEDIN(item,'API_ABSTRACT_SCHEMA.REPRESENTATION.ITEMS')
   * cntxt.representations_in_context) > 0 THEN
   RETURN (TRUE);
   -- Determine the bag of representation_items that reference item.
   ELSE
    y := QUERY(z <* USEDIN (item , '')
       'API_ABSTRACT_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));

    -- Ensure that the set is not empty.
    IF SIZEOF(y) > 0 THEN

    -- For each element in the set
    REPEAT i := 1 TO HIINDEX(y);

     -- check to see it is an item in the input cntxt.
     IF item_in_context(y[i], cntxt) THEN
      RETURN (TRUE);
     END_IF;
    END_REPEAT;
   END_IF;
  END_IF;
  -- Return false when all possible branches have been checked
  -- with no success.
  RETURN (FALSE);
 END_FUNCTION;
 (*
```

Определения аргументов:

item: элемент представления, проверяемый на предмет наличия связи с контекстом *cntxt*. Входной аргумент функции;

cntxt: контекст представления, для которого определяется соотношение с аргументом *item*. Входной аргумент функции.

6.3.3.3 Функция using_representations

Функция *using_representations* возвращает множество представлений с используемыми элементами представления *representation_items*.

Элемент представления используется в некотором представлении *representation*, если на него производится ссылка:

1) во множестве элементов представления;

2) элементом представления, используемым в представлении.

Примечание — Вторым условием является последовательная проверка, допускающая использование элемента представления в некотором представлении в качестве ветви дерева соответствующих элементов представления. Корнем дерева является сущность, используемая в представлении, удовлетворяющем первому условию.

Спецификация на языке EXPRESS:

```
*)
FUNCTION using_representations(item : representation_item)
 : SET OF representation;
 LOCAL
  results       : SET OF representation;
  result_bag    : BAG OF representation;
  intermediate_items : SET OF representation_item;
  i          : INTEGER;
 END_LOCAL;
 -- Find the representation in which the item is used and add to the
 -- results set.
 result_bag := USEDIN(item,'API_ABSTRACT_SCHEMA.REPRESENTATION.ITEMS');
 IF SIZEOF(result_bag) > 0 THEN
  REPEAT i := 1 TO HIINDEX(result_bag);
   results := results + result_bag[i];
  END_REPEAT;
 END_IF;
 -- Find the set of representation_items in which the item is used.
 intermediate_items := QUERY(z <* bag_to_set( USEDIN(item , '')) |
  'API_ABSTRACT_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));
 -- If the set of intermediate items is not empty;
 IF SIZEOF(intermediate_items) > 0 THEN
  -- For each element in the set recursively add the
  -- using_representations of that element.
  REPEAT i := 1 TO HIINDEX(intermediate_items);
   results := results + using_representations(intermediate_items[i]);
  END_REPEAT;
 END_IF;
 -- Return the set of representation in which the input item is used
 -- directly and indirectly (through intervening representation_items).
 RETURN (results);
END_FUNCTION;
(*
```

Определение аргумента:

item: элемент представления, для которого определяется представление. Входной аргумент функции.

6.3.4 Определения функций схемы API_ABSTRACT_SCHEMA: функции интерфейса прикладного программирования

Настоящий подраздел описывает функции интерфейса прикладного программирования, используемые схемой *api_abstract_schema*.

6.3.4.1 Функция tree_api_group_structure

Функция *tree_api_group_structure* устанавливает, является ли структура группы интерфейса прикладного программирования *api_group*, корнем которой является аргумент *group*, структурой дерева, установленной соотношением *api_group_assignment*.

Функция возвращает значение «true», если ни одна из групп *api_group* (назначенных для аргумента *group* прямо или косвенно) не назначена дважды. В противном случае функция возвращает значение «false».

Функция *tree_api_group_structure* вызывает функцию назначенной группы *assigned_api_group*, которая рекурсивно вычисляет группу *api_group*, назначенную для аргумента *group*.

Спецификация на языке EXPRESS:

```
*)
FUNCTION tree_api_group_structure(group : api_group) : BOOLEAN;
 LOCAL
  i        : INTEGER;
  j        : INTEGER;
  assigned_group : BAG [1:?] OF api_group;
 END_LOCAL;
 -- Determine the bag of the api_groups that are assigned to the
 -- group argument.
 assigned_group:= assigned_api_group (group);
 -- Determine that the bag has elements.
 IF SIZEOF(assigned_group) > 0 THEN
  -- Check each element of the bag against each element.
  REPEAT i := 1 TO HIINDEX(assigned_group);
   REPEAT j := 1 TO HIINDEX(assigned_group);
    -- If the two elements are the same
    -- return false.
    IF ((assigned_group[i] :=: assigned_group[j]) AND (i<> j)) THEN
     RETURN (FALSE);
    END_IF;
   END_REPEAT;
  END_REPEAT;
 END_IF;
 -- Return true when all elements are checked and
 -- no error conditions found.
 RETURN (TRUE);
END_FUNCTION;
(*
```

6.3.4.1.1 Функция assigned_api_group

Функция *assigned_api_group* используется функцией *tree_api_group_structure* для рекурсивного вычисления групп интерфейса прикладного программирования *api_group*, назначенных для аргумента *group*.

Спецификация на языке EXPRESS:

```
*)
FUNCTION assigned_api_group( group : api_group) : BAG [0:?] OF api_group;
 LOCAL
  i            : INTEGER;
  assignment       : SET OF api_group_assignment;
  assigned_items     : BAG OF api_grouped_item;
  local_assigned_groups : BAG OF api_group;
  assigned_groups     : BAG OF api_group;
 END_LOCAL;
 assigned_items     := [];
 local_assigned_groups := [];
 -- Determine the subset of the api_group_assignments that assign items to
 -- the group argument .
 assignment := USEDIN (group, 'API_ABSTRACT_SCHEMA.API_GROUP_ASSIGNMENT\'
               + 'GROUP_ASSIGNMENT.ASSIGNED_GROUP');
 -- gathers all the api_groupeed_items
 REPEAT i := 1 TO HIINDEX(assignment);
  assigned_items := assigned_items + assignment[i].items;
 END_REPEAT;
 -- Determine the subset of api_groupeed_item that are api_groups.
 local_assigned_groups := QUERY( z <* assigned_items |
                  'API_ABSTRACT_SCHEMA.API_GROUP' IN
                  TYPEOF(z)
                 );
```

```
 -- initializes the assigned_groups
 assigned_groups := local_assigned_groups;
 -- Determine that the subset has elements.
 IF SIZEOF(local_assigned_groups) > 0 THEN
  -- compute all the assigned api_group of the bag.
  REPEAT i := 1 TO HIINDEX(local_assigned_groups);
   assigned_groups := assigned_groups +
              assigned_api_group(local_assigned_groups[i]);
  END_REPEAT;
 END_IF;
RETURN (assigned_groups);
END_FUNCTION;
(*
```

6.3.4.2 Функция tree_api_set_structure

Функция *tree_api_set_structure* устанавливает факт структуризации сущности *api_set* (корнем которой является аргумент *set*) в виде дерева с помощью соотношения *api_set_assignment*.

Функция возвращает значение «true», если ни одна из структур *api_set* (назначенных аргументом *set*) не назначается дважды, прямо или косвенно, в противном случае функция возвращает значение «false».

Функция *tree_api_set_structure* вызывает функцию *assigned_api_set*, которая рекурсивно вычисляет значение сущности *api_set*, назначенной для аргумента *the_set*.

Спецификация на языке EXPRESS:

```
*)
FUNCTION tree_api_set_structure(the_set : api_set) : BOOLEAN;
 LOCAL
  i       : INTEGER;
  j       : INTEGER;
  assigned_set : BAG [1:?] OF api_set;
 END_LOCAL;
 -- Determine the bag of the api_sets that are assigned to the set
 -- argument .
 assigned_set:= assigned_api_set(the_set);
 -- Determine that the bag has elements.
 IF SIZEOF(assigned_set) > 0 THEN
  -- Check each element of the bag against each element.
  REPEAT i := 1 TO HIINDEX(assigned_set);
   REPEAT j := 1 TO HIINDEX(assigned_set);
    -- If the two elements are the same
    -- return false.
    IF ((assigned_set[i] :=: assigned_set[j]) AND (i<> j)) THEN
     RETURN (FALSE);
    END_IF;
   END_REPEAT;
  END_REPEAT;
 END_IF;
 -- Return true when all elements are checked and
 -- no error conditions found.
 RETURN (TRUE);
END_FUNCTION;
(*
```

6.3.4.2.1 Функция assigned_api_set

Функция *assigned_api_set* используется функцией *tree_api_set_structure* для рекурсивного вычисления значений сущности *api_set*, назначенной для аргумента *the_set*.

Спецификация на языке EXPRESS:

```
*)
FUNCTION assigned_api_set( the_set : api_set) : BAG [0:?] OF api_set;
 LOCAL
  i           : INTEGER;
  assignment      : SET OF api_set_assignment;
  assigned_items    : BAG OF api_set_item;
  local_assigned_sets : BAG OF api_set;
  assigned_sets     : BAG OF api_set;
 END_LOCAL;
 assigned_items    := [];
 local_assigned_sets := [];
 -- Determine the subset of the api_set_assignments that assign items to
 -- the set argument .
 assignment := USEDIN(the_set, 'API_ABSTRACT_SCHEMA.API_SET_ASSIGNMENT\'
               + 'GROUP_ASSIGNMENT.ASSIGNED_GROUP');
 -- gathers all the api_set_items
 REPEAT i := 1 TO HIINDEX(assignment);
  assigned_items := assigned_items + assignment[i].items;
 END_REPEAT;
 -- Determine the subset of api_set_item that are api_sets.
 local_assigned_sets := QUERY( z <* assigned_items |
                'API_ABSTRACT_SCHEMA.API_SET' IN
                TYPEOF(z)
               );
 -- initializes the assigned_sets
 assigned_sets := local_assigned_sets;
 -- Determine that the subset has elements.
 IF SIZEOF(local_assigned_sets) > 0 THEN
  -- compute all the assigned api_set of the bag.
  REPEAT i := 1 TO HIINDEX(local_assigned_sets);
   assigned_sets := assigned_sets +
             assigned_api_set(local_assigned_sets[i]);
  END_REPEAT;
 END_IF;
RETURN (assigned_sets );
END_FUNCTION;
(*
```

6.3.4.3 Функция api_legal_style_number

Функция *api_legal_style_number* устанавливает допустимость стиля, назначенного для элемента геометрического представления *geometric_representation_item*, сгенерированного интерфейсом прикладного программирования.

Спецификация на языке EXPRESS:

```
*)
FUNCTION api_legal_style_number(item: styled_item): BOOLEAN;
 LOCAL
  repr  : SET [0:?] of representation;
  nb_style: INTEGER;
 END_LOCAL;
 -- only one presentation_style_assignment
 IF SIZEOF(item.styles) > 1 THEN RETURN (FALSE); END_IF;
 -- one style is always allowed
 IF SIZEOF(item.styles[1].styles) = 1 THEN RETURN (TRUE); END_IF;
 -- identification of geometric space dimensionality
 repr:=USEDIN (item, 'API_ABSTRACT_SCHEMA.REPRESENTATION.ITEMS');
 -- only geometric_representation_item may be styled
 IF SIZEOF (repr) = 0 THEN RETURN (FALSE); END_IF;
 IF ( NOT ('API_ABSTRACT_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEX' IN
      TYPEOF(repr[1].context_of_items) )
```

```
  )
THEN
 RETURN (FALSE);
END_IF;
-- no hidden line in 3D
IF dimension_of (item.item) = 3 THEN
 IF ( QUERY(temps <*item.styles[1].styles |
       ('API_ABSTRACT_SCHEMA.API_PREDEFINED_OCCLUSION_STYLE' IN
       TYPEOF (temps))
         OR
       ('API_ABSTRACT_SCHEMA.' +
       'API_PREDEFINED_VIRTUALLY_SENT_STYLE' IN
       TYPEOF(temps))
      ) <> []
   )
 THEN
  RETURN (FALSE); -- hidden line elimination is 2D
 END_IF;
 IF ( 'API_ABSTRACT_SCHEMA.ANNOTATION_FILL_AREA' IN
    TYPEOF (item.item)
   )
 THEN -- annotation fill area in 3D
  RETURN (
      NOT (SIZEOF(QUERY( f_a_style <* item.styles[1].styles |
                'API_ABSTRACT_SCHEMA.FILL_AREA_STYLE' IN
                TYPEOF(f_a_style)
               )
           ) <> SIZEOF(item.styles[1].styles) - 1
        )
          AND
      NOT (SIZEOF(QUERY( f_a_style <* item.styles[1].styles |
                'API_ABSTRACT_SCHEMA.' +
                'API_EXTERNALLY_DEFINED_FILL_AREA_STYLE'
                IN TYPEOF(f_a_style)
               )
           ) <> 1
        )
      );
 ELSE -- any other geometric_representation_item
  RETURN (SIZEOF(item.styles[1].styles) = 1);
 END_IF;
END_IF; -- end 3D context
-- case of 2D space
nb_style := SIZEOF (item.styles[1].styles);
IF ( SIZEOF(QUERY( st <* item.styles[1].styles |
          'API_ABSTRACT_SCHEMA.' +
          'API_PREDEFINED_VIRTUALLY_SENT_STYLE' IN
          TYPEOF(st)
         )
      ) = 1
  )
THEN
 IF ( SIZEOF(QUERY( st <* item.styles[1].styles |
           'API_ABSTRACT_SCHEMA.' +
           'API_PREDEFINED_OCCLUSION_STYLE' IN
           TYPEOF(st)
          )
       ) = 1
   )
 THEN
  nb_style:=nb_style-2;
 ELSE
```

```
    RETURN (FALSE); -- virtually sent shall be bl involved
   END_IF;
  ELSE -- not virually sent
   IF ( SIZEOF(QUERY( st <* item.styles[1].styles |
              'API_ABSTRACT_SCHEMA.' +
              'API_PREDEFINED_OCCLUSION_STYLE' IN
              TYPEOF(st)
             )
         ) = 1
     )
   THEN
    nb_style:=nb_style-1;
   END_IF;
  END_IF;
  IF ( 'API_ABSTRACT_SCHEMA.ANNOTATION_FILL_AREA' IN
     TYPEOF (item.item)
    )
  THEN
   nb_style:=nb_style + SIZEOF(QUERY( f_a_style<*item.styles[1].styles |
                      'API_ABSTRACT_SCHEMA.'+
                      'FILL_AREA_STYLE' IN
                      TYPEOF(f_a_style)
                     )
                 );
  END_IF;
  RETURN (nb_style <= 1);
  END_FUNCTION;
 (*
```

6.4 Глобальные правила схемы API_ABSTRACT_SCHEMA

Настоящий подраздел описывает глобальные правила, ассоциированные с рассмотренными выше сущностями (содержащимися в схеме *api_abstract_schema*) и ограничивающие их использование и их соотношения.

6.4.1 Правило unique_shape_representation

Правило *unique_shape_representation* требует существования уникальной сущности представления формы *shape_representation* схемы *api_abstract_schema*. Указанное представление формы соответствует форме продукта, созданного с помощью интерфейса базы данных CAD.

Спецификация на языке EXPRESS:

```
 *)
 RULE unique_shape_representation FOR (shape_representation);
 WHERE
  WR1: SIZEOF( QUERY ( SHAPE <* shape_representation | TRUE)) =1;
 END_RULE;
 (*
```

Спецификация на языке EXPRESS:

```
*)
 END_SCHEMA; -- end API_ABSTRACT schema
(*
```

7 Функциональные спецификации интерфейса

Примечание — Логическое описание функций интерфейса и их привязок на языке FORTRAN дано в приложении А.

7.1 Соглашения об обозначениях

7.1.1 Представления функций

Заголовок функции описывает:

I. Имя функции;

II. Минимальный уровень интерфейса, для которого функция является обязательной;

III. Минимальный уровень геометрической мощности, для которого функция является обязательной.

Для каждого параметра списка указаны:

IV. Является ли параметр входным или выходным;

V. Имя параметра;

VI. Описание данных (система обозначений типов приведена в разделе 7.1.2);

VII. Суть параметра;

VIII. Для каждой сущности указываются ее допустимые типы (например, линия, базовая поверхность, точка и т. п.).

Кроме того, для перечислимых типов данных (например, [TDB, CAD]), могут указываться допустимые значения, числа удвоенной длины, целочисленные данные, ограничения диапазонов значений [например, (EPS ≤ *x* ≤ MAX)].

Привязки языка FORTRAN зависят от:

IX. Синтаксиса языка FORTRAN в части использования функций function и подпрограмм subroutine (отображение логического типа на тип FORTRAN рассмотрено в приложении А, раздел А.2);

X. Эффективности используемых функций при условии отсутствия ошибок записи. Эффективность функций, за исключением функций запроса и функций восстановления состояния ошибки *Reset_Error_State*, обеспечивается, если:

1) интерфейс не находится в состоянии ошибки;

2) используемая функция не выявила ошибок.

В противном случае функция возвращается в прикладную программу, а во втором случае она устанавливает значения ошибок, равные *error_variable* (ошибка переменной), *error_text* (ошибка текста) и *error_origin* (ошибка адреса начала программы):

XI. Особых примечаний, необходимых для использования и реализации функции.

Имеются внутренние ссылки на:

XII. Разделы или подразделы настоящего стандарта, определяющие понятия или области определения сущностей используемых функций.

Ошибки, выявляемые функциями, идентифицируются:

XIII. Номером ошибки или символом «—» отсутствия ошибок;

XIV. Сопутствующим сообщением об ошибке.

Нижеследующий пример иллюстрирует порядок представления функции:

Пример — Раскладка представления функции:

Имя функции: ***Уровень интерфейса:*** ***(II)***

(I) ***Уровень геометрической мощности:*** ***(III)***

Параметры:

Вход (выход)	***Имя***	***Тип данных***	***Смысл***	***Допустимые тип/значение***
IV	***(V)***	***(VI)***	***(VII)***	***(VIII)***

Привязка языка FORTRAN: (IX)

Результат выполнения функции: (X)

Примечание: (XI)

Внутренняя ссылка: (XII)

Ошибки:

XIII	***(XIV)***

7.1.2 Представления типов данных

Функции используют либо простые типы данных, либо комбинации простых типов.

Таблица 16 — Простые типы данных

Обозначение	Тип данных	Описание
I	Integer	Целое число
D	Double	Число с максимальным количеством значащих цифр для используемого языка программирования
N	entity_name_type	Идентификатор сущности
E	Enumeration	Упорядоченное множество значений. Множество упорядочивается путем перенумерации идентификаторов элементов, приобретающих значения
S	String	Последовательность символов

Комбинации простых типов — перечни значений одного простого типа.

Например:

n × *I* — перечень целых значений;

n × *D* — перечень значений удвоенной длины;

n × *N* — перечень идентификаторов сущности *entity_name_type*;

n × *S* — перечень строк.

Символ *n* указывает, что переменная является целочисленной.

Диапазоны допустимых значений или перенумерованные значения могут быть ограничены:

- условием. Например: > 0° или (EPS ≤ параметр ≤ MAX);
- стандартным диапазоном целочисленных значений. Например: (1…3);
- диапазоном целочисленных значений, где максимум определяется реализацией и прочими ограничениями. Например: (1…*n*);
- перечнем перенумерованных значений. Например: [TDB, CAD].

7.1.3 Имена сущностей и аббревиатуры

Для описания допустимых подтипов сущностей функций интерфейса используются нижеследующие краткие имена.

Таблица 17 — Краткие имена типов сущностей

Краткое имя	Смысловое значение
a1p	Axis1_placement (размещение оси координат)
a2p	Axis2_placement (локальная координатная система)
Afa	Заполненная область комментариев annotation_fill_area
Aps	Api_planar_surface (плоская поверхность интерфейса прикладного программирования)
Arc	Api_circular_arc (дуга окружности интерфейса прикладного программирования)
Blk	Block (блок)
Brs	Boolean_result (результат булевой операции с телами)
Con	Right_circular_cone (прямой круглый конус)
Ctr	Api_contour (контур интерфейса прикладного программирования)
Cyl	Right_circular_cylinder (прямой круговой цилиндр)
Dir	Direction (направление)
Eas	Extruded_area_solid (тело, полученное экструдированием области)

Окончание таблицы 17

Краткое имя	Смысловое значение
Elc	Api_elliptical_arc (дуга эллипса интерфейса прикладного программирования)
Fsh	Fill_area_style_hatching (стиль штриховки заполненной области)
Grp	Api_group (группа интерфейса прикладного программирования)
Hss	Half_space_solid (тело в полупространстве)
Hyp	Api_hyperbolic_arc (дуга гиперболы интерфейса прикладного программирования)
Lin	Api_line (линия интерфейса прикладного программирования)
Par	Api_parabolic_arc (дуга параболы интерфейса прикладного программирования)
Pln	Polyline (полилиния)
Pnt	Cartesian_point (декартова точка)
Ras	Revolved_area_solid (тело, полученное вращением области вокруг заданной оси)
Set	Set (множество)
Sph	Sphere (сфера)
Tor	Torus (тор)
Wdg	Right_angular_wedge (прямой клин)

Таблица 18 — Краткие имена наборов типов сущностей

Краткое имя	Смысловое значение
Basic	Элемент набора (lin, arc)
Conic_arc	Элемент набора (elc, hyp, par)
Curves	Элемент набора (basic, conic_arc, general)
Csg	Элемент набора (blk, con, cyl, sph, tor, wdg, brs)
Fill_area	Элемент набора (afa, fsh)
General	Элемент набора (pln, ctr)
Math	Элемент набора (dir, a1p, a2p)
Solid_model	Элемент набора (csg, sweep)
Solids	Элемент набора (solid_model, hss)
Sweep	Элемент набора (eas, ras)

Примечание — Графические сущности обозначают элементы геометрического представления, создаваемые с помощью функций интерфейса в соответствии с рисунком 2 настоящего стандарта.

7.1.4 Имена функций

Имена функций формируются с помощью кратких имен типов сущностей, используемых функцией, и аббревиатур (см. таблицу 19). Все части имени функции начинаются с заглавной буквы, части имени разделены символом подчеркивания «_».

Пример — Функция «Дублирование и сдвиг сущности, заданной направлением» (Duplicate and Shift an Entity defined by a Direction) приобретает имя Dup_Shift_Dir_Ent с помощью нижеследующих сокращений названий отдельных частей:

- Dup означает «дублирование» (duplicate);
- Dir означает «направление» (direction);
- Ent означает «сущность» (entity).

Таблица 19 — Аббревиатуры, используемые для имен функций

Аббревиатура	Смысловое значение
Chg	Изменение
Dup	Дублирование
Ent	Сущность
Gen	Генерация (построение)
Inq	Запрос
Ref_Sys	Ссылочная система
Sld	Твердотельный

7.2 Логическое описание функций интерфейса и привязки языка FORTRAN

Приложение А описывает каждую функцию интерфейса и соответствующие привязки языка FORTRAN 90. Привязки прочих языков описаны в других частях настоящего стандарта.

8 Таблицы интерфейса

Настоящий раздел предоставляет описания всех входных данных таблицы интерфейса и их значения по умолчанию.

8.1 Таблица описаний интерфейса

Нижеследующая таблица содержит все возможные записи, используемые в настоящем стандарте для описания интерфейса прикладного программирования. Значения указанных записей зависят от реализации.

Таблица 20 — Таблица описаний интерфейса

Имя	Тип	Смысловое значение
interface_level	I	Уровень интерфейса: от 1 до 3
hidden_line_capability	E	Включение операций с невидимыми линиями [off = выключить, on = включить]
max_interpolation_nodes_number	I	Количество точек интерполяции для моделирования (= 1)
contour_entities	n × S	Перечень кратких имен сущностей, допустимых для создания контура api_contour, за исключением базовых сущностей (по крайней мере сущностей «lin» и «arc»)

8.2 Таблица статуса интерфейса

Содержит записи таблицы статуса интерфейса, используемые в настоящем стандарте. Указанные значения есть значения по умолчанию. Они могут быть изменены путем инициализации процесса построения вида, выполняемого LMS.

Таблица 21 — Таблица статуса интерфейса

Имя	Тип	Значение	Описание
error_variable (ошибка переменной)	I	0	Отсутствует
error_origin (ошибка адреса)	S	Пусто	Отсутствует
error_text (ошибка текста)	S	Пусто	Отсутствует

Окончание таблицы 21

Имя	Тип	Значение	Описание
geometrical_power_level (уровень геометрической мощности)	I	1	2D-вид
hidden_line (невидимые линии)	E	См. таблицу 20	Равно значению сущности hidden_line_capability
hidden_line_involved (включение невидимых линий)	E	True	Невидимые линии включены [TRUE, FALSE]
interpolation_nodes_number (количество узлов интерполяции)	I	См. таблицу 20	Равно значению сущности max_interpolation_nodes_number
view_length_unit (единица длины для вида)	E	Метр	Метр [metre, inch]
view_length_scale_factor (масштабный фактор длины для вида)	D	10^{-3}	Масштабный фактор
view_angle_unit (единица угла для вида)	E	Градус	Градус [rad, deg, grad]
point_style	2 × S	«asterisk_point» (звездочка), ISO_13584_31	Стиль воспроизведения точки
curve_style	2 × S	«plain_solid_line» (плоская сплошная линия), ISO_13584_31	Стиль воспроизведения кривой
fill_area_style	2 × S	«opaque_fill_area» (непрозрачная, заполненная область), ISO_13584_31	Стиль воспроизведения заполненной области
surface_style	2 × S	«solid_surface» (сплошная поверхность), ISO_13584_31	Стиль воспроизведения поверхности
hatch_curve_font	S	«continuous» (непрерывная)	Вид типа линий штриховки
hatch_width	S	«thin_hatch_line» (тонкие линии штриховки)	Толщина линий штриховки
hatch_colour	S	«hatch_line_colour» (цвет линий штриховки)	Вид цвета линий штриховки
hidden_line_aspect	S	«hidden_line_invisible» (скрытые невидимые линии)	Стиль невидимых линий для сущностей интерфейса высокого уровня
view_level	D	0.0	Относительный уровень вида (виртуальная высота)

9 Размерности реализации интерфейса

9.1 Минимальные размерности буферов интерфейса и структурированных типов данных

Настоящий подраздел устанавливает минимальные размерности структурированных типов данных и минимальные требуемые возможности хранения данных, поддерживаемых интерфейсом прикладного программирования в соответствии с настоящим стандартом.

Таблица 22 — Размерности реализации интерфейса

Размерность вида	Тип	Значение
Количество сущностей в TDB	I	10 000
Количество точек на каждую полилинию	I	300
Количество сущностей на каждый контур api_contour	I	300
Количество внутренних границ на каждую заполненную область комментариев annotation_fill_area	I	100
Количество внутренних границ на каждую плоскую поверхность api_planar_surface	I	100
Количество групп в TDB	I	200
Размер стека группы	I	100
Размер стека множества	I	100
Количество символов в строке	I	256

Приложение А
(справочное)

Логическое описание функций интерфейса и привязки языка FORTRAN

А.1 Введение

Данное приложение является неотъемлемой частью настоящего стандарта. Приложение дает логическое описание каждой функции интерфейса и соответствующих привязок языка FORTRAN.

А.2 Отображения языка FORTRAN

А.2.1 Отображения для функций интерфейса

Подпрограммы и функции языка FORTRAN полностью соответствуют логическим функциям. Каждая функция имеет уникальное имя соответствующей подпрограммы или функции на языке FORTRAN, по которому производится ее запуск. В настоящем приложении содержится как логическое описание функции интерфейса, так и соответствующей привязки языка FORTRAN.

Максимальная длина имени подпрограммы или функции языка FORTRAN не превышает 31 буквы. Имя на языке FORTRAN — это имя логической функции интерфейса, написанное заглавными буквами. Например, логическому имени *Dup_Shift_Dir_Ent* функции интерфейса (дублирования и сдвига сущности в заданном направлении) соответствует имя DUP_SHIFT_DIR_ENT на языке FORTRAN.

А.2.2 Отображение данных логического типа

Таблица А.1 — Отображение данных логического типа на языке FORTRAN

Данные логического типа	Отображение на языке FORTRAN
Integer	INTEGER (целые числа)
List of integer	INTEGER (целочисленный регулярный массив), где длина (размерность массива) задается целочисленной переменной или целочисленной константой
Double	DOUBLE PRECISION (двойная точность)
List of double	DOUBLE PRECISION (регулярный массив чисел двойной точности), в котором длина (размерность массива) задается целочисленной переменной или целочисленной константой
Enumeration	INTEGER (целочисленный тип); все значения находятся в диапазоне от 0 до N – 1, где N — количество перенумерованных элементов
Entity_name_type	INTEGER (целочисленный тип); нуль = 0; unknown (неизвестное значение) = отрицательное целое число
List of entity_name_type	INTEGER (регулярный массив имен сущностей), в котором длина (размерность массива) задается целочисленной переменной или целочисленной константой
String	CHARACTER (LEN = *), содержимое строки
List of string	CHARACTER (LEN = *), регулярный массив символов, в котором длина (размерность массива) задается целочисленной переменной (N) или целочисленной константой

А.2.3 Ограничения языка FORTRAN для программ поставщиков деталей

А.2.3.1 Основы языка

Основы синтаксиса языка программирования FORTRAN см. ИСО 1539:1991 (E).

А.2.3.2 Исключенные утверждения

Программы на языке FORTRAN используются библиотеками деталей. Работа указанных программ в различных вычислительных средах и различных CAD поддерживается вспомогательными технологиями (например, компиляцией, редактированием, интерпретацией, трансляцией и т. п.).

Для обеспечения работоспособности указанных программ запрещены нижеследующие утверждения языка FORTRAN:

1) утверждения программной архитектуры:

PROGRAM;
ENTRY;
STOP;
BLOCK DATA;
2) утверждения ввода/вывода:
READ, WRITE, FORMAT;
OPEN, CLOSE, INQUIRE;
REWIND, BACKSPACE, ENDFILE;
3) особые утверждения организации данных языка FORTRAN:
COMMON;
EQUIVALENCE;
DATA;
SAVE.

A.2.3.3 Устаревшие возможности языка FORTRAN

Нижеследующие возможности объявлены устаревшими в языке FORTRAN. Сейчас они все еще допустимы, но в следующей редакции стандарта использоваться уже не будут. Следовательно, нужно стараться не использовать их в новых программах и исключать из старых программ:

- Arithmetic-IF (условное выполнение арифметических операций);
- Alternate-Return from subroutine (альтернативный выход из подпрограммы);
- ASSIGN (назначение);
- Assigned FORMAT specifier (описание назначенного формата);
- Assigned GOTO (безусловный переход по метке);
- использование для организации цикла DO нецелочисленных переменных;
- использование циклов DO без окончания CONTINUE;
- использование ответвления END IF снаружи блока IF;
- использование описателя типа H;
- PAUSE (пауза).

A.2.3.4 Рекомендуемые утверждения языка FORTRAN

Следует избегать неявного описания типа данных. Поэтому каждая программа должна содержать нижеследующую директиву:

IMPLICIT NONE (неявное описание типа данных отсутствует).

A.3 Функции интерфейса

Настоящее подмножество содержит только функции интерфейса, используемые в библиотеках программ поставщиков деталей. Описание данных функций содержится в настоящем приложении.

A.3.1 Перечень функций интерфейса уровня 1

В таблице А.2 представлено подмножество функций интерфейса уровня 1 (см. раздел 5.1.1 и таблицу 20 «Таблица описаний интерфейса»). Данные функции работают в реализациях, соответствующих уровню 1 интерфейса.

Таблица А.2 — Перечень функций интерфейса уровня 1

Раздел	Имя функции	Параметр	Мощность
A.4.1	Функции управления данными		
A.4.1.1	Clear_Tdb (очистка временной базы данных)	—	≥ 0
A.4.1.2	Fix_Ent (выбор сущностей CAD)	N, ENTLST	≥ 1
A.4.2	Функции управления ошибками		
A.4.2.1	Inq_Error_State (запрос состояния ошибки)	ERRNUM, ERRSRC, ERRTXT, ERR	≥ 0
A.4.2.2	Reset_Error_State (перезагрузка состояния ошибки)	—	≥ 0
A.4.3	Функции запроса возможностей интерфейса		
A.4.3.1	Inq_Level (запрос уровня интерфейса)	LEVEL, ERR	≥ 0
A.4.3.2	Inq_Hidden_Line_Capability (запрос включения невидимых линий)	HLCAPA, ERR	≥ 0

Продолжение таблицы А.2

Раздел	Имя функции	Параметр	Мощность
A.4.3.3	Inq_Contour_Ent (запрос сущности контура)	N, TYPLST, ERR	≥ 0
A.4.3.4	Inq_Interface_Dimension (запрос размерности интерфейса)	NUMLST, ERR	≥ 0
A.4.4	Функции запроса системных записей интерфейса		
A.4.4.1	Inq_Hidden_Line (запрос о невидимых линиях)	HIDMOD, ERR	≥ 0
A.4.4.2	Inq_Hidden_Line_Involvement (запрос включения невидимых линий)	HLI, ERR	≥ 0
A.4.4.3	Inq_Interpolation_Nodes (запрос числа узлов интерполяции)	NODENO, ERR	≥ 0
A.4.4.4	Inq_Geometrical_Power (запрос геометрической мощности)	POWER, ERR	≥ 0
A.4.4.5	Inq_Ovc_Unit (запрос единиц измерения VOVC)	VLUNI, VLSFAC, VAUNI, ERR	≥ 0
A.4.5	Включение системных функций интерфейса		
A.4.5.1	Set_Hidden_Line_Involvement (включение невидимых линий)	HLI (HLI)	≥ 0
A.5.1.1	Direction (направление)		
A.5.1.1.1	Dir_component (составляющая направления)	X, Y, Z, KFIX	≥ 1
A.5.1.1.2	Dir_2_Pnt	STAPNT, ENDPNT, KFIX	≥ 1
A.5.1.1.3	Dir_2_Dir_angle	REFDIR, ZDIR, ANGLE, KFIX	≥ 1
A.5.1.1.4	Dir_A2p_X	A2PNAM, KFIX	≥ 1
A.5.1.1.5	Dir_A2p_Y	A2PNAM, KFIX	≥ 1
A.5.1.3	Axis2_placement (локальная координатная система)		
A.5.1.3.1	A2p_3_Pnt	CENPNT, AXSPNT, REFPNT, KFIX	≥ 1
A.5.1.3.2	A2p_2_Dir	CENPNT, AXSDIR, REFDIR, KFIX	≥ 1
A.5.1.3.3	A2p_2_Dir_Xy	CENPNT, REFDIR, YAXDIR, KFIX	≥ 1
A.5.1.3.4	A2p_Position_Relative	REFLST, KFIX	≥ 1
A.5.1.3.5	A2p_Ref_Sys	KFIX	≥ 1
A.5.2.1	Точки с каноническим определением		
A.5.2.1.1	Pnt_Cartesian_Absolute	X, Y, Z, KFIX	≥ 1
A.5.2.1.2	Pnt_Cartesian_Relative	PNTNAM, DX, DY, DZ, KFIX	≥ 1
A.5.2.1.3	Pnt_Polar_Absolute	PHI, THETA, RAD, KFIX	≥ 1
A.5.2.1.4	Pnt_Polar_Relative	PNTNAM, PHI, THETA, RAD, KFIX	≥ 1
A.5.2.2	Задание точки ограничениями		
A.5.2.2.1	Pnt_Begin_Ent	ENTNAM, KFIX	≥ 1
A.5.2.2.2	Pnt_End_Ent	ENTNAM, KFIX	≥ 1
A.5.2.2.3	Pnt_Intersection_2_Ent	ENTNM1, ENTNM2, KFIX	≥ 1

Продолжение таблицы А.2

Раздел	Имя функции	Параметр	Мощность
A.5.2.2.4	Pnt_Tangential_Arc	ARCNAM, LINNAM, KFIX	≥ 1
A.5.2.2.5	Pnt_Center_Arc	ARCNAM, KFIX	≥ 1
A.5.2.2.6	Pnt_Middle_Ent	ENTNAM, KFIX	≥ 1
A.5.2.2.7	Pnt_Projection_Ent	PNTNAM, ENTNAM, KFIX	≥ 1
A.5.2.2.8	Pnt_Projection_A2p	PNTNAM, A2PNAM, KFIX	≥ 1
A.5.3.1.1	Отрезки прямой (api_line)		
A.5.3.1.1.1	Lin_2_Pnt	STAPNT, ENDPNT, KFIX	≥ 1
A.5.3.1.1.2	Lin_Pnt_Length_Dir	STAPNT, LEN, DIRNAM, KFIX	≥ 1
A.5.3.1.1.3	Lin_Tangential_Arc	STAPNT, ARCNAM,KFIX	≥ 1
A.5.3.1.1.4	Lin_Tangential_2_Arc	ARCNM1, ARCNM2, KFIX	≥ 1
A.5.3.1.1.5	Lin_Chamfer_2_Lin	LEN1, LEN2, LINNM1, LINNM2, KFIX	≥ 1
A.5.3.1.2	Окружности и дуги окружностей (api_circular_arc)		
A.5.3.1.2.1	Circle_Rad_A2p	RAD, A2PNAM, SENSE, KFIX	≥ 1
A.5.3.1.2.2	Arc_3_Pnt	STAPNT ,INTPNT, ENDPNT, KFIX	≥ 1
A.5.3.1.2.3	Arc_Rad_2_Angle_A2p	RAD, STAANG, ENDANG, A2P-NAM, SENSE, KFIX	≥ 1
A.5.3.1.2.4	Arc_Rad_3_Pnt	RAD, STAPNT, ENDPNT, HLP-PNT, KFIX	≥ 1
A.5.3.1.2.5	Arc_Rad_2_Pnt_A2p	RAD, PNTNM1, PNTNM2, A2P-NAM, SENSE, KFIX	≥ 1
A.5.3.1.2.6	Arc_Fillet_2_Ent	ENTNM1, ENTNM2, RAD, KFIX	≥ 1
A.5.3.1.2.7	Arc_Tangential_2_Ent	ENTNM1, ENTNM2, RAD, KFIX	≥ 1
A.5.3.1.2.8	Arc_Rad_2_Ent	RAD, ENTNM1, ENTNM2, IN1, IN2, MINLEN, KFIX	≥ 1
A.5.3.1.2.9	Arc_3_Ent	ENTNM1, ENTNM2, ENTNM3, IN1, IN2, IN3, KFIX	≥ 1
A.5.3.2.1	Эллипс и дуга эллипса (api_elliptical_arc)		
A.5.3.2.1.1	Ellipse_2_Diameter_A2p	SEMI1, SEMI2, A2PNAM, SENSE, KFIX	≥ 1
A.5.3.2.1.2	Elc_Gen	SEMI1, SEMI2, STAANG, EN-DANG, A2PNAM, SENSE, KFIX	≥ 1
A.5.3.2.2	Построение дуги гиперболы (api_hyperbolic_arc)		
A.5.3.2.2.1	Hyp_Gen	SEMAXI, SEMIMG, STAANG, ENDANG, A2PNAM, KFIX	≥ 1
A.5.3.2.3	Гиперболические дуги (api_parabolic_arc)		
A.5.3.2.3.1	Par_Gen	FOCLEN, STAANG, ENDANG, A2PNAM, KFIX	≥ 1
A.5.3.3.1	Полилинии		

Продолжение таблицы А.2

Раздел	Имя функции	Параметр	Мощность
A.5.3.3.1.1	Pln_Cartesian_Coordinate	N, XLST, YLST, ZLST, KFIX	≥ 1
A.5.3.3.1.2	Pln_Pnt_List	N, PNTLST, KFIX	≥ 1
A.5.3.3.2	Плоский контур (api_contour)		
A.5.3.3.2.1	Ctr_Gen	N, ENTLST, KFIX	≥ 1
A.5.4	Заполненная область		
A.5.4.1	Afa_Gen	CTRNAM, N, CTRLST, KFIX	= 1
A.5.4.2	Fsh_Gen	REFPNT, DIST1, DIST2, ANGLE	= 1
A.5.4.3	Hatch_Afa	HATCH, AFA, REFPNT	= 1
A.6.1	Структурные сущности в TDB		
A.6.1.1	Create_Grp	—	≥ 1
A.6.1.2	Close_Grp	—	≥ 1
A.6.1.3	Reopen_Grp	GRPNAM	≥ 1
A.6.1.4	Remove_Ent_Grp	ENTNAM	≥ 1
A.6.1.5	Gather_Ent_Grp	N, ENTLST	≥ 1
A.6.1.6	Add_Ent_Grp	GRPNAM, ENTNAM	≥ 1
A.6.2	Структурные сущности, отправляемые в CAD		
A.6.2.1	Open_Set	SETNAM	≥ 1
A.6.2.2	Close_Set	—	≥ 1
A.7.1	Дублирующая сущность		
A.7.1.1	Dup_Ent	ENTNAM, KFIX	≥ 1
A.7.2	Зеркальное отражение сущностей		
A.7.2.1	Mirror_Ent	ENTNAM, LINNAM	≥ 1
A.7.2.2	Dup_Mirror_Ent	ENTNAM, LINNAM, KFIX	≥ 1
A.7.3	Сущности сдвига		
A.7.3.1	Shift_Dir_Ent	ENTNAM, DIRNAM, SHFLEN	≥ 1
A.7.3.2	Shift_Displacement_Ent	ENTNAM, DX, DY, DZ	≥ 1
A.7.3.3	Dup_Shift_Dir_Ent	ENTNAM, DIRNAM, SHFLEN, KFIX	≥ 1
A.7.3.4	Dup_Shift_Displacement_Ent	ENTNAM, DX, DY, DZ, KFIX	≥ 1
A.7.4	Сущности вращения		
A.7.4.1	Rotate_Ent	ENTNAM, PNTNAM, ANG1, ANG2, ANG3	≥ 1
A.7.4.2	Dup_Rotate_Ent	ENTNAM, PNTNAM, ANG1, ANG2, ANG3, KFIX	≥ 1
A.7.5	Изменение сущностей		
A.7.5.1	Chg_Orientation_Ent	ENTNAM	≥ 1

Продолжение таблицы А.2

Раздел	Имя функции	Параметр	Мощность
A.7.5.2	Chg_Sense_Ent	ENTNAM	≥ 1
A.7.5.3	Homotetia_Ent	ENTNAM, PNTNAM, K	≥ 1
A.8.1	Утилиты геометрических сущностей		
A.8.1.1	Pnt_Retrieve_Coordinate	PNTNAM, X, Y, Z	≥ 1
A.8.1.2	Dir_Retrieve_Component	DIRNAM, X, Y, Z	≥ 1
A.8.1.3	A2p_Retrieve_Location	A2PNAM, PNTNAM	≥ 1
A.8.1.4	Lin_Retrieve_Dir	LINNAM, DIRNAM	≥ 1
A.8.1.6	Arc_Retrieve_A2p	ARCNAM, A2PNAM	≥ 1
A.8.1.7	Arc_Retrieve_Rad	ARCNAM, RAD	≥ 1
A.8.1.8	Arc_Retrieve_Sense	ARCNAM, SENSE	≥ 1
A.8.2	Утилиты запроса сущности		
A.8.2.1	Retrieve_Type_Ent	ENTNAM, TYPE	≥ 1
A.8.2.2	Retrieve_Member_Grp	GRPNAM, N, ENTLST	≥ 1
A.8.2.3	Retrieve_Ent_Ctr	CTRNAM, N, ENTLST	≥ 1
A.8.3	Вычисление утилит		
A.8.3.1	Distance_2_Pnt	PNTNM1, PNTNM2	≥ 1
A.8.3.2	Start_Angle_Arc	ARCNAM	≥ 1
A.8.3.3	End_Angle_Arc	ARCNAM	≥ 1
A.9.1	Построение и задание новых ссылочных систем		
A.9.1.1	Ref_Sys_3_Pnt	CENPNT, AXSPNT, REFPNT	≥ 1
A.9.1.2	Ref_Sys_2_Dir	CENPNT, AXSDIR, REFDIR	≥ 1
A.9.1.3	Ref_Sys_2_Dir_Xy	CENPNT, REFDIR, YAXDIR	≥ 1
A.9.1.4	Ref_Sys_Position_Relative	REFLST	≥ 1
A.9.1.5	Ref_Sys_A2p	A2PNAM	≥ 1
A.10.1	Задание глобальных записей для атрибутов визуализации		
A.10.1.1	Set_Point_Style	EXTSOU, PNTSTY	≥ 0
A.10.1.2	Set_Curve_Style	EXTSOU, CURSTY	≥ 0
A.10.1.3	Set_Fill_Area_Style	EXTSOU, AFASTY	≥ 0
A.10.1.4	Set_Surface_Style	EXTSOU, SURSTY	≥ 0
A.10.1.5	Set_Hatching_Width	WIDTH, ERR	≥ 0
A.10.1.6	Set_Hatching_Curve_Font	FONT	≥ 0
A.10.1.7	Set_Hatching_Colour	COLOUR	≥ 0
A.10.1.8	Set_Hidden_Line_Aspect	HIDSTY	≥ 0
A.10.1.9	Set_Relative_View_Level	RVL	≥ 0
A.10.2	Запросы атрибутов визуализации из глобальных записей		

Окончание таблицы А.2

Раздел	Имя функции	Параметр	Мощность
A.10.2.1	Inq_Point_Style	EXTSOU, PNTSTY, ERR	≥ 0
A.10.2.2	Inq_Curve_Style	EXTSOU, CURSTY, ERR	≥ 0
A.10.2.3	Inq_Fill_Area_Style	EXTSOU, AFASTY, ERR	≥ 0
A.10.2.4	Inq_Surface_Style	EXTSOU, SURSTY, ERR	≥ 0
A.10.2.5	Inq_Hatching_Width	WIDTH, ERR	≥ 0
A.10.2.6	Inq_Hatching_Curve_Font	FONT, ERR	≥ 0
A.10.2.7	Inq_Hatching_Colour	COLOUR, ERR	≥ 0
A.10.2.8	Inq_Hidden_Line_Aspect	HIDSTY, ERR	≥ 0
A.10.2.9	Inq_Relative_View_Level	RVL, ERR	≥ 0
A.10.3	Изменение стиля воспроизведения сущностей		
A.10.3.1	Chg_Point_Style	PNTNAM, EXTSOU, PNTSTY	≥ 1
A.10.3.2	Chg_Curve_Style	ENTNAM, EXTSOU, CURSTY	≥ 1
A.10.3.3	Chg_Fill_Area_Style	AFAN AM, EXTSOU, AFASTY	≥ 1
A.10.3.4	Chg_Surface_Style	ENTNAM, EXTSOU, SURSTY	≥ 1
A.10.3.5	Chg_Hatching_Width	FSHNAM, WIDTH	≥ 1
A.10.3.6	Chg_Hatching_Curve_Font	FSHNAM, FONT	≥ 1
A.10.3.7	Chg_Hatching_Colour	FSHNAM, COLOUR	≥ 1
A.10.3.8	Chg_Hidden_Line_Aspect	ENTNAM, HIDSTY	≥ 1
A.10.3.9	Chg_Relative_View_Level	ENTNAM, RVL	≥ 1
A.10.4	Запрос стиля элемента из сущности		
A.10.4.1	Retrieve_Point_Style	PNTNAM, EXTSOU, PNTSTY	≥ 1
A.10.4.2	R et ri ev e_Curve_Style	ENTNAM, EXTSOU, CURSTY	≥ 1
A.10.4.3	Retrieve_Fill_Area_Style	AFANAM, EXTSOU, AFASTY	≥ 1
A.10.4.4	Retrieve_Surface_Style	ENTNAM, EXTSOU, SURSTY	≥ 1
A.10.4.5	Retrieve_Hatching_Width	FSHNAM, WIDTH	≥ 1
A.10.4.6	Retrieve_Hatching_Curve_Font	FSHNAM, FONT	≥ 1
A.10.4.7	Retrieve_Hatching_Colour	FSHNAM, COLOUR	≥ 1
A.10.4.8	Retrieve_Hidden_Line_Aspect	ENTNAM, HIDSTY	≥ 1
A.10.4.9	Retrieve_Relative_View_Level	ENTNAM, RVL	≥ 1

А.3.2 Перечень функций интерфейса уровня 2

В таблице А.3 представлено подмножество функций интерфейса, определяющее полное подмножество добавочных функций интерфейса уровня 2 (см. раздел 5.1.1 и таблицу 20 «Таблица описаний интерфейса»). Эти функции можно вызывать для реализации, соответствующей интерфейсу уровня 2, в добавление к функциям интерфейса уровня 1.

Таблица А.3 — Функции интерфейса уровня 2

Раздел	Имя функции	Параметр	Мощность
A.5.1.1	Направление		
A.5.1.1.6	Dir_A2P_Z	A2PNAM, KFIX	≥ 2
A.5.1.2	Axis1_placement (одна ось)		
A.5.1.2.1	A1p_Gen	PNTNAM, DIRNAM, KFIX	≥ 2
A.5.1.2.2	A1p_2_Pnt	STAPNT, ENDPNT, KFIX	≥ 2
A.5.2.1	Точки с каноническим определением		
A.5.2.1.5	PNT_Cylinder_Absolute	PHI, RAD, HEIGHT, KFIX	≥ 2
A.5.2.1.6	PNT_Cylinder_Relative	PNTNAM, PHI, RAD, HEIGHT, KFIX	≥ 2
A.5.5	Сущности поверхности		
A.5.5.1	Aps_Gen	CTRNAM, N, CTRLST, KFIX	≥ 2

А.3.3 Перечень функций интерфейса уровня 3

В таблице А.4 представлено подмножество функций интерфейса, определяющее полное подмножество дополнительных функций интерфейса уровня 3 (см. раздел 5.1.1 и таблицу 20 «Таблица описаний интерфейса»). Эти функции вызываются для реализации, соответствующей интерфейсу уровня 3, в добавление к функциям интерфейсов уровней 1 и 2.

Таблица А.4 — Функции интерфейса уровня 3

Раздел	Имя функции	Параметр	Мощность
A.5.6.1	Сущности конструктивной блочной геометрии		
A.5.6.1.1	Sph_Gen	RAD, CNTPNT, KFIX	= 3
A.5.6.1.2	Con_Gen	ANGLE, HEIGHT, RAD, A1PNAM, KFIX	= 3
A.5.6.1.3	Cyl_Gen	RAD, HEIGHT, A1PNAM, KFIX	= 3
A.5.6.1.4	Tor_Gen	MAJOR, MINOR, A1PNAM, KFIX	= 3
A.5.6.1.5	Blk_Gen	LENX, LENY, LENZ, A2PNAM, KFIX	= 3
A.5.6.1.6	Wdg_Gen	LENX, LENY, LENZ, LTX, A2PNAM, KFIX	= 3
A.5.6.2	Регуляризованные булевы операции конструктивной блочной геометрии		
A.5.6.2.1	Union_Sld	BOPNM1, BOPNM2, KFIX	= 3
A.5.6.2.2	Intersection_Sld	BOPNM1, BOPNM2, KFIX	= 3
A.5.6.2.3	Difference_Sld	BOPNM1, BOPNM2, KFIX	= 3
A.5.6.3	Сущности тел, полученные путем очерчивания		
A.5.6.3.1	Sld_Extrusion	SRFNAM, DIRNAM, DEPTH, KFIX	= 3
A.5.6.3.2	Sld_Revolution	SRFNAM, ANG, A1PNAM, KFIX	= 3
A.5.6.4	Построение «труб» в конструктивной блочной геометрии		
A.5.6.4.1	Sld_Pipe	PLNNAM, SRFNAM, RAD, KFIX	= 3
A.5.6.5	Построение тела в полупространстве		
A.5.6.5.1	Hss_Gen	A2PNAM, AGREMF	= 3
A.8.1	Утилиты геометрических сущностей		
A.8.1.5	Hss_Retrieve_A2p	HSSNAM, A2PNAM	= 3

А.4 Функции управления интерфейсом

Функции управления интерфейсом обеспечивают управление интерфейсом из прикладных программ (например, отправку геометрических данных в CAD), состоянием ошибки, запросом исходных данных из таблицы интерфейса.

А.4.1 Функции управления данными

Очистка временной базы данных	Clear_TDB
Выбор сущности CAD	Fix_Ent

А.4.1.1 Очистка временной базы данных

Имя функции:	Уровень интерфейса:	1
Clear_TDB	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
—	—	—	—	—

Привязка языка FORTRAN:
CALL CLEAR_TDB ( )

Результат применения функции

Очистка временной базы данных (TDB). При возникновении ошибки TDB не изменяется.

Примечание — Если существует открытый 2D-вид и сущности внутри TDB, отправленные виртуально (с назначенным стилем *api_pre_defined_vitually_sent_style*), то указанные сущности не должны быть стерты. Они остаются в TDB до окончания процесса удаления невидимых линий.

Внутренние ссылки: 5.3.4, 5.3.5, 6.2.5.5.

Ошибки

—	Ошибок нет		

А.4.1.2 Выбор сущностей CAD

Имя функции:	Уровень интерфейса:	1
Fix_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	N	1	Длина ENTLST	≥ 1
Ввод	ENTLST	n × N	Перечень имен сущностей	(pnt, curves, afa,aps, math, solid_model,grp)

Привязка языка FORTRAN:
CALL FIX_ENT (N, ENTLST)

Результат применения функции

Все сущности заданного перечня ENTLST удаляются из групповой структуры, которой они принадлежат, и отправляются из TDB в CAD. Доступ к указанным сущностям прекращается. Имена указанных сущностей становятся неизвестными. При возникновении ошибки никакие изменения не производятся.

Примечания

1 Если используемая сущность является экземпляром типа группы *api_group* интерфейса прикладного программирования, то данная группа должна быть замкнутой, все сущности, ссылающиеся на данную группу, направляются в CAD. Имя данной группы становится неизвестным, как и имена ссылочных сущностей.

2 Если существует открытый 2D-вид и запись *hidden_line* в таблице статуса интерфейса содержит ON (включено), то все сущности списка ENTLST с включенными невидимыми линиями (HLI) отправляются только виртуально. Это означает, что указанные сущности приобретают предварительно определенный стиль виртуальной отправки *api_pre_defined_vitually_sent_style*. После окончания процесса удаления невидимых линий они остаются внутри временных баз данных (TDB) до момента отправки в CAD.

3 Если сущность отправлена виртуально, то она удаляется из структуры *api_group*. Она должна принадлежать корневой группе, которая не может быть использована в какой-либо групповой функции.

Внутренние ссылки: 5.3.4, 5.3.5, 5.5, 6.1.19, 6.2.5.5.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
5	Целое значение находится вне допустимого диапазона	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности		

А.4.2 Функции управления ошибками

Запрос состояния ошибки	Inq_Error_State
Перезагрузка состояния ошибки	Reset_Error_State

А.4.2.1 Запрос состояния ошибки

Имя функции:	Уровень интерфейса:	1
Inq_Error_State	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	ERRNUM	I	Текущее значение параметра ошибки	0 или error_variable
Вывод	ERRSRC	S	Имя функции, в которой возникла ошибка	
Вывод	ERRTXT	S	Текст сообщения об ошибке	
Вывод	ERR	E	*error_indicator* (индикатор ошибки функции запроса)	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_ERROR_STATE (ERRNUM, ERRSRC, ERRTXT, ERR)

Результат использования функции

Функция обеспечивает текущий статус ошибки интерфейса *error_status* путем возвращения текущих значений переменных *error_variable*, *error_origin* и *error_text*.

Если интерфейс находится в состоянии ошибки (*error_state* = *true*), то:

- текущее значение ошибки переменной и текущие значения строк ошибки адреса и ошибки текста таблицы статуса интерфейса присваиваются переменной ERRNUM и строкам ERRSRC и ERRTXT соответственно;
- значение ERRNUM равно 0, и строки ERRSRC и ERRTXT являются пустыми.

Если интерфейс не находится в состоянии ошибки (*error_state* = *false*), то:

- индикатор ошибки *error_indicator* с именем ERR сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Нет.

Внутренние ссылки: 5.8.

Ошибки

—	Ошибок нет		

А.4.2.2 Перезагрузка состояния ошибки

Имя функции:	Уровень интерфейса:	1
Reset_Error_State	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
—	—	—	—	—

Привязка языка FORTRAN:
CALL RESET_ERROR_STATE ( )

Результат использования функции

Функция повторно формирует текущее состояние ошибки интерфейса.
Если интерфейс находится в состоянии ошибки (*error_state* = *true*), то:
- интерфейс выходит из состояния ошибки и повторно задает текущее значение ошибки переменной *error_variable* и текущие значения строк ошибки адреса *error_origin* и ошибки текста *error_text* в таблице статуса интерфейса. При этом значение ошибки переменной обнуляется, строки ошибки адреса и ошибки текста становятся пустыми.
Если интерфейс не находится в состоянии ошибки (*error_state* = *false*), то:
- функция не выполняется и никакие изменения не производятся.

Примечание — Нет.

Внутренние ссылки: 5.8.

Ошибки

—	Ошибок нет		

А.4.3 Функции запроса возможностей интерфейса

Запрос уровня интерфейса	Inq_Level
Запрос включения невидимых линий	Inq_Hidden_Line_Capability
Запрос сущности контура	Inq_Contour_Ent
Запрос размерности интерфейса	Inq_Interface_Dimension

А.4.3.1 Запрос уровня интерфейса

Имя функции:	Уровень интерфейса:	1
Inq_Level	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	LEVEL	I	Значение записи *interface_level*	(1…3)
Вывод	ERR	E	Индикатор запроса *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_LEVEL (LEVEL, ERR)

Результат использования функции

Функция доставляет значение записи *interface_level* из таблицы описаний интерфейса. Индикатор ошибки *error_indicator* (ERR) сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Нет.

Внутренние ссылки: 5.1.1, 8.1.

Ошибки

—	Ошибок нет		

А.4.3.2 Запрос включения невидимых линий

Имя функции:	Уровень интерфейса:	1
Inq_Hidden_Line_Capability	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	HLCAPA	E	Значение записи *hidden_line_capability*	[OFF, ON]
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_HIDDEN_LINE_CAPABILITY (HLCAPA, ERR)

Результат использования функции

Функция доставляет значение записи *hidden_line_capability* из таблицы описаний интерфейса. Индикатор ошибки *error_indicator* (ERR) сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Нет.

Внутренние ссылки: 5.3.5, 8.1.

Ошибки

—	Ошибок нет		

А.4.3.3 Запрос сущности контура

Имя функции:	Уровень интерфейса:	1
Inq_Contour_Ent	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	N	I	Длина перечня TYPLST (количество записей)	(2…6)
Вывод	TYPLST	n × S	Перечень кратких имен типов сущностей, допустимых для представления *api_contour*	(lin, arc, elc, par, hyp, pln)
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_CONTOUR_ENT (N, TYPLST, ERR)

Результат использования функции

Функция доставляет перечень кратких имен типов сущностей (допустимых для представления контура) в виде записи *contour_entities* из таблицы описаний интерфейса с двумя добавлениями по умолчанию. Минимальное количество типов сущностей перечня TYPLST равно 2 (это сущность «*lin*» представления линий *api_line* и сущность «*arc*» представления дуги окружности *api_circular_arc*, которые непременно должны присутствовать в каждой реализации интерфейса). Индикатор ошибки *error_indicator* (ERR) сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Количество символов в каждой позиции перечня TYPLST(i) равно 3.

Внутренние ссылки: 6.1.14.2, 7.1.3, 8.1.

Ошибки

—	Ошибок нет		

А.4.3.4 Запрос размерности интерфейса

Имя функции:	Уровень интерфейса:	1
Inq_Interface_Dimension	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	NUMLST	9 × I	Перечень минимальных размерностей буферов интерфейса	См. ниже
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_INTERFACE_DIMENSION (NUMLST, ERR)

Результат использования функции

Функция доставляет список целых чисел, равных минимальным размерностям буферов интерфейса, в соответствии с таблицей 21 настоящего стандарта:
NUMLST(1) — количество сущностей, содержащихся в TDB;
NUMLST(2) — количество узлов полилинии;
NUMLST(3) — количество сущностей на контуре *api_contour* интерфейса прикладного программирования;
NUMLST(4) — количество внутренних границ заполненной области комментариев *annotation_fill_area*;
NUMLST(5) — количество внутренних границ на плоской поверхности *api_planar_surface* интерфейса прикладного программирования;
NUMLST(6) — количество групп, содержащихся в TDB;
NUMLST(7) — размер стека группы;
NUMLST(8) — размер стека множества;
NUMLST(9) — количество символов в строке.
Индикатор ошибки *error_indicator* (ERR) сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Нет.

Внутренние ссылки: 9.1.

Ошибки

—	Ошибок нет		

А.4.4 Функции запроса системных записей интерфейса

Запрос о невидимых линиях	Inq_Hidden_Line
Запрос включения невидимых линий	Inq_Hidden_Line_Involvement
Запрос числа узлов интерполяции	Inq_Interpolation_Nodes
Запрос геометрической мощности	Inq_Geometrical_Power
Запрос единиц измерения OVC	Inq_OVC_Unit

А.4.4.1 Запрос о невидимых линиях

Имя функции:	Уровень интерфейса:	1
Inq_Hidden_Line	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	HIDMOD	E	Текущее значение записи *hidden_line*	[ON, OFF]
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_HIDDEN_LINE (HIDMOD, ERR)

Результат использования функции

Функция доставляет текущее значение записи *hidden_line* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* (ERR) сообщает о проблемах, возникающих в процессе использования функции.

Примечание — Нет.

Внутренние ссылки: 5.3.5, 8.2.

Ошибки

—	Ошибок нет		

А.4.4.2 Запрос включения невидимых линий

Имя функции:

Inq_Hidden_Line_Involvement

Уровень интерфейса:	1
Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	HLI	E	Текущее значение записи *hidden_line_involved*	[TRUE, FALSE]
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_HIDDEN_LINE_INVOLVEMENT (HLI, ERR)

Результат использования функции

Функция доставляет текущее значение записи *hidden_line_involved* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* (ERR) сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Нет.

Внутренние ссылки: 5.3.5, 8.2.

Ошибки

—	Ошибок нет		

А.4.4.3 Запрос числа узлов интерполяции

Имя функции:

Inq_Interpolation_Nodes

Уровень интерфейса:	1
Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	NODENO	I	Текущее значение записи *interpolation_nodes_number*	≥ 1
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_INTERPOLATION_NODES (NODENO, ERR)

Результат использования функции

Функция доставляет применимое и доступное значение записи *interpolation_nodes_number* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* (ERR) сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Нет.

Внутренние ссылки: 6.1.13, 8.1.

Ошибки

—	Ошибок нет		

А.4.4.4 Запрос геометрической мощности

Имя функции:	Уровень интерфейса:	1
Inq_Geometrical_Power	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	POWER	I	Текущее значение записи *geometrical_power_level*	(0…3)
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_GEOMETRICAL_POWER (POWER, ERR)

Результат использования функции

Функция доставляет текущее значение записи *geometrical_power_level* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* (ERR) сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Нет.

Внутренние ссылки: 5.1.1, 8.2.

Ошибки

—	Ошибок нет		

А.4.4.5 Запрос единиц измерения OVC

Имя функции:	Уровень интерфейса:	1
Inq_OVC_unit	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	VLUNI	E	Текущее значение записи *view_length_unit*	[METRE, INCH]
Вывод	VLSFAC	D	Текущее значение записи *view_length_scale_factor*	
Вывод	VAUNI	E	Текущее значение записи *view_angle_unit*	[RAD, DEG, GRAD]
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_OVC_UNIT (VLUNI, VLSFAC, VAUNI, ERR)

Результат использования функции

Функция доставляет единицы измерения длины *OVC_length_unit*, единицы измерения угла *OVC_angle_unit*, текущие значения единицы длины *view_length_unit*, масштабного фактора единицы длины *view_length_scale_factor* и единицы угла *view_angle_unit* из таблицы статуса интерфейса.

Единица измерения длины получается путем умножения текущего значения единицы на масштабный фактор единицы длины. Единица измерения угла соответствует единице угла.

Индикатор ошибки *error_indicator* (ERR) сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Нет.

Внутренние ссылки: 5.3.2, 8.2.

Ошибки

—	Ошибок нет		

А.4.5 Включение системных функций интерфейса

Включение невидимых линий	Set_Hidden_Line_Involvement

А.4.5.1 Включение невидимых линий

Имя функции:	Уровень интерфейса:	1
Set_Hidden_Line_Involvement	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	HLI	E	Значение записи *hidden_line_involved*	[TRUE, FALSE]

Привязка языка FORTRAN:
CALL SET_HIDDEN_LINE_INVOLVEMENT (HLI)

Результат использования функции

Функция устанавливает новое текущее значение записи *hidden_line_involved* таблицы статуса интерфейса, если значение параметра HLI равно «true». При этом каждая сущность кривой и каждая сущность заполненной области, созданная с помощью функции интерфейса, должна быть включена в процесс удаления невидимых линий. Если значение HLI равно «false», то указанные созданные сущности в данный процесс не включены. При возникновении ошибки никакие изменения не производятся.

Примечание — Нет.

Внутренние ссылки: 5.3.5, 8.2.

Ошибки

1001	Перечислимое значение находится вне установленного диапазона

А.5 Функции геометрических данных

А.5.1 Математические сущности

А.5.1.1 Сущность direction

Задание вектора направления его компонентами	Dir_Component
Задание вектора направления двумя точками	Dir_2_Pnt
Задание вектора направления двумя направлениями и углом	Dir_2_Dir_Angle
Задание оси *X* сущностью *axis2_placement*	Dir_A2p_X
Задание оси *Y* сущностью *axis2_placement*	Dir_A2p_Y
Задание оси *Z* сущностью *axis2_placement*	Dir_A2p_Z

A.5.1.1.1 Задание вектора направления его компонентами

Имя функции:	Уровень интерфейса:	1
Dir_Component	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	X	D	*X*-компонента в направлении оси (Ox) текущей OVC	См. примечание (1)
Ввод	Y	D	*Y*-компонента в направлении оси (Oy) текущей OVC	См. примечание (1)
Ввод	Z	D	*Z*-компонента в направлении оси (Oz) текущей OVC	См. примечание (1)
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданного направления direction	dir

Привязка языка FORTRAN:
NAME = DIR_COMPONENT (X, Y, Z, KFIX)

Результат использования функции

Создает сущность направления *direction* путем задания компонент направлений *direction_ratio* вдоль ортогональных направлений *x*, *y* и *z*, вычисленных по заданным параметрам *X*, *Y* и *Z* соответственно. Функция возвращает имя созданного направления. Полученному направлению назначается нулевой стиль *null_style*.

Значение модуля вектора направления лежит в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечания

1 Значения компонент |*X*|, |*Y*| или |*Z*| не должны попадать в интервал между нулевым значением *ZERO_value* и допуском EPS;

2 Если текущий открытый вид определен как 2D-вид (соответствует значению 1 записи *geometrical_power_level* таблицы статуса интерфейса), то значение компоненты *Z* игнорируется интерфейсом.

Внутренние ссылки: 6.1.9.3, 6.2.1.2.

Ошибки

7	Действительное значение находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
102	Модуль вектора направления находится вне установленного диапазона [EPS, MAX]	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.1.1.2 Задание вектора направления двумя точками

Имя функции:	Уровень интерфейса:	1
Dir_2_Pnt	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	STAPNT	N	Имя начальной декартовой точки *cartesian_point*	pnt
Ввод	ENDPNT	N	Имя конечной декартовой точки *cartesian_point*	pnt
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданного направления *direction*	dir

Привязка языка FORTRAN:
NAME = DIR_2_PNT (STAPNT, ENDPNT, KFIX)

Результат использования функции

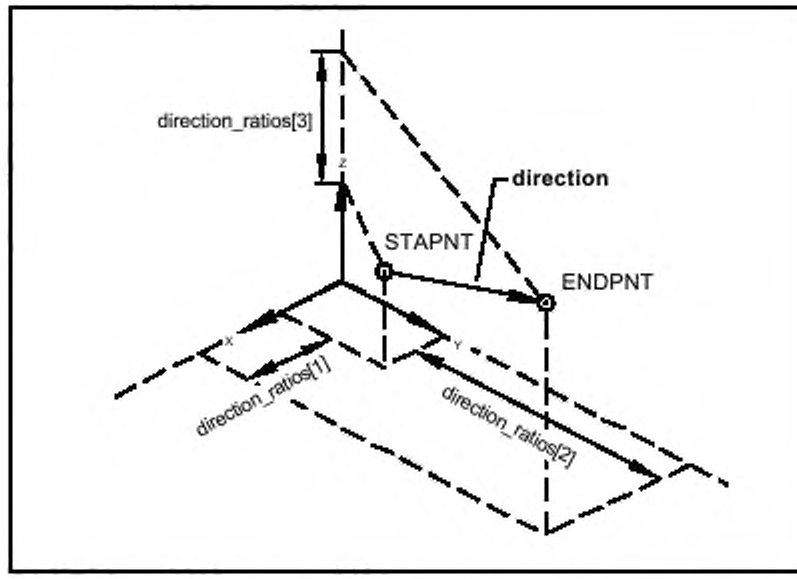
Функция создает сущность направления *direction* компонентами сущности *direction_ratio* вдоль ортогональных направлений *x*, *y* и *z*.

Величины *P1* и *P2* рассматриваются как синонимы двух заданных декартовых точек *cartesian_point*: начальной точки STAPNT и конечной точки ENDPNT соответственно.

Вычисляется разность *P2* — *P1*. Полученные три производные величины хранятся как компоненты направления для осей *X*, *Y* и *Z*. Функция возвращает имя созданного направления. Стиль полученного направления равен *null_style*.

Расстояние между двумя указанными декартовыми точками должно лежать в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то значение компоненты *Z* игнорируется интерфейсом.



Direction_ratios — компонента направления; direction — направление; STAPNT — начальная точка; ENDPNT — конечная точка

Рисунок А.1 — Функция Dir_2_Pnt

Внутренние ссылки: 6.1.9, 6.1.9.3, 6.2.1.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
103	Расстояние между двумя точками лежит вне установленного диапазона [EPS, MAX]	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

А.5.1.1.3 Задание вектора направления двумя направлениями и углом

Имя функции:	Уровень интерфейса:	1
Dir_2_Dir_Angle	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	REFDIR	N	Имя ссылочного направления *direction*	dir
Ввод	ZDIR	N	Имя направления *z*	dir
Ввод	ANGLE	D	Значение угла	(0° ≤ ANGLE ≤ 360°)
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданного направления *direction*	dir

Привязка языка FORTRAN:
NAME = DIR_2_DIR_ANGLE (REFDIR, ZDIR, ANGLE, KFIX)

Результат использования функции

Функция создает нормированную сущность *direction* с помощью сущности *direction_ratio* по производным компонентам направлений *x*, *y* и *z*, вычисленным для текущей инициализации вида.

Создается поименованная нормированная сущность *direction*. Параметру ANGLE присваивается значение ориентированного угла из REFDIR.

Открытый 2D-вид отсутствует.

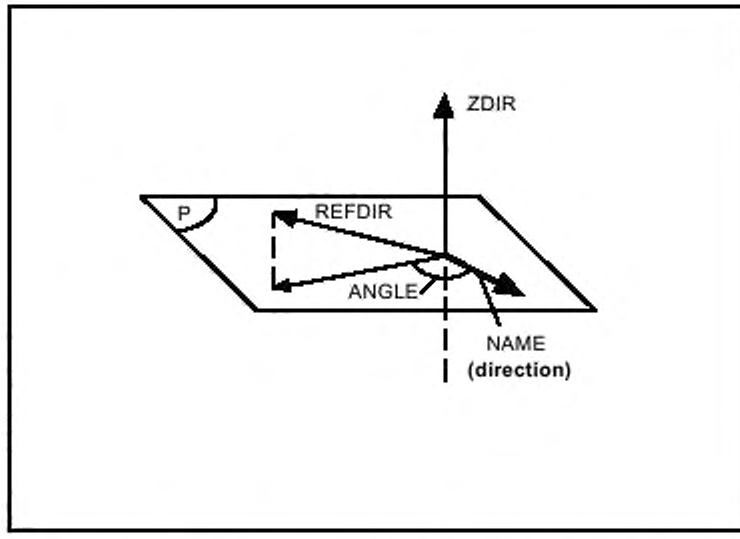
В случае открытого 3D-вида:

- пусть *P* — плоскость, перпендикулярная направлению ZDIR. Создается нормированная поименованная сущность *direction*. При этом значение ориентированного угла ортогональной проекции REFDIR на плоскость *P* равно ANGLE.

В обоих случаях функция возвращает имя созданного направления *direction*. Полученное направление имеет нулевой стиль *null_style*.

Если заданное значение угла ANGLE, измеренное в текущей координатной системе *OVC_angle_unit*, меньше 0 *ZERO_value*, то имя созданной сущности *direction* — REFDIR. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то заданный параметр ZDIR игнорируется интерфейсом.



ZDIR — направление оси Z; REFDIR — ссылочное направление; ANGLE — угол; NAME (direction) — поименованное направление

Рисунок А.2 — Функция Dir_2_Dir_Angle

Внутренние ссылки: 6.1.9, 6.1.9.3, 6.2.1.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
4	Значение плоского угла находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
117	Заданные направления параллельны	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

А.5.1.1.4 Задание оси X сущностью axis2_placement

Имя функции:	Уровень интерфейса:	1
Dir_A2p_X	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	A2PNAM	N	Имя сущности *axis2_placement*	a2p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданного направления *direction*	dir

Привязка языка FORTRAN:
NAME = DIR_A2P_X (A2PNAM, KFIX)

Результат использования функции

Функция создает нормированную сущность направления *direction*, коллинеарную оси (Ox), заданной сущностью *axis2_placement*. Функция возвращает имя созданной сущности *direction*. Полученное направление имеет нулевой стиль *null_style*.

При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Нет.

Внутренние ссылки: 6.1.9.3, 6.1.9.7, 6.1.9.8.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.1.1.5 Задание оси Y сущностью axis2_placement

Имя функции:	Уровень интерфейса:	1
Dir_A2p_Y	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	A2PNAM	N	Имя сущности *axis2_placement*	a2p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданного направления *direction*	dir

Привязка языка FORTRAN:
NAME = DIR_A2P_Y (A2PNAM, KFIX)

Результат использования функции

Функция создает нормированную сущность направления *direction*, коллинеарную оси (Oy), заданной сущностью *axis2_placement*. Функция возвращает имя созданной сущности *direction*. Полученное направление имеет нулевой стиль *null_style*.

При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Нет.

Внутренние ссылки: 6.1.9.3, 6.1.9.7, 6.1.9.8.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.1.1.6 Задание оси Z сущностью axis2_placement

Имя функции:	Уровень интерфейса:	2
Dir_A2p_Z	Уровень геометрической мощности:	2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	A2PNAM	N	Имя сущности *axis2_placement*	a2p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *direction*	dir

Привязка языка FORTRAN:
NAME = DIR_A2P_Z (A2PNAM, KFIX)

Результат использования функции

Функция создает нормированную сущность направления *direction*, коллинеарную оси (Oz), заданной сущностью *axis2_placement*. Функция возвращает имя созданной сущности *direction*. Полученное направление имеет нулевой стиль *null_style*.

При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то возникает ошибка, значение имени возвращаемой сущности равно 0.

Внутренние ссылки: 6.1.9.3, 6.1.9.8.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с уровнем используемого интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.1.2 Сущность axis1_placement (одна ось)

Генерация axis1_placement	A1p_Gen
Построение axis1_placement по двум точкам	A1p_2_Pnt

A.5.1.2.1 Генерация axis1_placement

Имя функции:	Уровень интерфейса:	2
A1p_Gen	Уровень геометрической мощности:	2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PNTNAM	N	Имя декартовой точки *cartesian_point*	pnt
Ввод	DIRNAM	N	Имя направления *direction*	dir
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Ввод	NAME	N	Имя созданной сущности *axis1_placement*	a1p

Привязка языка FORTRAN:
NAME = A1P_GEN (PNTNAM, DIRNAM, KFIX)

Результат использования функции

С помощью сущности *axis1_placement* создается одна координатная ось в 3D-пространстве.

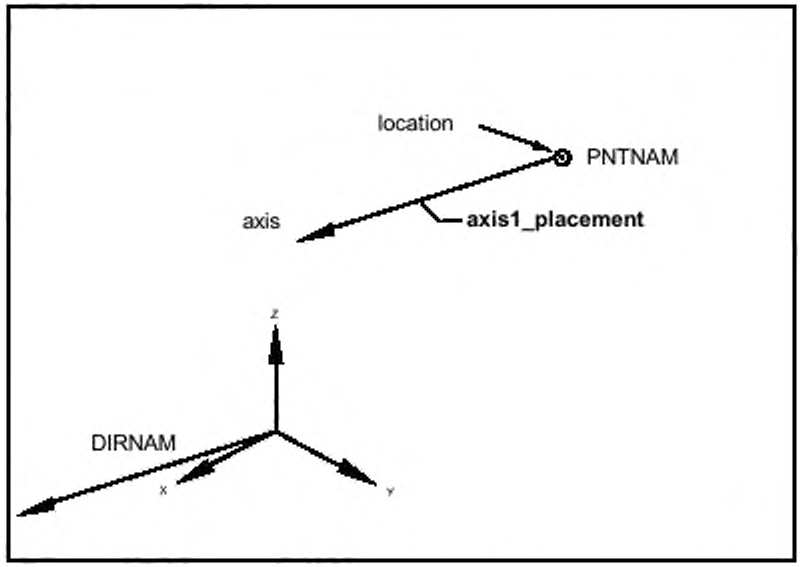
Заданная декартова точка *cartesian_point* с именем PNTNAM дублируется как точка *p1*. Данная точка имеет нулевой стиль *null_style*.

Заданное направление *direction* с именем DIRNAM дублируется как направление *d*. Данное направление имеет нулевой стиль.

Создается экземпляр сущности *axis1_placement*, расположенной в точке *p1* вдоль направления *d*. Функция возвращает имя полученной сущности. Стиль полученной сущности — нулевой.

Все входные параметры являются обязательными. При возникновении ошибки имя сущности равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то возникает ошибка, значение имени сущности равно 0.



Location — размещение заданной точки; PNTNAM — опорная точка; axis — ось, axis1_placement — сущность axis1_placement; DIRNAM — заданное направление

Рисунок А.3 — Функция A1p_Gen

Внутренние ссылки: 6.1.9, 6.1.9.6, 6.2.1.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с применяемым уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

А.5.1.2.2 Построение сущности axis1_placement по двум точкам

Имя функции:	Уровень интерфейса:	2
A1p_2_Pnt	Уровень геометрической мощности:	2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	STAPNT	N	Имя начальной точки *cartesian_point*	pnt
Ввод	ENDPNT	N	Имя конечной точки *cartesian_point*	pnt
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *axis1_placement*	a1p

Привязка языка FORTRAN:
NAME = A1P_2_PNT (STAPNT, ENDPNT, KFIX)

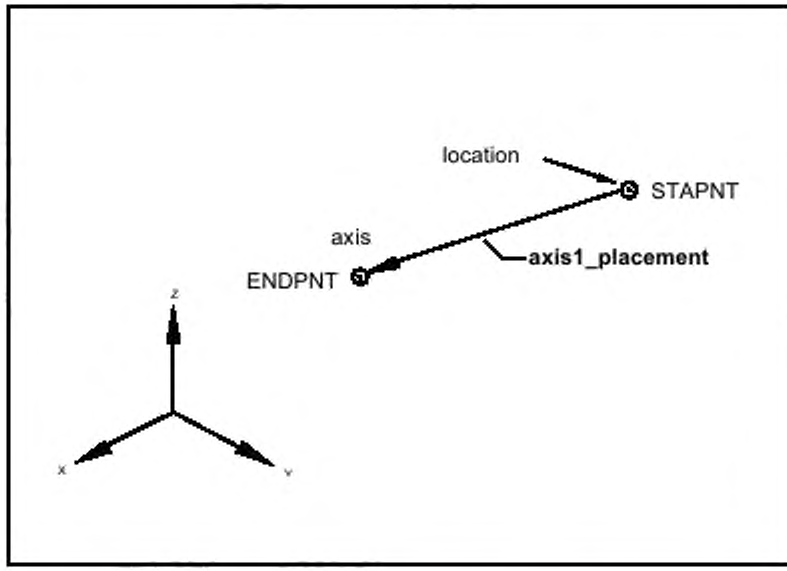
Результат использования функции

Функция создает сущность *axis1_placement*, которая задает координатную ось в 3D-пространстве. Заданная декартова точка *cartesian_point* с именем STAPNT дублируется как точка *p1*, имеющая нулевой стиль *null_style*. Пусть *P2* является синонимом второй заданной точки cartesian_point ENDPNT. Тогда расстояние между двумя декартовыми точками cartesian_point должно лежать в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Создается экземпляр *d* направления *direction* с компонентами *direction_ratio*, полученными по точкам *P2* — *p1*. Полученное направление имеет нулевой стиль.

Создается экземпляр сущности *axis1_placement* по расположению заданной точки *p1* и оси *d*. Функция возвращает имя сущности axis1_placement, ее стиль — нулевой.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то возникает ошибка и значение имени сущности равно 0.



Location — размещение заданной точки; STAPNT — начальная точка, axis — ось; axis1_placement — сущность axis1_placement; ENDPNT — конечная точка

Рисунок А.4 — Функция A1 p_Pnt

Внутренние ссылки: 6.1.9, 6.1.9.6, 6.2.1.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
103	Расстояние между двумя точками лежит вне установленного диапазона [EPS, MAX]	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	203	Функция несовместима с применением уровня интерфейса
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.1.3 Сущность axis2_placement (локальная координатная система)

Построение сущности *axis2_placement* по трем точкам	A2p_3_Pnt
Построение сущности *axis2_placement* по двум направлениям	A2p_2_Dir
Построение сущности *axis2_placement* по двум направлениям (Ox) и (Oy)	A2p_2_Dir_Xy
Построение сущности *axis2_placement* путем относительного позиционирования	A2p_Position_Relative
Построение сущности *axis2_placement* с помощью ссылочной координатной системы	A2p_Ref_Sys

A.5.1.3.1 Построение сущности axis2_placement по трем точкам

Имя функции:

A2p_3_Pnt

Уровень интерфейса:	1
Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	CENPNT	N	Имя начальной декартовой точки *cartesian_point*	pnt
Ввод	AXSPNT	N	Имя декартовой точки *cartesian_point*, отложенной в направлении оси *Z* (игнорируется для 2D-вида)	pnt
Ввод	REFPNT	N	Имя декартовой точки *cartesian_point*, отложенной либо в направлении аппроксимации оси *X*, либо в точном направлении оси *X* в случае 2D-вида	pnt
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *axis2_placement*	a2p

Привязка языка FORTRAN:
NAME = A2P_3_PNT (CENPNT, AXSPNT, REFPNT, KFIX)

Результат использования функции

Функция создает сущность *axis2_placement*, которая представляет собой ортогональную локальную координатную систему (LCS) в текущей ссылочной базовой координатной системе OVC. Тип созданной сущности *axis2_placement* зависит от инициализации открытого вида, то есть создается экземпляр сущности *axis2_placement_2d* в случае использования 2D-вида или создается экземпляр сущности *axis2_placement_3d* в случае использования 3D-вида. Для создания сущности *axis2_placement_3d* используются три точки CENPNT, AXPNT и REFPNT для получения начала координат (O) и две оси (Oz и Ox) локальной координатной системы. Для создания сущности *axis2_placement_2d* используются только две из трех заданных точек (CENPNT и REFPNT). Их достаточно для создания начала координат (O) и оси (Ox) локальной координатной системы.

Заданная декартова точка *cartesian_point* с именем CENPNT дублируется как точка *p1*. Данная декартова точка используется для определения начала LCS, ее стиль — нулевой *null_style*.

Если создается экземпляр сущности *axis2_placement_3d*:

- пусть точки *P2* и *P3* являются синонимами двух заданных декартовых точек AXSPNT и REFPNT соответственно;

- для направления *direction d1* создается экземпляр с компонентами *direction_ratio*, определенными точками *P2 — p1*. Указанное направление используется для определения точного направления локальной оси *Z*. Полученное направление имеет нулевой стиль. При этом расстояние между двумя указанными декартовыми точками должно находиться в диапазоне [EPS, MAX];

- для направления *direction d2* создается экземпляр с компонентами *direction_ratio*, определенными точками *P3 — p1*. Указанное направление используется для определения аппроксимации направления оси *X*. Полученное направление имеет нулевой стиль. При этом расстояние между двумя декартовыми точками должно находиться в диапазоне [EPS, MAX];

- для сущности *axis2_placement_3d* создается экземпляр *p1* по точке отсчета *location* и экземпляр оси *d1* по базовому направлению *d2* с помощью сущности *ref_direction*. Стиль данной сущности *axis2_placement_3d* — нулевой. Функция возвращает имя полученной сущности *axis2_placement_3d*.

В случае создания экземпляра *axis2_placement_2d*:

- пусть *P3* является синонимом одной заданной декартовой точки REFPNT;

- создается экземпляр *d2* сущности *direction* с компонентами *direction_ratio*, определенными точками *P3 — p1*. Указанное направление используется для определения точного направления локальной оси *X*. Полученное направление имеет нулевой стиль. При этом расстояние между двумя указанными декартовыми точками должно находиться в диапазоне [EPS, MAX];

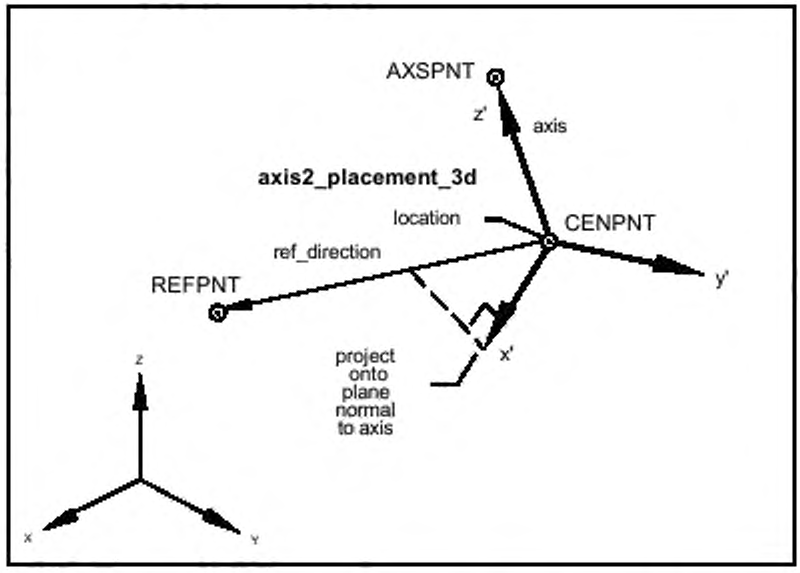
- для сущности *axis2_placement_2d* создается экземпляр *p1* по началу координат *location* и базовому направлению *ref_direction d2*. Указанная сущность *axis2_placement_2d* имеет нулевой стиль. Функция возвращает имя полученной сущности *axis2_placement_2d*.

При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечания

1 При необходимости выполняется настройка базового направления *ref_direction* для обеспечения его ортогональности направлению оси *Y*. Настройка выполняется путем проектирования базового направления *ref_direction* на плоскость, перпендикулярную оси *axis*.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то заданный параметр AXSPNT игнорируется интерфейсом.



AXSPNT — точка на оси Z; axis — ось; axis2_placement_3d — сущность axis2_placement_3d, location — начало координат; CENPNT — точка отсчета; ref_direction — ссылочное направление, REFPNT — базовая точка; project onto plane normal to axis — проекция на плоскость, перпендикулярную оси

Рисунок — А.5 Функция A2p_3_Pnt

Внутренние ссылки: 6.1.9, 6.1.9.7, 6.1.9.8, 6.2.1.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
103	Расстояние между двумя точками не входит в установленный диапазон [EPS, MAX]	105	Попытка создания вырожденного направления в процессе создания сущности
116	Заданные точки линейно зависимы	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.1.3.2 Построение сущности axis2_placement по двум направлениям

Имя функции:	Уровень интерфейса:	1
A2p_2_Dir	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	CENPNT	N	Имя декартовой точки *cartesian_point*, определяющей начало координат	pnt
Ввод	AXSDIR	N	Имя направления оси *Z* (игнорируется для 2D-вида)	dir
Ввод	REFDIR	N	Имя аппроксимации направления оси *X* либо точного направления оси *X* в случае 2D-вида	dir
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *axis2_placement*	a2p

Привязка языка FORTRAN:
NAME = A2P_2_DIR (CENPNT, AXSDIR, REFDIR, KFIX)

Результат использования функции

Функция создает сущность *axis2_placement*, которая является ортогональной локальной координатной системой (LCS) в текущей базовой координатной системе OVC. Тип созданной сущности *axis2_placement* зависит от инициализации открытого вида, то есть в случае 2D-вида создается экземпляр сущности *axis2_placement_2d*, а в случае 3D-вида создается экземпляр *axis2_placement_3d*. При создании сущности *axis2_placement_3d* три заданных параметра CENPNT, AXDIR и REFDIR используются для создания начала координат (O) и двух осей (Oz и Ox) локальной координатной системы. При создании сущности *axis2_placement_2d* только два из трех указанных параметров (CENPNT и REFDIR) используются для создания начала координат (O) и оси (Ox) локальной координатной системы.

Заданная декартова точка CENPNT дублируется как точка *p1*. Данная точка используется для определения локального начала координат и имеет нулевой стиль *null_style*. Затем:

- два направления *direction*, AXSDIR и REFDIR, дублируются как направления *d1* и *d2* соответственно. Указанные направления определяют точное направление локальной оси *Z* и аппроксимацию направления локальной оси *X*. Направления имеют нулевой стиль.

В случае создания экземпляра *axis2_placement_3d*:

- сущность *axis2_placement_3d* создает экземпляр *p1*с началом координат *location*, осью *d1* и базовым направлением *ref_direction d2*. Функция возвращает имя сущности *axis2_placement_3d*. Сущность имеет нулевой стиль.

Направление REFDIR дублируется как направление *d2*, имеющее нулевой стиль. Указанное направление используется для определения точного направления локальной оси *X*.

В случае создания экземпляра *axis2_placement_2d*:

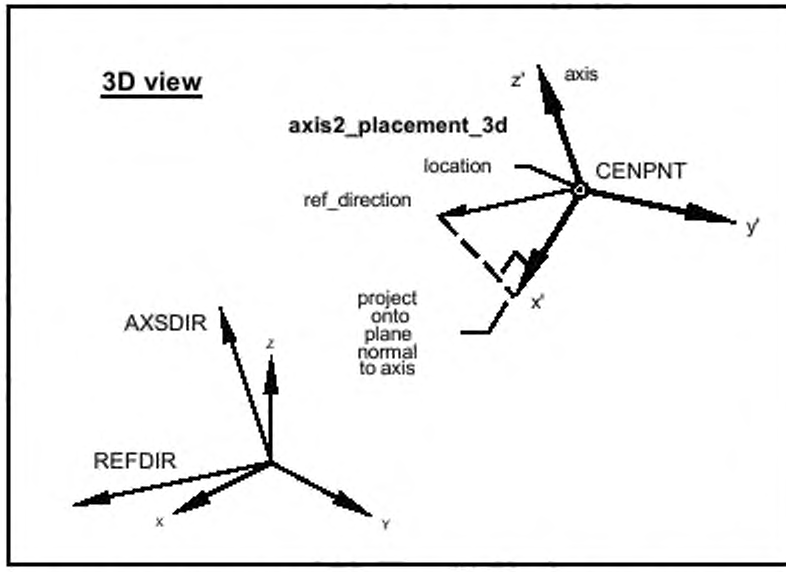
- создается экземпляр сущности *axis2_placement_2d* с началом координат *location p1* и базовым направлением *ref_direction d2*. Функция возвращает имя сущности *axis2_placement_2d*, имеющей нулевой стиль.

При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечания

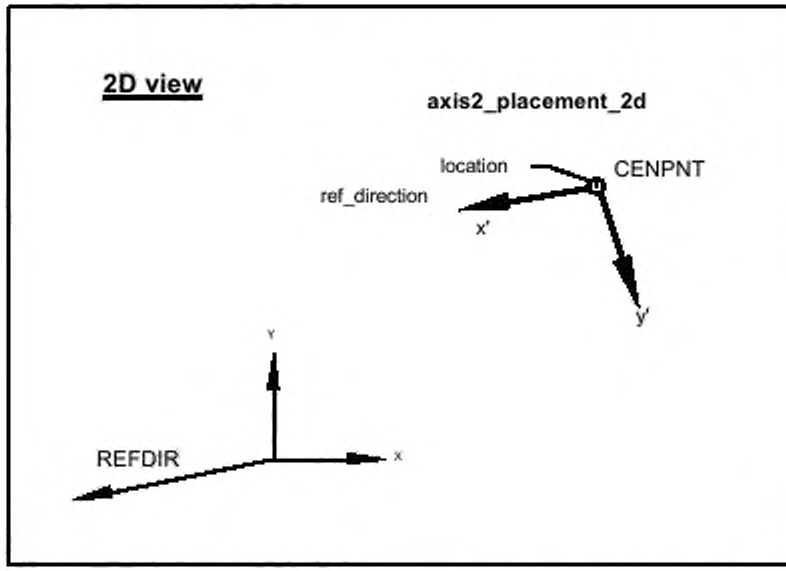
1 При необходимости производится настройка базового направления *ref_direction* для обеспечения его ортогональности направлению оси. Настройка производится путем проектирования направления *ref_direction* на плоскость, перпендикулярную оси *axis*.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то параметр AXSDIR игнорируется интерфейсом.



Axis — ось; 3D view — 3D-вид; axis2_placement_3d — сущность axis2_placement_3d, location — размещение, CENPNT — начало координат; ref_direction — ссылочное направление; project onto plane normal to axis — проекция на плоскость, перпендикулярную оси, AXSDIR — вектор направления оси; REFDIR — вектор базового направления

Рисунок А.6 — Функция A2p_2_Dir (3D-вид)



2D view — 2D-вид; axis2_placement_2d — сущность axis2_placement_2d: location — размещение: CENPNT — начало координат; ref_direction — ссылочное направление; REFDIR — вектор базового направления

Рисунок А.7 — Функция A2p_2_Dir (2D-вид)

Внутренние ссылки: 6.1.9, 6.1.9.7, 6.1.9.8, 6.2.1.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
101	Попытка создания вырожденной сущности	117	Заданные направления параллельны
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.1.3.3 Построение сущности axis2_placement по двум направлениям (Ox) и (Oy)

Имя функции:	Уровень интерфейса:	1
A2p_2_Dir_Xy	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	CENPNT	N	Имя декартовой точки *cartesian_point*, определяющей начало координат	pnt
Ввод	REFDIR	N	Имя точного направления оси *X*	dir
Ввод	YAXDIR	N	Имя аппроксимации направления оси *Y* (игнорируется для 2D-вида)	dir
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *axis2_placement*	a2p

Привязка языка FORTRAN:
NAME = A2P_2_DIR_XY (CENPNT, REFDIR, YAXDIR, KFIX)

Результат использования функции

Функция создает сущность *axis2_placement*, которая является ортогональной локальной координатной системой (LCS) в текущей базовой координатной системе OVC. Тип созданной сущности зависит от инициализации открытого вида, то есть создается либо экземпляр сущности *axis2_placement_2d* в случае 2D-вида, либо экземпляр сущности *axis2_placement_3d* в случае 3D-вида. При создании сущности *axis2_placement_3d* три заданных параметра CENPNT, REFDIR и YAXDIR используются для создания начала координат (O) и двух осей (Ox и Oy) локальной координатной системы. При создании сущности *axis2_placement_2d* только два из трех заданных параметров (CENPNT и REFDIR) используются для создания начала координат (O) и оси (Ox) локальной координатной системы.

Заданная декартова точка *cartesian_point* CENPNT дублируется как точка *p1*, используемая для определения положения начала координат локальной системы. Заданное направление *direction*, REFDIR, дублируется как направление *d1*, используемое для определения точного направления локальной оси *X*. Две указанные сущности имеют нулевой стиль *null_style*.

В случае создания экземпляра *axis2_placement_3d*:

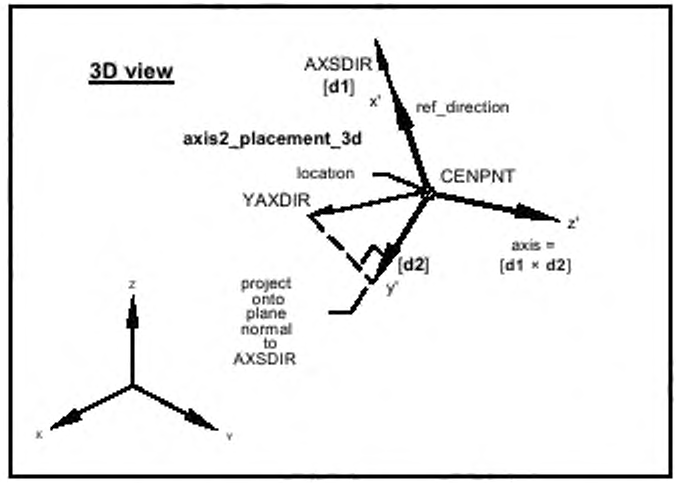
- создается направление *direction d2* путем вычисления проекции нормированного направления YAXDIR на плоскость, перпендикулярную направлению *d1*. Указанное направление имеет нулевой стиль *null_style*;
- создается экземпляр *d3* направления *direction*. Его атрибуты определены векторным произведением направлений *d1* и *d2*. Указанное направление задает точное направление локальной оси *Z*, имеющее нулевой стиль;
- создается экземпляр сущности *axis2_placement_3d* с началом координат *location p1*, осью *d3* и базовым направлением *ref_direction d1*. Функция возвращает имя созданной сущности *axis2_placement_3d*, имеющей нулевой стиль.

В случае создания экземпляра сущности *axis2_placement_2d*:

- создается экземпляр сущности *axis2_placement_2d* с началом координат *location p1* и базовым направлением *ref_direction d1*. Функция возвращает имя созданной сущности *axis2_placement_2d*, имеющей нулевой стиль.

При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то заданный параметр YAXDIR игнорируется интерфейсом.



AXSDIR — вектор направления оси; 3D view — 3D-вид; ref_direction — ссылочное направление, axis2_placement_3d — сущность axis2_placement_3d; location — размещение; CENPNT — начало координат; YAXDIR — имя аппроксимированного направления оси Y: axis = [d1 × d2] — ось [d1 × d2], project onto plane normal to AXSDIR — проекция на плоскость, перпендикулярную вектору направления оси

Рисунок А.8 — Функция A2p_2_Dir_Xy

Внутренние ссылки: 6.1.9, 6.1.9.7, 6.1.9.8, 6.2.1.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
101	Попытка создания вырожденной сущности	117	Заданные направления параллельны
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.1.3.4 Построение сущности axis2_placement путем относительного позиционирования

Имя функции:	Уровень интерфейса:	1
A2p_Position_Relative	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/ значение
Ввод	REFLST	6 × D	Перечень из 6 пунктов, содержащий описания последовательных поворотов и смещений	
			1) поворот в плоскости (Oxy) вокруг оси *Z* в ссылочной координатной системе OVC;	(–360° ≤ REFLST(1) ≤ 360°)
			2) поворот в плоскости (Ozy) вокруг оси *X* в базовой системе координат OVC;	(–360° ≤ REFLST(2) ≤ 360°)
			3) поворот в плоскости (Ozx) вокруг оси *Y* базовой координатной системы OVC;	(–360° ≤ REFLST(3) ≤ 360°)
			4) смещение вдоль оси (Ox) базовой координатной системы OVC;	(0.0 или (EPS ≤ |REFLST(4)| ≤ MAX))
			5) смещение вдоль оси (Oy) базовой координатной системы OVC;	(0.0 или (EPS ≤ |REFLST(5)| ≤ MAX))
			6) смещение вдоль оси (Oz) базовой координатной системы OVC	(0.0 или (EPS ≤ |REFLST(6)| ≤ MAX))
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности axis2_placement	a2p

Привязка языка FORTRAN:
NAME = A2P_POSITION_RELATIVE (REFLST, KFIX)

Результат использования функции

Функция создает сущность *axis2_placement*, которая является локальной координатной системой (LCS) в базовой координатной системе OVC. Тип созданной сущности *axis2_placement* зависит от инициализации открытого вида, то есть создается экземпляр сущности *axis2_placement_2d* в случае 2D-вида или экземпляр сущности *axis2_placement_3d* в случае 3D-вида.

В случае создания экземпляра сущности *axis2_placement_3d*:

- создается экземпляр сущности *axis2_placement_3d* как копия базовой координатной системы OVC. Все неявно созданные экземпляры сущности *axis2_placement_3d* и сама указанная сущность имеют нулевой стиль *null_style*;

- матрицы преобразования, содержащиеся внутри заданного параметра REFLIST, применяются к новой сущности *axis2_placement_3d* в следующей последовательности:

1) вращение вокруг оси *Z* базовой координатной системы OVC;

2) вращение вокруг оси *X* базовой координатной системы OVC;

3) вращение вокруг оси *Y* базовой координатной системы OVC;

4) смещение начала координат новой сущности *axis2_placement_3d* вдоль осей *X*, *Y* и *Z* базовой координатной системы OVC;

- функция возвращает имя созданной сущности *axis2_placement_3d*, имеющей нулевой стиль.

В случае создания экземпляра сущности *axis2_placement_2d*:

- создается экземпляр сущности *axis2_placement_2d* как копия базовой координатной системы OVC;

- матрицы преобразования, содержащиеся внутри заданного параметра REFLIST, применяются к новой сущности *axis2_placement_2d* в следующей последовательности:

1) вращение в плоскости (Oxy) базовой координатной системы OVC;

2) смещение начала координат новой сущности *axis2_placement_2d* вдоль осей *X* и *Y* базовой координатной системы OVC;

- функция возвращает имя полученной сущности *axis2_placement_2d*.

Значения углов измеряются в единицах угла *OVC_angle_unit*, смещения измеряются в единицах длины *OVC_length_unit*. Указанные значения либо должны быть равны 0, либо должны находиться в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечания

1 Если вычисленная евклидова норма смещения лежит в диапазоне [*ZERO_value*, EPS], то смещение не производится, ошибок нет.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то значения переменных REFLST(2, 3) поворотов относительно осей *X* и *Y* и значение переменой REFLST(6) для смещения вдоль оси *Z* игнорируются интерфейсом.

Внутренние ссылки: 5.1.3, 6.1.9, 6.1.9.7, 6.1.9.8, 6.2.1.2.

Ошибки

3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
101	Попытка создания вырожденной сущности	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.1.3.5 Построение сущности axis2_placement с помощью ссылочной координатной системы

Имя функции:	Уровень интерфейса:	1
A2P_REF_SYS	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *axis2_placement*	a2p

Привязка языка FORTRAN:
NAME = A2P_REF_SYS (KFIX)

Результат использования функции

Функция создает сущность *axis2_placement*, которая является локальной координатной системой (LCS) (в базовой координатной системе OVC) с расположением и ориентацией, идентичной расположению и ориентации базовой координатной системы OVC.

Тип созданной сущности *axis2_placement* зависит от инициализации открытого вида, то есть создается экземпляр сущности *axis2_placement_2d* в случае 2D-вида либо экземпляр сущности *axis2_placement_3d* в случае 3D-вида.

В случае создания 2D-вида:

- создается экземпляр сущности *axis2_placement_2d* с расположением, осью и базовым направлением *ref_direction*, определяемыми относительно текущей базовой координатной системы. Функция возвращает имя сущности *axis2_placement_2d*, имеющей нулевой стиль *null_style*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

В случае создания 3D-вида:

- создается экземпляр сущности *axis2_placement_3d* с расположением и базовым направлением *ref_direction*, определенными относительно базовой координатной системы OVC. Функция возвращает имя сущности *axis2_placement_3d*, имеющей нулевой стиль *null_style*.

Примечание — Нет.

Внутренние ссылки: 5.3.1, 6.1.9.7, 6.1.9.8, 6.2.1.2.

Ошибки

201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.2 Сущности point

А.5.2.1 Точки с каноническим определением

Точки с абсолютными декартовыми координатами	Pnt_Cartesian_Absolute
Точки с относительными декартовыми координатами	Pnt_Cartesian_Relative
Точки с абсолютными полярными координатами	Pnt_Polar_Absolute
Точки с относительными полярными координатами	Pnt_Polar_Relative
Точки с абсолютными цилиндрическими координатами	Pnt_Cylinder_Absolute
Точки с относительными цилиндрическими координатами	Pnt_Cylinder_Relative

А.5.2.1.1 Точки с абсолютными декартовыми координатами

Имя функции:	Уровень интерфейса:	1
Pnt_Cartesian_Absolute	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	X	D	*X*-координата декартовой точки *cartesian_point*	(0.0 или (EPS ≤ |X| ≤ MAX))
Ввод	Y	D	*Y*-координата декартовой точки *cartesian_point*	(0.0 или (EPS ≤ |Y| ≤ MAX))
Ввод	Z	D	*Z*-координата декартовой точки *cartesian_point*	(0.0 или (EPS ≤ |Z| ≤ MAX))
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_CARTESIAN_ABSOLUTE (X, Y, Z, KFIX)

Результат использования функции

Функция создает сущность точки *cartesian_point* с декартовыми координатами *X*, *Y* и *Z*. Указанные координаты вычисляются по заданным параметрам *X*, *Y* и *Z* соответственно (это декартовы координаты указанной точки в базовой координатной системе OVC). Настоящая сущность имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*.

Заданные координаты измерены в единицах длины *OVC_length_unit*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то заданный параметр *Z* игнорируется интерфейсом.

Внутренние ссылки: 6.1.9.2, 6.2.4, 8.2.

Ошибки

3	Значение меры длины находится вне допустимого диапазона	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.2.1.2 Точки с относительными декартовыми координатами

Имя функции:	Уровень интерфейса:	1
Pnt_Cartesian_Relative	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PNTNAM	N	Имя ссылочной декартовой точки *cartesian_point*	pnt
Ввод	DX	D	Координата *X* декартовой точки *cartesian_point*	(0.0 или (EPS ≤ |DX| ≤ MAX))
Ввод	DY	D	Координата *Y* декартовой точки *cartesian_point*	(0.0 или (EPS ≤ |DY| ≤ MAX))
Ввод	DZ	D	Координата *Z* декартовой точки *cartesian_point*	(0.0 или (EPS ≤ |DZ| ≤ MAX))
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_CARTESIAN_RELATIVE (PNTNAM, DX, DY, DZ, KFIX)

Результат использования функции

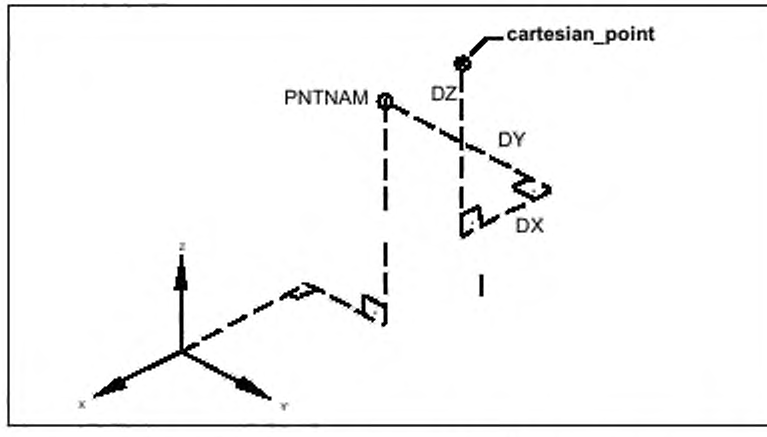
Функция создает сущность *cartesian_point* декартовой точки с координатами *X*, *Y* и *Z*. Эти координаты вычисляются по параметрам *DX*, *DY* и *DZ* (приращения декартовых координат в базовой координатной системе OVC), заданным относительно ссылочной декартовой точки *cartesian_point* с именем PNTNAM. Настоящая сущность *cartesian_point* имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*.

Заданные координаты измеряются в единицах длины *OVC_length_unit*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечания

1 Если вычисленное расстояние между исходной точкой PNTNAM и новой созданной декартовой точкой *cartesian_point* лежит в диапазоне [*ZERO_value*, EPS], то копии координат точки PNTNAM можно использовать для создания новой декартовой точки.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то заданный параметр *DZ* игнорируется интерфейсом.



Cartesian_point — результирующая точка; PNTNAM — исходная точка

Рисунок А.9 — Функция Pnt_Cartesian_Relative

Внутренние ссылки: 6.1.9.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.2.1.3 Точки с абсолютными полярными координатами

Имя функции:

Pnt_Polar_Absolute

Уровень интерфейса:	1
Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PHI	D	Угол в плоскости (Oxy) относительно оси (Ox) текущей OVC	(–360° ≤ PHI ≤ 360°)
Ввод	THETA	D	Угол между осью (Oz) и плоскостью (Oxy) текущей OVC	(–360° ≤ THETA ≤ 360°)
Ввод	RAD	D	Расстояние декартовой точки *cartesian_point* от начала координат	(0.0 или (EPS ≤ RAD ≤ MAX))
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_POLAR_ABSOLUTE (PHI, THETA, RAD, KFIX)

Результат использования функции

Функция создает сущность декартовой точки *cartesian_point* с координатами *X*, *Y* и *Z*. Указанные полярные координаты определяются параметрами PHI, THETA и RAD (координатами точки в базовой координатной системе OVC). Данная декартова точка имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*.

Заданные углы вычисляются в единицах угла *OVC_angle_unit*, расстояния измеряются в единицах длины *OVC_length_unit*. Измеренные величины равны 0 либо лежат в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* таблицы статуса интерфейса равно 1), то заданный параметр THETA, неявно определяющий значение *Z*, игнорируется интерфейсом.

Внутренние ссылки: 6.1.9.2, 6.2.4, 8.2.

Ошибки

3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.2.1.4 Точки с относительными полярными координатами

Имя функции:	Уровень интерфейса:	1
Pnt_Polar_Relative	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PNTNAM	N	Имя ссылочной декартовой точки *cartesian_point*	pnt
Ввод	PHI	D	Угол в плоскости (Oxy) относительно оси (Ox) виртуальной базовой координатной системы	(–360° ≤ PHI ≤ 360°)
Ввод	THETA	D	Угол между осью (Oz) и плоскостью (Oxy) виртуальной базовой координатной системы	(–360° ≤ THETA ≤ 360°)
Ввод	RAD	D	Расстояние от декартовой точки *cartesian_point* до начала координат	(0.0 или (EPS ≤ RAD ≤ MAX))
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_POLAR_RELATIVE (PNTNAM, PHI, THETA, RAD, KFIX)

Результат использования функции

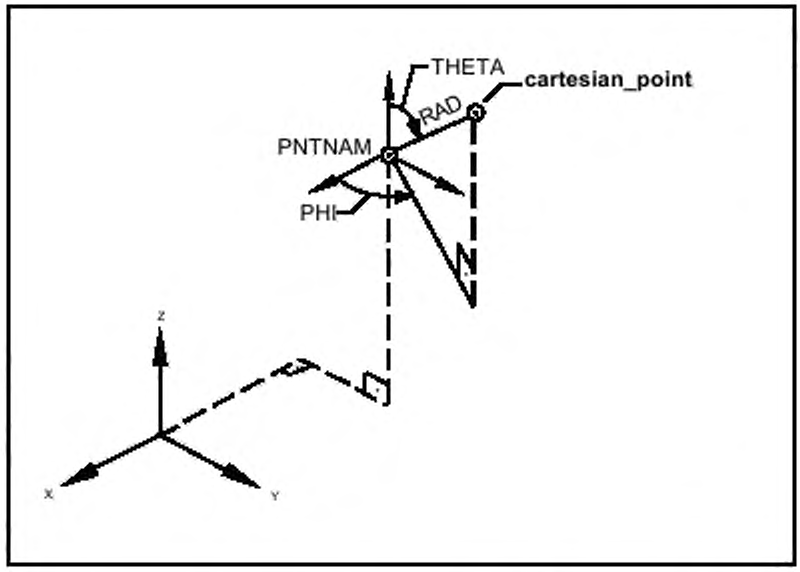
Функция создает сущность *cartesian_point* декартовой точки с координатами *X*, *Y* и *Z*. Данные координаты вычисляются по полярным координатам, являющимся производными параметров PHI, THETA и RAD (координат в базовой координатной системе OVC) относительно заданной ссылочной декартовой точки *cartesian_point* PNTNAM. Указанная декартова точка имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*.

Заданные углы вычисляются в единицах угла *OVC_angle_unit*, а расстояния измеряются в единицах длины *OVC_length_unit*. Указанные величины равны 0 либо лежат в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечания

1 Если вычисляемое расстояние между исходной точкой PNTNAM и новой созданной декартовой точкой *cartesian_point* лежит в диапазоне [*ZERO_value*, EPS], то копии координат точки PNTNAM используются для создания новой сущности *cartesian_point*.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то заданный параметр THETA, неявно определяющий значение *Z*, игнорируется интерфейсом.



THETA — угол «тета»; cartesian_point — результирующая декартова точка; RAD — расстояние от исходной точки до результирующей декартовой точки; PNTNAM — исходная точка; PHI — угол «фи»

Рисунок А.10 — Функция Pnt_Polar_Relative

Внутренние ссылки: 6.1.9.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.2.1.5 Точки с абсолютными цилиндрическими координатами

Имя функции:	Уровень интерфейса:	2
Pnt_Cylinder_Absolute	Уровень геометрической мощности:	2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PHI	D	Угол поворота в плоскости (Oxy) относительно оси (Ox) в текущей координатной системе OVC	(–360° ≤ PHI ≤ 360°)
Ввод	RAD	D	Длина проекции на плоскость (Oxy) текущей OVC	(0.0 или (EPS ≤ RAD ≤ MAX))

Ввод	HEIGHT	D	Расстояние декартовой точки *cartesian_point* от плоскости (Oxy) текущей OVC	(EPS ≤ |HEIGHT| ≤ MAX)
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_CYLINDER_ABSOLUTE (PHI, RAD, HEIGHT, KFIX)

Результат использования функции

Функция создает сущность *cartesian_point* с координатами *X*, *Y* и *Z*. Эти координаты вычисляются по цилиндрическим координатам PHI, RAD и HEIGHT базовой координатной системы OVC. Указанная декартова точка *cartesian_point* имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*.

Заданный угол вычисляется в единицах угла *OVC_angle_unit*, параметры RAD (радиус) и HEIGHT (высота) измеряются в единицах длины *OVC_length_unit*. Длина проекции на плоскость (Oxy) (радиус RAD) равна 0 либо лежит в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то возникает ошибка, значение имени сущности равно 0.

Внутренние ссылки: 6.1.9.2, 6.2.4, 8.2.

Ошибки

3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.2.1.6 Точки с относительными цилиндрическими координатами

Имя функции:	Уровень интерфейса:	2
Pnt_Cylinder_Relative	Уровень геометрической мощности:	2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PNTNAM	N	Имя ссылочной точки *cartesian_point*	pnt
Ввод	PHI	D	Поворот в плоскости (Oxy) относительно оси (Ox) виртуальной ссылочной координатной системы	(–360° ≤ PHI ≤ 360°)
Ввод	RAD	D	Длина проекции на плоскость (Oxy) виртуальной ссылочной координатной системы	(0.0 или (EPS ≤ RAD ≤ MAX))

Ввод	HEIGHT	D	Расстояние по вертикали от декартовой точки *cartesian_point* до плоскости (Oxy) виртуальной ссылочной координатной системы	(EPS ≤ |HEIGHT| ≤ MAX)
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_CYLINDER_RELATIVE (PNTNAM, PHI, RAD, HEIGHT, KFIX)

Результат использования функции

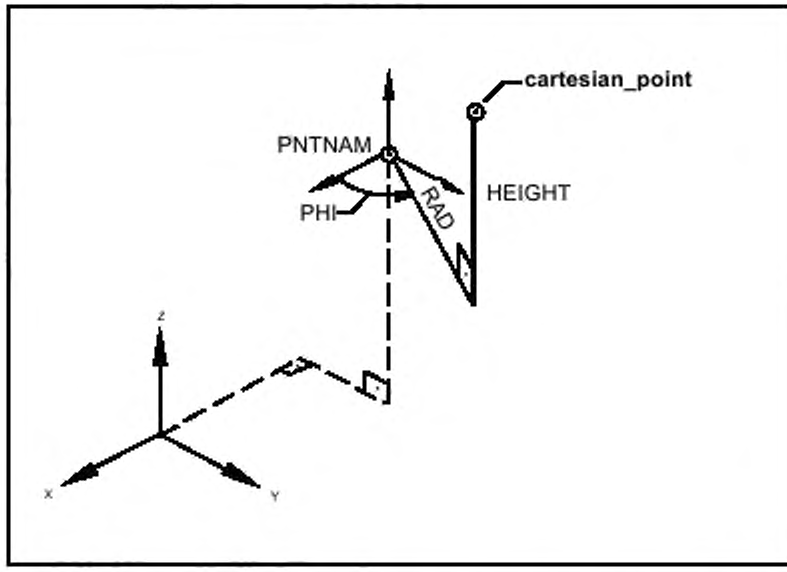
Функция создает сущность декартовой точки *cartesian_point* с координатами *X*, *Y* и *Z*. Эти координаты вычисляют по цилиндрическим координатам PHI, RAD и HEIGHT относительно заданной ссылочной декартовой точки *cartesian_point* с именем PNTNAM в базовой координатной системе OVC. Данная декартова точка имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*.

Заданный угол вычисляется в единицах угла *OVC_angle_unit*, параметры RAD (радиус) и HEIGHT (высота) измеряются в единицах длины *OVC_length_unit*. Длина проекции RAD равна 0 либо лежит в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечания

1 Если вычисленное расстояние между исходной точкой PNTNAM и новой созданной декартовой точкой *cartesian_point* лежит в диапазоне [*ZERO_value*, EPS], то для создания новой сущности *cartesian_point* используются копии координат точки PNTNAM.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то возникает ошибка, значение имени сущности равно 0.



Cartesian_point — новая декартова точка; PNTNAM — исходная декартова точка; RAD — цилиндрическая координата (радиус); HEIGHT — цилиндрическая координата (высота); PHI — цилиндрическая координата (угол «фи»)

Рисунок А.11 — Функция Pnt_Cylinder_Relative

Внутренние ссылки: 6.1.9.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.2.2 Задание точки ограничениями

Начальная точка кривой	Pnt_Begin_Ent
Конечная точка кривой	Pnt_End_Ent
Точка пересечения двух базовых сущностей	Pnt_Intersection_2_Ent
Точка касания дуги окружности	Pnt_Tangential_Arc
Точка в центре дуги окружности	Pnt_Center_Arc
Точка в середине базовой сущности	Pnt_Middle_Ent
Точка как проекция на базовую сущность	Pnt_Projection_Ent
Точка как проекция на сущность	a2pPnt_Projection_A2p

A.5.2.2.1 Начальная точка кривой

Имя функции:

Pnt_Begin_Ent

Уровень интерфейса:	1
Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	curves
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_BEGIN_ENT (ENTNAM, KFIX)

Результат использования функции

Функция создает сущность декартовой точки *cartesian_point* в начале заданной сущности кривой ENTNAM. Координаты *X*, *Y* и *Z* созданной декартовой точки вычислены либо по значению параметра (*parameter_1*), либо по ссылке на декартову точку (*point_1*) атрибута точки вычленения *trim_1* заданной базовой сущности ENTNAM или сущности *conic_arc*.

Если заданная общая сущность кривой ENTNAM является экземпляром типа полилинии *polyline*, то функция создает декартову точку с координатами *X*, *Y* и *Z*, вычисленными по первой декартовой точке перечня атрибутов точек.

Если заданная общая сущность кривой ENTNAM является экземпляром типа *api_contour*, то функция создает декартову точку с координатами *X*, *Y* и *Z*, вычисленными по первой точке первого сегмента комбинированной кривой *composite_curve_segment*, являющейся начальной точкой контура *api_contour* интерфейса прикладного программирования.

Указанная декартова точка имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то значение *Z* игнорируется интерфейсом.

Внутренние ссылки: 6.1.9.2, 6.1.12, 6.1.13, 6.1.14, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.2.2.2 Конечная точка кривой

Имя функции:	Уровень интерфейса:	1
Pnt_End_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	curves
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_END_ENT (ENTNAM, KFIX)

Результат использования функции

Функция создает сущность декартовой точки *cartesian_point* в конце заданной сущности кривой ENTNAM. Координаты *X*, *Y* и *Z* созданной декартовой точки *cartesian_point* вычисляются либо по значению параметра (*parameter_2*), либо по ссылочной декартовой точке (*point_2*) атрибута точки вычленения *trim_2*, заданной базовой сущности ENTNAM или сущности *conic_arc*.

Если заданная общая сущность кривой ENTNAM является экземпляром типа полилинии *polyline*, то функция создает декартову точку с координатами *X*, *Y* и *Z*, вычисленную по последней декартовой точке перечня атрибутов точек.

Если заданная общая сущность кривой ENTNAM является экземпляром типа контур *api_contour*, то функция создает декартову точку *cartesian_point* с координатами *X*, *Y* и *Z*, вычисленными по первой точке первого сегмента комбинированной кривой *composite_curve_segment*, которая является конечной точкой контура *api_contour* интерфейса прикладного программирования.

Полученная декартова точка имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то значение *Z* игнорируется интерфейсом.

Внутренние ссылки: 6.1.9.2, 6.1.12, 6.1.13, 6.1.14, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.2.2.3 Точка пересечения двух базовых сущностей

Имя функции:	Уровень интерфейса:	1
Pnt_Intersection_2_Ent	Уровень геометрической мощности:	1, 2, 3

Параметр

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNM1	N	Имя первой сущности	basic
Ввод	ENTNM2	N	Имя второй сущности	basic
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_INTERSECTION_2_ENT (ENTNM1, ENTNM2, KFIX)

Результат использования функции

Функция создает сущность *cartesian_point* как точку, принадлежащую обеим заданным базовым сущностям ENTNM1 и ENTNM2. Координаты *X*, *Y* и *Z* созданной декартовой точки *cartesian_point* вычисляются в точке пересечения сущностей ENTNM1 и ENTNM2 в указанном порядке. Если результат пересечения не уникален, то вычисляются оба пересечения. Затем одна из двух точек пересечения выбирается в нижеследующем порядке.

Из двух точек пересечения выбирается та, что лежит ближе к первой точке вычленения *trim_1* заданной сущности *api_line* интерфейса прикладного программирования.

Если одна из заданных сущностей ENTNM1 или ENTNM2 является экземпляром типа *api_line*, а обе заданные сущности ENTNM1 и ENTNM2 являются экземплярами типа *api_circular_arc*, то:

1) создается вектор v_n, равный векторному произведению вектора *v1* (это вектор, проведенный из центра базовой кривой *basis_curve* первой дуги окружности *api_circular_arc* в центр базовой кривой второй дуги окружности) и вектора *v2* (это вектор, перпендикулярный плоскости (Oxy) первой дуги окружности, если значение флажка обхода кривой *sense_agreement* равно «true», либо вектор противоположного направления, если значение флажка обхода кривой равно «false»);

2) вектор v_{res} создается как вектор, проведенный из центра базовой кривой первой дуги окружности в точку пересечения;

3) пересечение точек должно гарантировать, что скалярное произведение $v_{res}\, v_n > 0$ положительно.

Полученная декартова точка *cartesian_point* имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечания

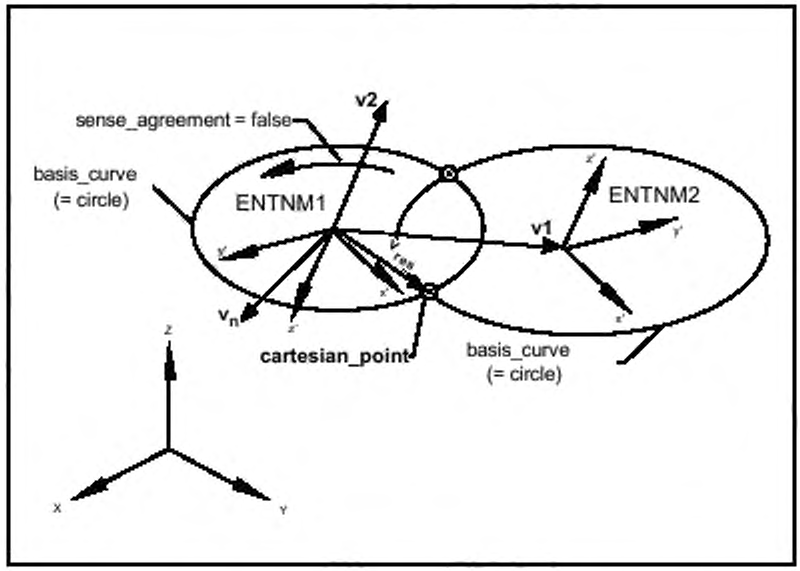
1 Пересечение имеет место, если минимальное расстояние между двумя соединяемыми сущностями (ENTNM1 и ENTNM2) меньше [*ZERO_value*]. При этом существует не более двух точек с указанным качеством, присущим двум отдельным сущностям.

2 Декартова точка *cartesian_point* создается, если она лежит внутри параметрического диапазона [*trim_1*, *trim_2*] для обеих заданных базовых сущностей кривых. В противном случае возникает ошибка.

3 Если расстояние между выбранной декартовой точкой *cartesian_point* и точкой начала (конца) (*trim_1* или *trim_2*) заданного базовой сущностью кривой лежит в диапазоне [*ZERO_value*, EPS], то ошибок нет и координаты указанной точки вычленения используются для создания новой декартовой точки.

4 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1) и процесс выбора для сущностей ENTNM1 and ENTNM2 типа *api_circular_arc* имеет место в виртуальном 3D-пространстве, то значение *Z* игнорируется интерфейсом.

5 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса не менее 2), то обе заданные сущности ENTNM1 и ENTNM2 должны лежать в одной плоскости.



Sense_agreement = false — значение флага направления обхода контура равно «false», basis_curve (= circle) — базовая кривая (окружность), ENTNM1 — первая сущность; ENTNM2 — вторая сущность, v_{res} — результирующий вектор; v_n — нормальный вектор; cartesian_point — декартова точка

Рисунок А.12 — Функция Pnt_Intersection_2_Ent (3D-вид)

Внутренние ссылки: 6.1.9.2,6.1.12,6.2.4,8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
110	Попытка создания точки вне параметрического диапазона сущности кривой	115	Заданные сущности являются идентичными
118	Заданные сущности кривой являются параллельными (концентрическими)	119	Заданные сущности не лежат в одной плоскости
122	Нет пересечения заданной сущности кривой	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

А.5.2.2.4 Точка касания дуги окружности

Имя функции:	Уровень интерфейса:	1
Pnt_Tangential_Arc	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ARCNAM	N	Имя сущности *api_circular_arc*	arc
Ввод	LINNAM	N	Имя ссылочной сущности *api_line*	lin
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_TANGENTIAL_ARC (ARCNAM, LINNAM, KFIX)

Результат использования функции

Создает декартову точку *cartesian_point* как точку касания дуги окружности *api_circular_arc* с именем ARCNAM заданной ссылочной прямой *api_line* с именем LINNAM. Указанная ссылочная прямая не должна содержать центра базовой кривой *basis_curve* сущности ARCNAM. Координаты *X*, *Y* и *Z* созданной декартовой точки вычисляются с помощью точек, лежащих на сущности ARCNAM, и точек, лежащих:

- на линии, параллельной ссылочной линии, касающейся базовой кривой заданной сущности ARCNAM;
- ближе к бесконечной прямой LINNAM.

Указанная декартовая точка имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

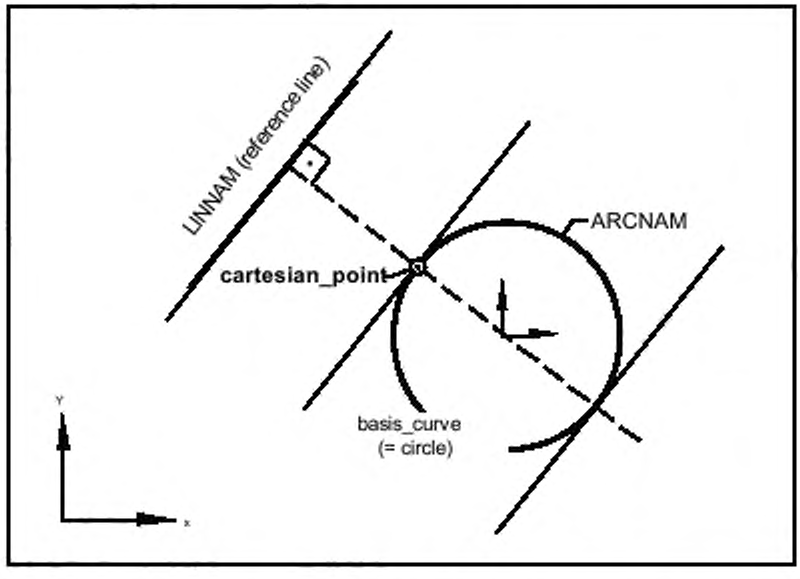
Примечания

1 Декартова точка *cartesian_point* создается, если она лежит внутри параметрического диапазона [*trim_1*, *trim_2*] заданной базовой кривой ARCNAM. В противном случае возникает ошибка;

2 Если расстояние между выбранной декартовой точкой и точкой начала (конца) (*trim_1* или *trim_2*) заданной базовой кривой ARCNAM лежит в диапазоне [*ZERO_value*, EPS], то ошибок нет и координаты указанной точки вычленения используются для создания новой сущностью *cartesian_point*;

3 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то значение *Z* игнорируется интерфейсом;

4 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса не менее 2), то обе заданные сущности ARCNAM и LINNAM должны лежать в одной плоскости.



LINNAM (reference line) — ссылочная прямая линия; ARCNAM — дуга окружности; cartesian_point — декартова точка; basis_curve (= circle) — базовая кривая (окружность)

Рисунок А.13 — Функция Pnt_Tangential_Arc

Внутренние ссылки: 6.1.9.2, 6.1.12, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
110	Попытка создания точки вне параметрического диапазона сущности кривой	119	Заданные сущности не лежат в одной плоскости
127	Геометрическое построение нецелесообразно	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.2.2.5 Точка в центре дуги окружности

Имя функции:	Уровень интерфейса:	1
Pnt_Center_Arc	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ARCNAM	N	Имя сущности *api_circular_arc*	curves
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_CENTER_ARC (ARCNAM, KFIX)

Результат использования функции

Функция создает сущность *cartesian_point* в центре заданной сущности *api_circular_arc* с именем ARCNAM. Координаты *X*, *Y* и *Z* созданной декартовой точки *cartesian_point* вычисляются по расположению *position.location* базовой кривой *basis_curve* сущности *api_circular_arc* ARCNAM. Полученная декартова точка имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то значение *Z* игнорируется интерфейсом.

Внутренние ссылки: 6.1.9.2, 6.1.12, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.2.2.6 Точка в середине базовой сущности

Имя функции:	Уровень интерфейса:	1
Pnt_Middle_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	basic
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_MIDDLE_ENT (ENTNAM, KFIX)

Результат использования функции

Функция создает сущность *cartesian_point* в середине заданной базовой сущности ENTNAM. Координаты *X*, *Y* и *Z* созданной декартовой точки *cartesian_point* вычисляются по точкам, лежащим в середине параметрического диапазона между первой и второй точками вычленения заданной сущности ENTNAM. Указанная декартова точка имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечания

1 Сущность *cartesian_point* создается, если вычисляемое расстояние между данной декартовой точкой и обеими точками начала и конца (*trim_1* или *trim_2*) заданной базовой кривой ENTNAM больше EPS. В противном случае возникает ошибка.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то значение *Z* игнорируется интерфейсом.

Внутренние ссылки: 6.1.9.2, 6.1.12, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
110	Попытка создания точки вне параметрического диапазона сущности кривой	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.2.2.7 Точка как проекция на базовую сущность

Имя функции:	Уровень интерфейса:	1
Pnt_Projection_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PNTNAM	N	Имя проектируемой сущности *cartesian_point*	pnt
Ввод	ENTNAM	N	Имя сущности	basic
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_PROJECTION_ENT (PNTNAM, ENTNAM, KFIX)

Результат использования функции

Функция создает сущность *cartesian_point* как точку на заданной базовой сущности ENTNAM путем проецирования заданной декартовой точки *cartesian_point* с именем PNTNAM.

Если заданная базовая сущность ENTNAM является экземпляром типа линии *api_line* интерфейса прикладного программирования, то функция создает сущность *cartesian_point* с координатами *X*, *Y* и *Z*, вычисленными по точке, полученной ортогональным проецированием точки PNTNAM на прямую, которая является базовой кривой *basis_curve* сущности *api_line* интерфейса прикладного программирования.

Если заданная базовая сущность ENTNAM является экземпляром типа дуги окружности *api_circular_arc*, то функция создает сущность *cartesian_point* с координатами *X*, *Y* и *Z*, полученными в результате пересечения бесконечной прямой, построенной по точке PNTNAM, по центру заданной дуги окружности *api_circular_arc* ENTNAM и по самой дуге окружности *api_circular_arc*. Если возможны две точки пересечения, то выбирается ближайшая к точке PNTNAM.

Новая созданная декартова точка *cartesian_point* имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

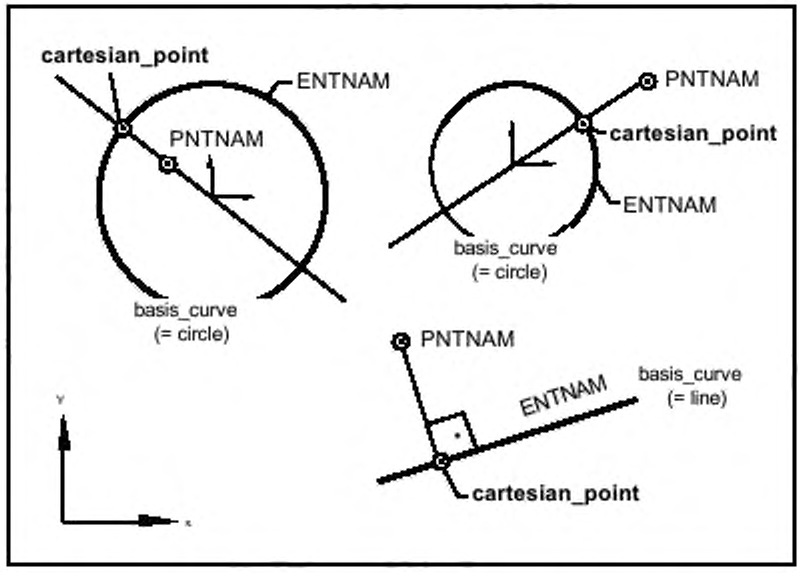
Примечания

1 Декартова точка *cartesian_point* создается, если она находится внутри параметрического диапазона [*trim_1, trim_2*] заданной базовой кривой ENTNAM. В противном случае возникает ошибка.

2 Если расстояние, измеренное между новой декартовой точкой и одной точкой начала (конца) (*trim_1* или *trim_2*) заданной базовой кривой ENTAM, лежит в диапазоне [*ZERO_value*, EPS], то координаты точки вычленения используются для создания новой декартовой точки *cartesian_point*.

3 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то значение *Z* игнорируется интерфейсом.

4 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 2), то обе заданные сущности PNTNAM и ENTNAM должны лежать в одной плоскости.



Cartesian_point — декартова точка; ENTNAM — сущность; PNTNAM — точка; basis_curve (= circle) — базовая кривая (окружность)

Рисунок А.14 — Функция Pnt_Projection_Ent

Внутренние ссылки: 6.1.9.2, 6.1.12, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
110	Попытка создания точки вне параметрического диапазона сущности кривой	119	Заданные сущности не лежат в одной плоскости
127	Геометрическое построение нецелесообразно	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.2.2.8 Точка как проекция на сущность axis2_placement

Имя функции:	Уровень интерфейса:	1
Pnt_Projection_A2p	Уровень геометрической мощности:	1, 2, 3

Параметры

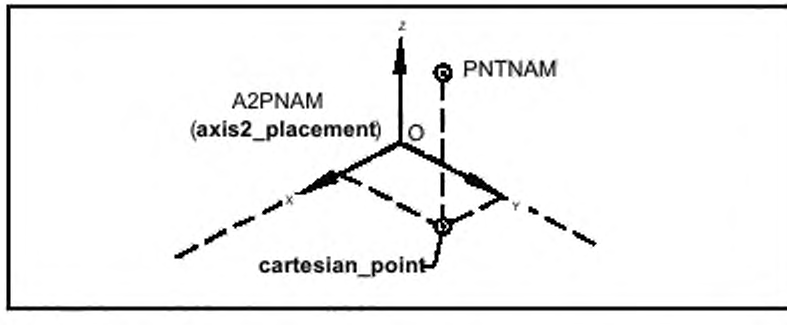
Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PNTNAM	N	Имя проектируемой декартовой точки *cartesian_point*	pnt
Ввод	A2PNAM	N	Имя сущности *axis2_placement*	a2p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *cartesian_point*	pnt

Привязка языка FORTRAN:
NAME = PNT_PROJECTION_A2P (PNTNAM, A2PNAM, KFIX)

Результат использования функции

Функция создает сущность *cartesian_point* путем проецирования заданной декартовой точки *cartesian_point* с именем PNTNAM на плоскость (Oxy) сущности *axis2_placement* с именем A2PNAM. Координаты *X*, *Y* и *Z* созданной *cartesian_point* вычисляются по точке на плоскости (Oxy) для сущности *axis2_placement* A2PNAM. Расстояние от данной точки до сущности PNTNAM является наименьшим. Указанная декартова точка имеет текущую запись *point_style* в таблице статуса интерфейса. Функция возвращает имя полученной сущности *cartesian_point*. При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то ошибок нет и координаты вновь созданной декартовой точки копируются из сущности PNTNAM.



PNTNAM — исходная точка, A2PNAM (axis2_placement) — координатная плоскость локальной координатной системы; cartesian_point — проекция точки на плоскость

Рисунок А.15 — Функция Pnt_Projection_A2p

Внутренние ссылки: 6.1.9.2, 6.1.9.7, 6.1.9.8, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.3 Сущности кривой

А.5.3.1 Базовые сущности кривой

А.5.3.1.1 Отрезки прямой api_line

Построение отрезка по двум точкам	Lin_2_Pnt
Построение отрезка по точке, длине отрезка и его направлению	Lin_Pnt_Length_Dir
Построение отрезка, касательного к дуге одной окружности	Lin_Tangential_Arc
Построение отрезка, касательного к дугам двух окружностей	Lin_Tangential_2_Arc
Диагональное сопряжение двух отрезков	Lin_Chamfer_2_Lin

А.5.3.1.1.1 Построение отрезка по двум точкам

Имя функции:

Lin_2_Pnt

Уровень интерфейса:	1
Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	STAPNT	N	Имя начальной точки *cartesian_point*	pnt
Ввод	ENDPNT	N	Имя конечной точки *cartesian_point*	pnt
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_line*	lin

Привязка языка FORTRAN:
NAME = (STAPNT, ENDPNT, KFIX)

Результат использования функции

Функция создает отрезок прямой *api_line* по двум заданным декартовым точкам *cartesian_point*, где PNTNM1 — начальная точка и PNTNM2 — конечная точка.

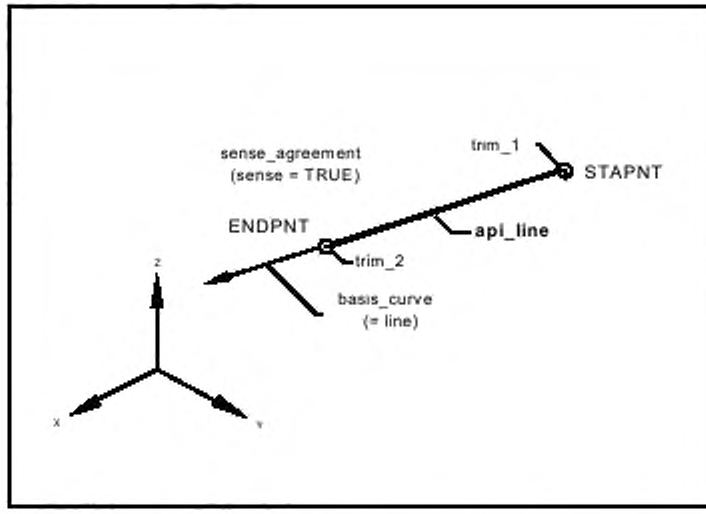
Начальная и конечная точки дублируются как точки *p1* и *p2* соответственно. Они имеют нулевой стиль *null_style*. Затем:

- создается экземпляр *d* направления *direction* с компонентами *direction_ratio*, определенными как разность *p2 — p1*. Указанное направление имеет нулевой стиль;
- создается экземпляр *v* вектора *vector*, направление которого совпадает с направлением *d*, модуль равен *||p2 — p1||*. Настоящий вектор имеет нулевой стиль;
- создается экземпляр линии *l* по точке *pnt p1* и направлению *dir v*. Настоящая линия имеет нулевой стиль;

- создается экземпляр сущности *api_line* с линией *l* в качестве базовой кривой *basis_curve*, точки *p1* и *p2* являются точками вычленения *trim_1* и *trim_2* соответственно. Атрибут направления обхода отрезка *sense_agreement* равен «true», значение атрибута главного представления *master_representation* зависит от реализации. Полученная сущность *api_line* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной сущности *api_line*.

Расстояние между двумя заданными точками лежит в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



Trim_1 — точка вычленения 1, sense_agreement (sense = TRUE) — положительное направление отрезка (совпадает с направлением оси); STAPNT — начальная точка отрезка; ENDPNT — конечная точка отрезка; api_line — сущность линии; trim_2 — точка вычленения 1: basis_curve (= line) — базовая кривая (прямая линия)

Рисунок А.16 — Функция Lin_2_Pnt

Внутренние ссылки: 6.1.9, 6.1.12.1, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
101	Попытка создания вырожденной сущности	103	Расстояние между двумя точками находится вне установленного диапазона [EPS, MAX]
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.3.1.1.2 Построение отрезка по точке, длине отрезка и его направлению

Имя функции:	Уровень интерфейса:	1
Lin_Pnt_Length_Dir	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	STAPNT	N	Имя начальной точки *cartesian_point*	pnt
Ввод	LEN	D	Длина отрезка *api_line*	(EPS ≤ LEN ≤ MAX)
Ввод	DIRNAM	N	Имя направления отрезка	dir
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_line*	lin

Привязка языка FORTRAN:
NAME = LIN_PNT_LENGTH_DIR (STAPNT, LEN, DIRNAM, KFIX)

Результат использования функции

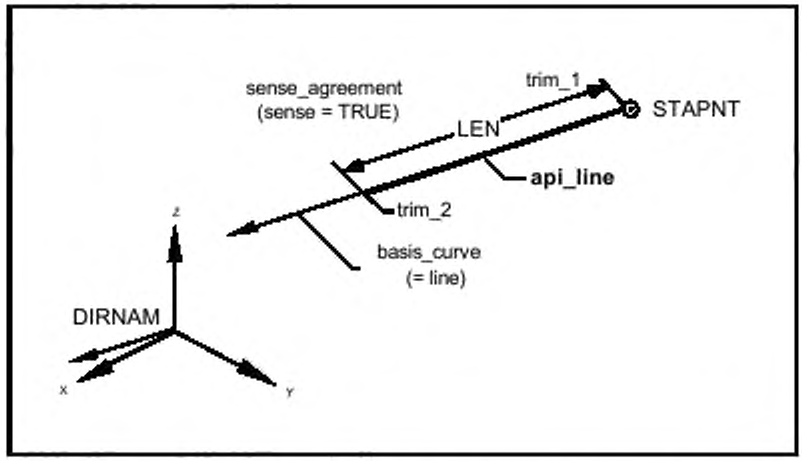
Функция создает отрезок прямой *api_line* по начальной декартовой точке *cartesian_point*, длине и направлению *direction*.

Начальная декартова точка с именем STAPNT дублируется как точка *p1*, направление с именем DIRNAM дублируется как вектор *d*. Обе сущности имеют нулевой стиль *null_style*. Затем:

- создается экземпляр *v* вектора *vector* с направлением *d* и единичным модулем *magnitude* (нормированный вектор). Полученный вектор имеет нулевой стиль;
- создается экземпляр *l* линии *line* по точке *pnt p1* и направлению *dir v*. Полученная линия имеет нулевой стиль;
- создается экземпляр *api_line* интерфейса прикладного программирования с линией *l* в качестве базовой кривой *basis_curve*. Точка *p1* является точкой вычленения *trim_1*, значение параметра LEN (длина отрезка) определяет вторую точку вычленения *trim_2*. Значение атрибута направления обхода отрезка *sense_agreement* равно «true», значение атрибута главного представления *master_representation* зависит от реализации. Полученная сущность *api_line* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной сущности *api_line*.

Значение параметра длины LEN лежит в диапазоне [EPS, MAX] и измеряется в единицах длины *OVC_length_unit*. При возникновении ошибки сущность не создается и функция возвращает нулевое имя элемента.

Примечание — Нет.



Trim_1 — точка вычленения 1; sense_agreement (sense = TRUE) — положительное направление отрезка (совпадает с направлением оси); STAPNT — начальная точка отрезка; LEN — заданная длина отрезка; api_line — сущность линии; trim_2 — точка вычленения 2; basis_curve (= line) — базовая кривая (прямая линия); DIRNAM — заданное направление отрезка

Рисунок А.17 — Функция Lin_Pnt_Length_Dir

Внутренние ссылки: 6.1.9, 6.1.12.1, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.3.1.1.3 Построение отрезка, касательного к дуге одной окружности

Имя функции:	Уровень интерфейса:	1
Lin_Tangential_Arc	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	STAPNT	N	Имя начальной точки *cartesian_point*	pnt
Ввод	ARCNAM	N	Имя дуги окружности *api_circular_arc*	arc
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_line*	lin

Привязка языка FORTRAN:
NAME = LIN_TANGENTIAL_ARC (STAPNT, ARCNAM, KFIX)

Результат использования функции

Функция создает отрезок прямой *api_line*, касательный к заданной дуге окружности *api_circular_arc* с именем ARCNAM. Начальная точка отрезка задается сущностью *cartesian_point* с именем STAPNT. Данная начальная точка дублируется точкой *p1*, имеющей нулевой стиль *null_style*. Затем:

- создаются экземпляры *p2* и *p3* двух декартовых точек *cartesian_point* как двух возможных точек касания базовой кривой *basis_curve* типа дуги окружности *api_circular_arc* с именем ARCNAM. Полученные декартовы точки имеют нулевой стиль;
- из полученных двух декартовых точек *p2* и *p3* выбирается та, в которой направление касательной (в положительном направлении с учетом значения флажка сущности ARCNAM) совпадает с направлением перехода из начальной точки *p1* в полученную точку касания. Пусть точка *p4* является синонимом данной выбранной точки;
- создается экземпляр *d* направления *direction* с компонентами *direction_ratio*, вычисленными как разность *p4 — p1*. Указанное направление имеет нулевой стиль;
- создается экземпляр *v* вектора *vector* с направлением *d* и модулем ||*p4 — p1*||. Настоящий вектор имеет нулевой стиль;
- создается экземпляр *l* линии *line* по точке *pnt p1* и направлению *dir v*. Данная линия имеет нулевой стиль;
- создается экземпляр *l* линии *api_line* как базовой кривой *basis_curve*. Точки *p1* и *p4* являются точками вычленения *trim_1* и *trim_2* соответственно. Значение атрибута направления обхода отрезка *sense_agreement* равно «true», значение атрибута *master_representation* зависит от реализации. Настоящая сущность *api_line* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной сущности *api_line*. При возникновении ошибки сущность не создается и функция возвращает нулевое имя элемента.

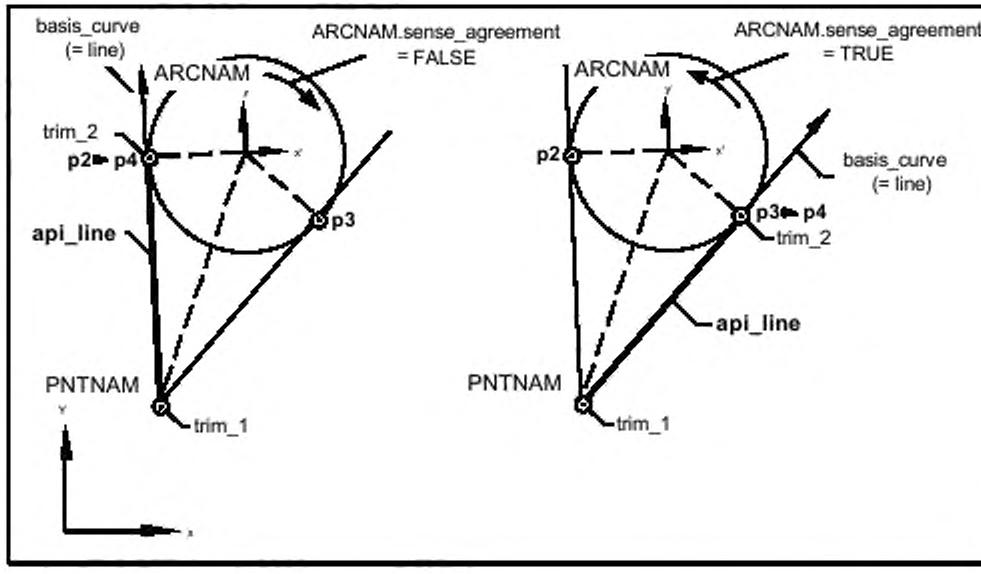
Примечания

1 Заданная начальная точка отрезка *cartesian_point p1* должна лежать вне базовой кривой *basis_curve* дуги окружности *api_circular_arc* с именем ARCNAM.

2 Если расстояние между вычисляемой точкой касания *p4* и одной из точек вычленения (*trim_1* или *trim_2*) заданной дуги окружности лежит в диапазоне [*ZERO_value*, EPS], то координаты данной точки вычленения используются вместо координат вычисляемой точки.

3 Сущность *api_line* создается, если выбранная точка касания *p4* лежит внутри параметрического диапазона [*trim_1*, *trim_2*] дуги окружности ARCNAM. При этом длина данной сущности лежит в диапазоне [EPS, MAX]. В противном случае возникает ошибка.

4 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса не менее 2), то декартова точка *cartesian_point* с именем PNTNAM и дуга окружности ARCNAM должны лежать в одной плоскости.



Basis_curve (= line) — базовая кривая (прямая линия); ARCNAM.sense_agreement = FALSE — отрицательное направление дуги; ARCNAM.sense_agreement = TRUE — положительное направление дуги; ARCNAM — дуга окружности с именем; trim_2 — точка вычленения 2; api_line — отрезок касательной, PNTNAM — начальная точка, trim_1 — точка вычленения 1

Рисунок А.18 — Функция Lin_Tangential_Arc

Внутренние ссылки: 6.1.9, 6.1.12, 6.1.12.1, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
101	Попытка создания вырожденной сущности	105	Попытка создания вырожденного направления в процессе создания сущности
110	Попытка создания точки вне параметрического диапазона сущности кривой	119	Заданные сущности не лежат в одной плоскости
127	Геометрическое построение нецелесообразно	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.3.1.1.4 Построение отрезка, касательного к дугам двух окружностей

Имя функции:	Уровень интерфейса:	1
Lin_Tangential_2_Arc	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ARCNM1	N	Имя первой дуги окружности *api_circular_arc*	arc
Ввод	ARCNM2	N	Имя второй дуги окружности *api_circular_arc*	arc
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_line*	lin

Привязка языка FORTRAN:
NAME = LIN_TANGENTIAL_2_ARC (ARCNM1, ARCNM2, KFIX)

Результат использования функции

Функция создает отрезок прямой *api_line*, направленный по линии касания двух заданных дуг окружностей *api_circular_arc*. Отрезок начинается из точки касания первой дуги *api_circular_arc* с именем ARCNM1 и заканчивается в точке касания второй дуги *api_circular_arc* с именем ARCNM2. Положительное направление обхода дуг (неявно заданное значениями флажков и точками вычленения) определяет направление полученного отрезка *api_line*. Ниже дуги *C1* и *C2* рассматриваются как синонимы дуг ARCNM1 и ARCNM2 соответственно. Затем:

- создаются экземпляры всех возможных декартовых точек *cartesian_point*, которые являются точками касания базовых кривых *basis_curve* с именами *C1* и *C2*. Все указанные декартовы точки имеют нулевой стиль *null_style*;

- определяются точки касания на дугах *C1* и *C2*. Пусть *p1* и *p2* — синонимы данных точек соответственно. При построении направление касательной в точке *C1* совпадает с направлением касательной в точке *C2* (с учетом значения флажка направления обхода дуг окружности *api_circular_arc*, соответствующих точкам *C1* и *C2*);

- создается экземпляр *d* направления *direction* с компонентами *direction_ratio*, определенными путем вычитания *p2 — p1*. Полученное направление имеет нулевой стиль;

- создается экземпляр *v* вектора *vector*, направление которого совпадает с направлением *d*, модуль вектора равен ||*p2 — p1*||. Полученный вектор имеет нулевой стиль;

- создается экземпляр *l* линии *line*, содержащий точку *pnt*, равную *p1* и имеющий направление *dir*, равное *v*. Полученная сущность имеет нулевой стиль;

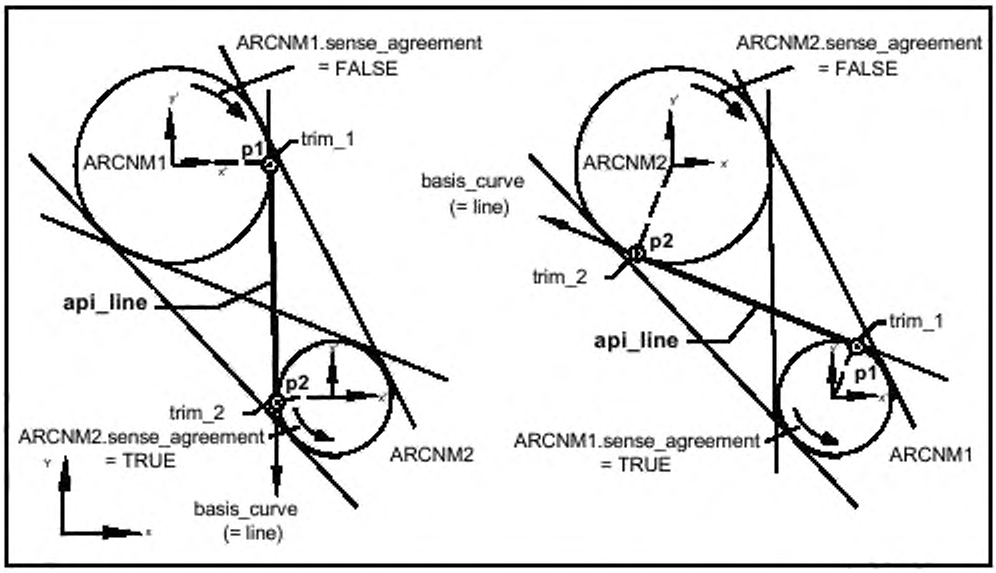
- создается экземпляр *l* линии *api_line* как базовая кривая *basis_curve* с точками *p1* и *p2* в качестве точек вычленения *trim_1* и *trim_2* соответственно. Значение атрибута направления отрезка *sense_agreement* равно «true», а значение атрибута главного представления *master_representation* зависит от реализации. Настоящая сущность *api_line* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true»), полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной сущности *api_line*. При возникновении ошибки сущность не создается и функция возвращает нулевое имя элемента.

Примечания

1 Сущность *api_line* создается, если выбранные точки касания *p1* и *p2* лежат внутри параметрического диапазона [*trim_1*, *trim_2*] соответствующих дуг окружностей *api_circular_arc*. При этом расстояние между двумя точками *p1* и *p2* находится в диапазоне [EPS, MAX]. В противном случае возникает ошибка.

2 Если расстояние между вычисленными точками касания *p1* и *p2* и одной из точек вычленения (*trim_1* или *trim_2*) соответствующей заданной дуги окружности *api_circular_arc* лежит в диапазоне [*ZERO_value*, EPS], то ошибок нет и координаты точки вычленения используются вместо вычисленных координат.

3 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса не менее 2), то обе заданные дуги окружности должны лежать в одной плоскости.



ARCNM1.sense_agreement = FALSE — стрелка указывает направление обхода дуги первой окружности (значение флага «false»), ARCNM2 sense_agreement = FALSE — стрелка указывает направление обхода дуги второй окружности (значение флага «false»); trim_1 — точка вычленения 1, ARCNM1 — первая дуга окружности, ARCNM2 — вторая дуга окружности; basis_curve (= line) — базовая кривая (прямая линия); trim_2 — точка вычленения 2; api_line — отрезок касательной; ARCNM2.sense_agreement = TRUE — стрелка указывает направление обхода дуги первой окружности (значение флажка равно «true»); ARCNM1.sense_agreement = TRUE — стрелка указывает направление обхода дуги второй окружности (значение флажка равно «true»)

Рисунок А.19 — Функция Lin_Tangential_2_Arc

Внутренние ссылки: 6.1.9, 6.1.12, 6.1.12.1, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
101	Попытка создания вырожденной сущности	105	Попытка создания вырожденного направления в процессе создания сущности
110	Попытка создания точки вне параметрического диапазона сущности кривой	115	Заданные сущности идентичны
118	Заданные сущности кривой являются параллельными/концентрическими	119	Заданные сущности не лежат в одной плоскости
127	Геометрическое построение нецелесообразно	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

А.5.3.1.1.5 Диагональное сопряжение двух отрезков

Имя функции:	Уровень интерфейса:	1
Lin_Chamfer_2_Lin	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	LEN1	D	Отрезок на первой прямой *api_line*	(EPS ≤ LEN1 ≤ MAX)
Ввод	LEN2	D	Отрезок на второй прямой *api_line*	(EPS ≤ LEN2 ≤ MAX)
Ввод	LINNM1	N	Имя первой прямой *api_line*	lin
Ввод	LINNM2	N	Имя второй прямой *api_line*	lin
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_line*	lin

Привязка языка FORTRAN:
NAME = LIN_CHAMFER_2_LIN (LEN1, LEN2, LINNM1, LINNM2, KFIX)

Результат использования функции

Функция создает сущность отрезка прямой *api_line* как скос между двумя прямыми *api_line*. Исходные прямые типа *api_line* с именами LINNM1 и LINNM2 соответственно должны иметь точку пересечения, лежащую на каждой из этих прямых. Данные прямые отрезаются, при этом имена их не изменяются. Прямые остаются во временной базе данных до момента их выбора. Вновь созданная сущность *api_line* (скос *chamfer_line*) начинается от нового конца первой прямой *api_line* (точка вычленения *trim_2* на прямой LINNM1) и заканчивается новым началом второй прямой *api_line* (точка вычленения *trim_1* на прямой LINNM2). После вычленения прямые больше не пересекаются. Затем:

- создается экземпляр *p1* декартовой точки *cartesian_point* пересечения прямых LINNM1 и LINNM2. Точка пересечения *cartesian_point* имеет нулевой стиль *null_style*;

- создается экземпляр *p2* декартовой точки на расстоянии LEN1 от точки пересечения *p1*. Указанная точка отложена на прямой *api_line* с именем LINNM1 в отрицательном направлении. Точка *cartesian_point* имеет нулевой стиль;

- создается экземпляр *p3* декартовой точки на расстоянии LEN2 от точки пересечения *p1*. Указанная точка отложена по прямой *api_line* с именем LINNM2 в положительном направлении. Точка имеет нулевой стиль;

- создается экземпляр *d* направления *direction* с компонентами *direction_ratio*, полученными путем вычитания *p3 — p2*. Указанное направление имеет нулевой стиль;

- создается экземпляр *v* вектора с направлением *d* и модулем ||*p3 — p2*||. Настоящий вектор имеет нулевой стиль;

- создается экземпляр *l* линии по точке *pnt p2* и направлению *dir v*. Настоящая линия имеет нулевой стиль;

- создается экземпляр сущности *api_line* с прямой *l* как базовой кривой *basis_curve* и точками *p2* и *p3* как точками вычленения *trim_1* и *trim_2* соответственно. Значение атрибута обхода кривой *sense_agreement* равно «true», значение атрибута главного представления *master_representation* зависит от реализации. Настоящая сущность *api_line* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной сущности *api_line*;

- заданная сущность *api_line* с именем LINNM1 определяется повторно с точкой вычленения *trim_2*, равной *p2*. Заданная сущность с *api_line* с именем LINNM2 также определяется повторно с точкой вычленения *trim_1*, равной *p3*.

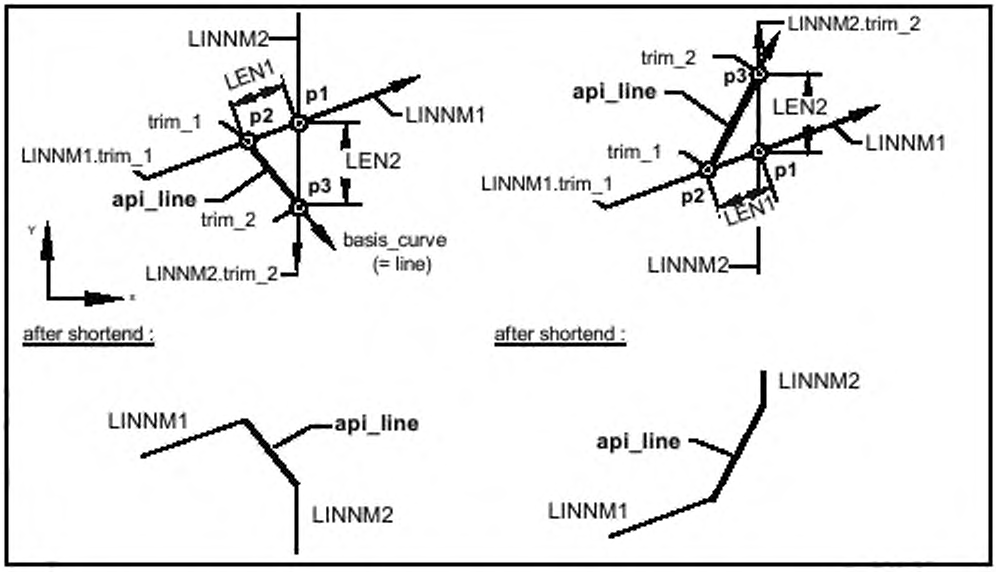
Значения длин LEN1 и LEN2, лежащие в диапазоне [EPS, MAX], измеряются в единицах длины *OVC_length_unit*. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Заданная длина LEN1 должна быть меньше расстояния между точкой пересечения *p1* и точкой вычленения *trim_1* сущности *api_line* с именем LINNM1. Заданная длина LEN2 должна быть меньше расстояния между точкой пересечения *p1* и точкой вычленения *trim_2* сущности *api_line* с именем LINNM2. В противном случае возникает ошибка.

2 Сущность *api_line* создается, если ее длина находится в диапазоне [EPS, MAX]. При этом длины обоих заданных сущностей LINNM1 и LINNM2 после вычленения должны быть больше EPS. В противном случае возникает ошибка.

3 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса не менее 2), то обе заданные сущности должны лежать в одной плоскости.



LINNM2.trim_2 — отрезанная прямая 2; LINNM2 — исходная прямая 2, trim_2 — точка вычленения 2, LEN1 — длина 1; api_line — полученный скос, LINNM1 — исходная прямая 1, LEN2 — длина 2, trim_1 — точка вычленения 1, LINNM1.trim_1 — отрезанная прямая 1; basis_curve (= line) — базовая кривая (прямая линия); after shortend — скос прямых после вычленения

Рисунок А.20 — Функция Lin_Chamfer_2_Lin

Внутренние ссылки: 6.1.9, 6.1.12, 6.1.12.1, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
105	Попытка создания вырожденного направления в процессе создания сущности	108	Попытка создания вырожденной базовой кривой в процессе создания сущности
111	Попытка создания сегмента линии, длиной вне установленного диапазона [EPS, MAX]	115	Заданные сущности являются идентичными
119	Заданные сущности не лежат в одной плоскости	120	Задан слишком длинный отрезок
122	Заданные сущности кривых не пересекаются	127	Геометрическое построение нецелесообразно
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.3.1.2 Построение окружностей и их дуг (сущности api_circular_arc)

Построение окружности по радиусу и сущности *axis2_placement*	Circle_Rad_A2p
Построение дуги окружности по трем точкам	Arc_3_Pnt
Построение дуги окружности по радиусу, двум углам и сущности *axis2_placement*	Arc_Rad_2_angle_A2p

Построение дуги окружности по радиусу и трем точкам	Arc_Rad_3_Pnt
Построение дуги окружности по радиусу, двум точкам и сущности *axis 2_placement*	Arc_Rad_2_Pnt_A2p
Построение дуги окружности как сопряжение двух сущностей	Arc_Fillet_2_Ent
Построение дуги окружности, касательной к двум сущностям	Arc_Tangential_2_Ent
Построение дуги окружности по радиусу и двум сущностям	Arc_Rad_2_Ent
Построение дуги окружности по трем сущностям	Arc_3_Ent

A.5.3.1.2.1 Построение окружности по радиусу и сущности axis2_placement

Имя функции:	Уровень интерфейса:	1
Circle_Rad_A2p	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	RAD	D	Радиус окружности *api_circular_arc*	(EPS ≤ RAD ≤ MAX)
Ввод	A2PNAM	N	Имя сущности *axis2_placement*	a2p
Ввод	SENSE	E	Знак направления обхода кривой	[TRUE, FALSE]
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_circular_arc*	arc

Привязка языка FORTRAN:
NAME = CIRCLE_RAD_A2P (RAD, A2PNAM, SENSE, KFIX)

Результат использования функции

Функция создает полную окружность как сущность *api_circular_arc* по радиусу RAD, по сущности *axis2_placement* (начало локальной координатной системы с именем A2PNAM) и по установленному положительному направлению обхода (заданному значением флажка «sense») окружности *api_circular_arc* в совокупности с сущностью *circle* как базовой кривой *basis_curve*.

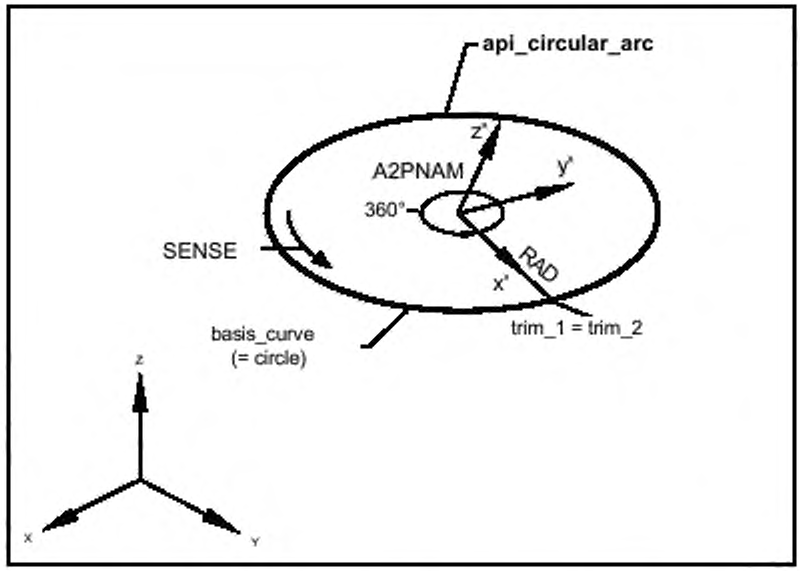
Сущность *axis2_placement* (с именем A2PNAM) дублируется как точка *a2p1*, имеющая нулевой стиль *null_style*. Затем:

- создается экземпляр окружности с центром *a2p1* и радиусом *rad*. Данная окружность имеет нулевой стиль;

- создается экземпляр сущности *api_circular_arc* с элементом *c* как базовой кривой *basis_curve*. Значение параметра равно 0° в точке вычленения *trim_1*, значение параметра равно 360° в точке вычленения *trim_2*. Флажок направления обхода окружности *sense_agreement* принимает значение SENSE, значение атрибута *master_representation* зависит от реализации. Построенный атрибут *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной дуги окружности *api_circular_arc* интерфейса прикладного программирования.

Значение радиуса окружности RAD лежит в диапазоне [EPS, MAX]. Радиус измеряется в единицах длины *OVC_length_unit*. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Интерфейс обеспечивает замкнутость созданной сущности *api_circular_arc* (декартовы координаты точки вычленения *trim_1* равны декартовым координатам точки вычленения *trim_2*).



Api_circular_arc — построенная окружность; A2PNAM — центр окружности находится в начале локальной координатной системы; SENSE — положительное направление обхода окружности; RAD — радиус окружности; trim_1 = trim_2 — условие замкнутости окружности (начальная точка окружности совпадает с конечной точкой окружности); basis_curve (= circle) — базовая кривая (окружность)

Рисунок А.21 — Функция Circle_Rad_A2p

Внутренние ссылки: 6.1.9, 6.1.12, 6.1.12.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.3.1.2.2 Построение дуги окружности по трем точкам

Имя функции:	Уровень интерфейса:	1
Arc_3_Pnt	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	STAPNT	N	Имя начальной точки *cartesian_point*	pnt
Ввод	INTPNT	N	Имя промежуточной точки *cartesian_point*	pnt
Ввод	ENDPNT	N	Имя конечной точки *cartesian_point*	pnt
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_circular_arc*	arc

Привязка языка FORTRAN:
NAME = ARC_3_PNT (STAPNT, INTPNT, ENDPNT, KFIX)

Результат использования функции

Функция создает дугу окружности *api_circular_arc* по трем декартовым точкам *cartesian_point* (начальная точка STAPNT, промежуточная точка INTPNT и конечная точка ENDPNT). Начальная точка (с именем STAPNT) определяет начало дуги окружности, конечная точка (с именем ENDPNT) определяет конец дуги окружности и промежуточная точка (с именем INTPNT) определяет плоскость, в которой лежит окружность и сектор окружности *api_circular_arc*.

Декартовы точки STAPNT и ENDPNT дублируются как точки *p1* и *p3* соответственно. Они имеют нулевой стиль *null_style*. Точка *p2* является синонимом точки INTPNT. Затем:

- создается экземпляр *p4* декартовой точки как центр дуги окружности. Ее координаты вычисляются по координатам трех точек: *p1*, *p2* и *p3*. Полученная декартова точка имеет нулевой стиль;

- создается экземпляр *d1* направления *direction* с компонентами *direction_ratio*, определенными вычитанием *p1 — p4*. Указанное направление имеет нулевой стиль.

В случае 3D-вида:

- создается экземпляр направления *d0* с компонентами *direction_ratio*, вычисляемыми посредством векторного произведения вектора *p1 — p3* и вектора *P2 — p3*. Указанное направление имеет нулевой стиль;

- создается экземпляр *a2p1* сущности *axis2_placement_3d* с началом координат *p4*, осью *d0* и ссылочным направлением *d1* сущности *ref_direction*. Данная сущность имеет нулевой стиль.

В случае 2D-вида:

- создается экземпляр *a2p1* сущности *axis2_placement_2d* с началом координат *p4* и ссылочным направлением *d1* сущности *ref_direction*. Данная сущность имеет нулевой стиль.

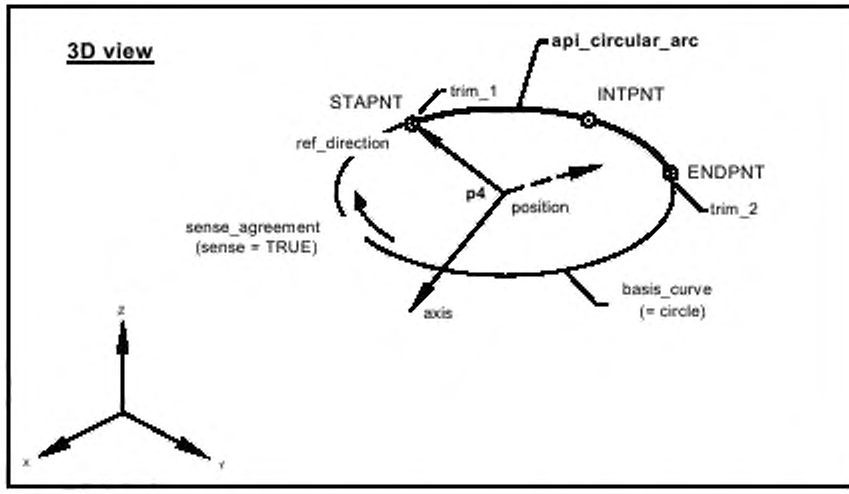
Затем:

- создается экземпляр *c* окружности с центром *a2p1* в начале координат сущности *axis2_placement* и радиусом, извлеченным из ||*p1 — p4*||. Полученная окружность имеет нулевой стиль;

- создается экземпляр *c* сущности *api_circular_arc* в качестве базовой кривой *basis_curve*, точки *p1* и *p3* используются как точки вычленения *trim_1* и *trim_2* соответственно. Значение атрибута направления контура *sense_agreement* вычисляется по положению декартовых точек *cartesian_point p1*, *P2* и *p3*, значение атрибута главного представления *master_representation* зависит от реализации. Полученная дуга окружности *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя построенной дуги окружности *api_circular_arc*.

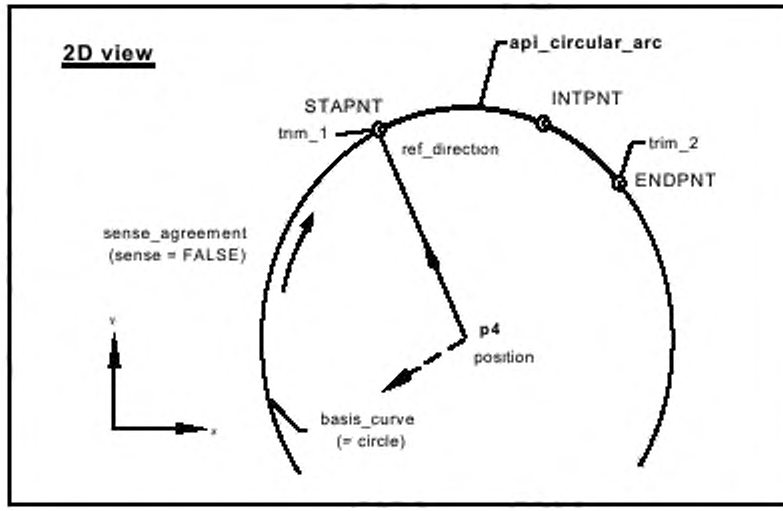
Расстояние между двумя из трех точек не должно лежать в диапазоне [*ZERO_value*, EPS]. Кроме того, промежуточная точка INTPNT не должна лежать на одной прямой с начальной и конечной точками с точностью EPS. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Сущность *api_circular_arc* создается, если радиус лежит в диапазоне [EPS, MAX], а длина отрезка, проведенного из точки *trim_1* в точку *trim_2* (в положительном направлении в соответствии с сущностью *sense_agreement*), не меньше допуска EPS. В противном случае возникает ошибка.



Api_circular_arc — построенная окружность, 3D view — 3D-вид, trim_1 — первая точка вычленения; INTPNT — промежуточная точка; STAPNT — начальная точка; ref_direction — ссылочное направление, ENDPNT — конечная точка, position — центр окружности; trim_2 — вторая точка вычленения; sense_agreement (sense = TRUE) — направление обхода (значение атрибута SENSE равно «true»); basis_curve (= circle) — базовая кривая (окружность); axis — ось

Рисунок А.22 — Функция Arc_3_Pnt (3D-вид)



Api_circular_arc — построенная окружность; 2D view — 2D-вид; INTPNT — промежуточная точка; STAPNT — начальная точка; trim_1 — первая точка вычленения; trim_2 — вторая точка вычленения; ref_direction — ссылочное направление; ENDPNT — конечная точка; sense_agreement (sense = FALSE) — направление обхода (значение атрибута SENSE равно «false»); position — центр окружности; basis_curve (= circle) — базовая кривая (окружность)

Рисунок А.23 — Функция Arc_3_Pnt (2D-вид)

Внутренние ссылки: 6.1.9, 6.1.12, 6.1.12.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
101	Попытка создания вырожденной сущности	105	Попытка создания вырожденного направления в процессе создания сущности
106	Попытка создания вырожденной локальной координатной системы *axis2_placement* в процессе создания сущности	112	Попытка создания дуги длиной меньше EPS
115	Заданные сущности являются идентичными	116	Заданные точки линейно зависимы
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.3.1.2.3 Построение дуги окружности по радиусу, двум углам и сущности axis2_placement

Имя функции:

Arc_Rad_2_Angle_A2p

Уровень интерфейса:	1
Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	RAD	D	Радиус окружности *api_circular_arc*	(EPS ≤ RAD ≤ MAX)
Ввод	STAPNT	D	Начальный угол в плоскости (Oxy) относительно оси (Ox) заданной сущности *a2p*	(0° ≤ STAANG ≤ 360°)
Ввод	ENDANG	D	Конечный угол в плоскости (Oxy) относительно оси (Ox) заданной сущности *a2p*	(0° ≤ ENDANG ≤ 360°)
Ввод	A2PNAM	N	Имя сущности *axis2_placement*	a2p
Ввод	SENSE	E	Знак направления обхода кривой	[TRUE, FALSE]
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_circular_arc*	arc

Привязка языка FORTRAN:
NAME = ARC_RAD_2_ANGLE_A2P (RAD, STAANG, ENDANG, A2PNAM, SENSE, KFIX)

Результат использования функции

Функция создает дугу окружности *api_circular_arc* по радиусу (RAD), двум углам, началу координатной системы *axis2_placement* (с именем A2PNAM) и заданному положительному направлению обхода (параметр SENSE) созданной сущности *api_circular_arc* в совокупности с сущностью *circle* в качестве базовой кривой *basis_curve* и вместе с начальной и конечной точками, неявно определенными по двум углам STAANG (начальный угол) и ENDANG (конечный угол) соответственно.

Сущность *axis2_placement* (с именем A2PNAM) дублируется как точка *a2p1*, имеющая нулевой стиль *null_style*. Затем:

- создается экземпляр *c* окружности с центром *a2p1* и радиусом RAD. Полученная окружность имеет нулевой стиль;

- создается экземпляр *с* сущности *api_circular_arc* с базовой кривой *basis_curve*, значение параметра STAANG соответствует точке вычленения *trim_1*, значение параметра ENDANG соответствует точке вычленения trim_2. Значение атрибута *sense_agreement* равно параметру SENSE, значение атрибута *master_representation* зависит от реализации. Полученная дуга окружности *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной дуги окружности *api_circular_arc* интерфейса прикладного программирования.

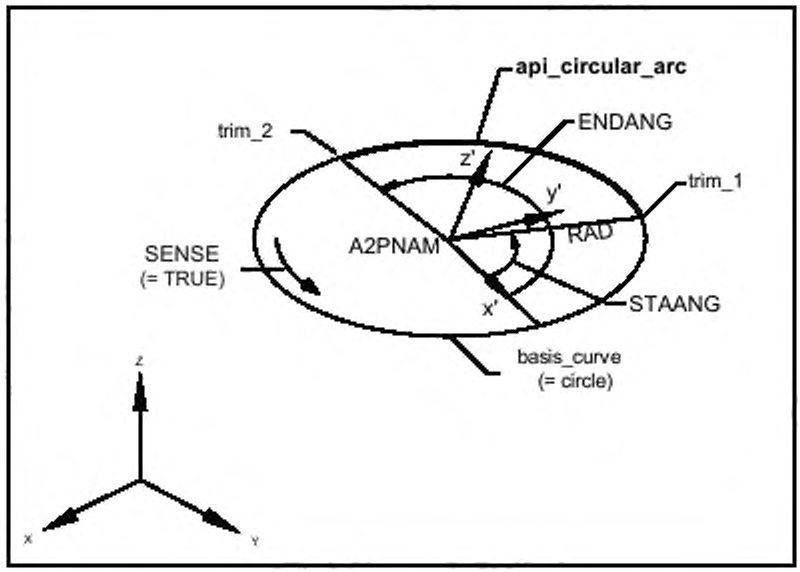
Значения радиуса RAD лежат в диапазоне [EPS, MAX] и измеряются в единицах длины *OVC_length_unit*. Заданные углы измеряются в единицах угла *OVC_angle_unit*, вычисляются в плоскости (Oxy) заданной локальной координатной системы A2PNAM. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Сущность *api_circular_arc* создается, если вычисленное значение радиуса лежит в диапазоне [EPS, MAX] и длина дуги окружности, проведенной из точки вычленения *trim_1* в точку вычленения *trim_2* и совместимой с заданным направлением обхода контура *sense_agreement*, не меньше допуска EPS. В противном случае возникает ошибка.

2 Если два значения параметров STAANG и ENDANG определяют одну и ту же точку в диапазоне [*ZERO_value*, EPS], то обе точки вычленения *trim_1* и *trim_2* являются идентичными и созданная дуга окружности является полной окружностью. Интерфейс гарантирует замкнутость созданной сущности *api_circular_arc* (декартовы координаты точки вычленения *trim_1* равны декартовым координатам точки вычленения *trim_2*).

3 Сущность *api_circular_arc* создается, если длина дуги окружности, проведенной из точки вычленения *trim_1* в точку вычленения *trim_2*, для заданного значения атрибута обхода контура *sense_agreement*, не меньше установленного допуска EPS. В противном случае возникает ошибка.



Api_circular_arc — построенная дуга окружности; ENDANG — конечный угол; trim_2 — вторая точка вычленения; trim_1 — первая точка вычленения, RAD — радиус; A2PNAM — начало локальной координатной системы; SENSE (= TRUE) — направление обхода контура («true»); STAANG — начальный угол: basis_curve (= circle) — базовая кривая (окружность)

Рисунок А.24 — Функция Arc_Rad_2_Angle_A2p

Внутренние ссылки: 6.1.9, 6.1.12, 6.1.12.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
101	Попытка создания вырожденной сущности	112	Попытка создания дуги длиной меньше EPS
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.3.1.2.4 Построение дуги окружности по радиусу и трем точкам

Имя функции:	Уровень интерфейса:	1
Arc_Rad_3_Pnt	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	RAD	D	Радиус дуги *api_circular_arc*	(EPS ≤ RAD ≤ MAX)
Ввод	STAPNT	N	Имя начальной точки *cartesian_point*	pnt
Ввод	ENDPNT	N	Имя конечной точки *cartesian_point*	pnt
Ввод	HLPPNT	N	Имя вспомогательной точки *cartesian_point*	pnt
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_circular_arc*	arc

Привязка языка FORTRAN:
NAME = ARC_RAD_3_PNT (RAD, STAPNT, ENDPNT, HLPPNT, KFIX)

Результат использования функции

Функция создает дугу окружности *api_circular_arc* по радиусу (RAD) и трем декартовым точкам *cartesian_point* (начальная точка STAPNT, конечная точка ENDPNT и вспомогательная точка HLPPNT). Начальная точка (STAPNT) определяет начало дуги окружности, конечная точка (ENDPNT) определяет конец дуги окружности, вспомогательная точка (HLPPNT) задает плоскость и сектор дуги окружности *api_circular_arc* интерфейса прикладного программирования.

Точки STAPNT, ENDPNT, HLPPNT дублируются как точки *p1*, *p2*, *p3* соответственно, они имеют нулевой стиль *null_style*.

В случае 3D-вида создается экземпляр *d0* направления *direction* с компонентами *direction_ratio*, вычисляемыми посредством векторного произведения вектора *p1* — *p3* и вектора *p2* — *p3*. Указанное направление имеет нулевой стиль. Затем:

- определяется виртуальная плоскость путем задания декартовой точки *p3* и направления *d0*, перпендикулярного к виртуальной плоскости;
- создаются экземпляры двух декартовых точек *p4* и *p5* с координатами, равными координатам точек пересечения двух виртуальных окружностей в предварительно определенной виртуальной плоскости с радиусом RAD и центрами в точках *p1* и *p2* соответственно. Декартовы точки *p4* и *p5* имеют нулевой стиль;
- из декартовых точек *p4* и *p5* выбирается та, которая расположена ближе к точке *p3*. Она принимается за центр новой дуги окружности *api_circular_arc*. Новому центру назначается имя *p6*;
- создается экземпляр *d1* направления *direction* с компонентами *direction_ratio*, определенными вектором *p1* — *p6*. Указанное направление имеет нулевой стиль;
- создается экземпляр *a2p1* координатной плоскости *axis2_placement_3d* с началом координат *p6*, осью *d0* и ссылочным направлением *d1* для сущности *ref_direction*. Сущность *axis2_placement_3d* имеет нулевой стиль.

В случае 2D-вида создаются экземпляры декартовых точек *p4* и *p5* с координатами, вычисленными по точкам пересечения двух виртуальных окружностей в плоскости (Oxy) текущей базовой координатной системы вида объекта OVC.

Радиус указанных окружностей равен RAD, центры окружностей лежат в декартовых точках *p1* и *p2* соответственно. Указанные декартовы точки имеют нулевой стиль;

- из двух точек *p4* и *p5* выбирается та, которая расположена ближе к точке *p3*. Она является центром новой дуги окружности *api_circular_arc*, ей назначается имя *p6*;

- создается экземпляр *d1* направления *direction* с компонентами *direction_ratio*, определенными вектором *p1 — p6*. Указанное направление имеет нулевой стиль;

- создается экземпляр *a2p1* сущности *axis2_placement_2d* с началом координат *p6* и базовым направлением *d1*, определяемым сущностью *ref_direction*. Данная сущность *axis2_placement_2d* имеет нулевой стиль;

- создается экземпляр *c* окружности с центром *a2p1* и радиусом RAD. Настоящая окружность имеет нулевой стиль;

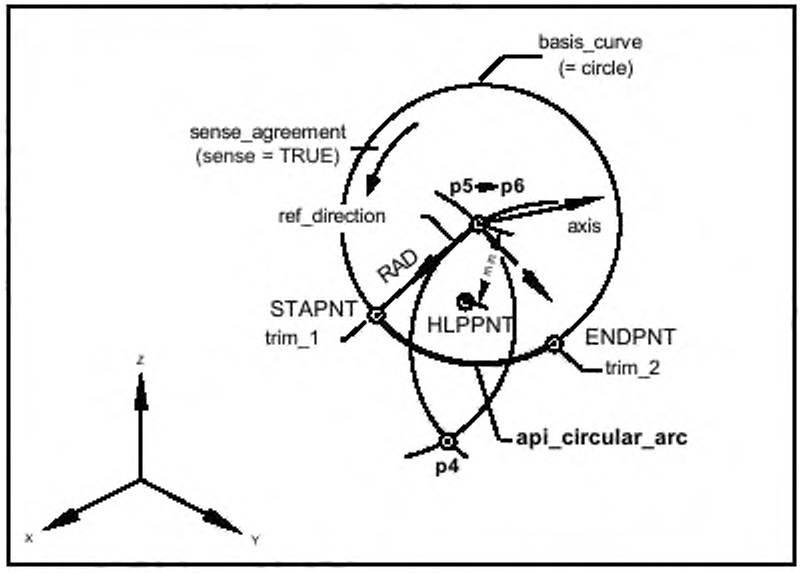
- создается экземпляр *c* дуги окружности *api_circular_arc* с базовой кривой *basis_curve* и точками *p1* и *p2* как точками вычленения *trim_1* и *trim_2* соответственно. Атрибут направления обхода дуги окружности *sense_agreement* вычисляется по точке p3, он определяет требуемый сектор окружности. Атрибут сущности главного представления *master_representation* зависит от реализации. Полученная дуга окружности *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной дуги окружности *api_circular_arc* интерфейса прикладного программирования.

Расстояние между любыми двумя из трех точек не должно лежать в диапазоне [*ZERO_value*, EPS]. Кроме того, вспомогательная точка *p3* не должна находиться в EPS-окрестности центра дуги окружности. Значение радиуса RAD лежит в диапазоне [EPS, MAX] и измеряется в единицах длины *OVC_length_unit*. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Сущность *api_circular_arc* создается, если длина дуги окружности, проведенной из точки вычленения *trim_1* в точку вычленения *trim_2* и совместимой со значением атрибута *sense_agreement*, не меньше допуска EPS. В противном случае возникает ошибка.

2 Сущность *api_circular_arc* создается, если вспомогательная точка HLPPNT определяет один уникальный сектор (расстояние между вычисленным центром окружности и настоящей вспомогательной точкой HLPPNT не равно RAD). В противном случае возникает ошибка.



Basis_curve (= circle) — базовая кривая (окружность); sense_agreement (sense = TRUE) — положительное направление обхода кривой; ref_direction — ссылочное направление, axis — ось; RAD — радиус; STAPNT — начальная точка, HLPPNT — вспомогательная точка; trim_1 — первая точка вычленения; ENDPNT — конечная точка; trim_2 — вторая точка вычленения; api_circular_arc — построенная дуга окружности

Рисунок А.25 — Функция Arc_Rad_3_Pnt

Внутренние ссылки: 6.1.9, 6.1.12, 6.1.12.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
105	Попытка создания вырожденного направления в процессе создания сущности	106	Попытка создания вырожденной локальной координатной системы *axis2_placement* в процессе создания сущности
112	Попытка создания дуги длиной меньше EPS	115	Заданные сущности идентичны
116	Заданные точки линейно зависимы	127	Геометрическое построение нецелесообразно
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.3.1.2.5 Построение дуги окружности по радиусу, двум точкам и сущности axis2_placement

Имя функции:

Arc_Rad_2_Pnt_A2p

Уровень интерфейса:	1
Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	RAD	D	Радиус дуги окружности *api_circular_arc*	(EPS ≤ RAD ≤ MAX)
Ввод	PNTNM1	N	Имя первой декартовой точки *cartesian_point*	pnt
Ввод	PNTNM2	N	Имя второй декартовой точки *cartesian_point*	pnt
Ввод	A2PNAM	N	Имя сущности *axis2_placement*	A2p
Ввод	SENSE	E	Знак направления обхода кривой	[TRUE, FALSE]
Вывод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
out	NAME	N	Имя созданной сущности *api_circular_arc*	arc

Привязка языка FORTRAN:
NAME = ARC_RAD_2_PNT_A2P (RAD, PNTNM1, PNTNM2, A2PNAM, SENSE, KFIX)

Результат использования функции

Функция создает дугу окружности *api_circular_arc* по радиусу (RAD), двум декартовым точкам *cartesian_point*, локальной координатной системе *axis2_placement* (A2PNAM) и заданному положительному направлению обхода контура, соответствующему значению флажка (SENSE) вновь созданной дуги окружности *api_circular_arc*. Построение производится в совокупности с сущностью *circle* в качестве базовой кривой *basis_curve* и вместе с начальной и конечной точками дуги, неявно определенными двумя точками PNTNM1 и PNTNM2 соответственно.

Начало локальной координатной системы A2PNAM дублируется как точка *a2p1* и имеет нулевой стиль *null_style*. Пусть точки *P1* и *P2* являются синонимами PNTNM1 и PNTNM2 декартовых точек соответственно. Затем:

- создается экземпляр *c* окружности с центром *a2p1* и радиусом RAD. Данная окружность имеет нулевой стиль;
- создается экземпляр *p3* декартовой точки *cartesian_point*, координаты которой вычисляются по декартовым координатам *cartesian_coordinates* начала координат *a2p1*. Полученная декартова точка является центром окружности и имеет нулевой стиль;
- создается экземпляр *d1* направления *direction* с компонентами *direction_ratio*, определенными как разность *P1 — p3*. Указанное направление имеет нулевой стиль;
- создается экземпляр вектора *v1* с направлением *d1* и модулем ||*P1 — p3*||. Данный вектор имеет нулевой стиль;
- создается экземпляр линии *l1* с точкой *pnt p3* и направлением *dir v1*. Данная линия имеет нулевой стиль;
- создается экземпляр *d2* направления *direction* с компонентами *direction_ratio*, определенными разностью *P2 — p3*. Указанное направление имеет нулевой стиль;
- создается экземпляр вектора *v2* с направлением *d2* и модулем ||*P2 — p3*||. Данный вектор имеет нулевой стиль;
- создается экземпляр линии *l2* с точкой *pnt p3* и направлением *dir v2*. Данная линия имеет нулевой стиль;
- создается экземпляр *p4* декартовой точки, координаты которой равны координатам точки пересечения линии *l1* с окружностью *c* для положительного направления линии *l1*. Полученная декартова точка имеет нулевой стиль;
- создается экземпляр *p5* декартовой точки, координаты которой есть координаты точки пересечения линии *l2* с окружностью *c* для положительного направления линии *l2*. Полученная декартова точка имеет нулевой стиль;
- создается экземпляр сущности *api_circular_arc* с дугой окружности *c* как базовой кривой *basis_curve*, точками *p4* и *p5* как точками вычленения *trim_1* и *trim_2* соответственно. Значение атрибута обхода кривой *sense_agreement* равно значению параметра SENSE, значение атрибута главного представления *master_representation* зависит от реализации. Полученная сущность *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной дуги окружности *api_circular_arc* интерфейса прикладного программирования.

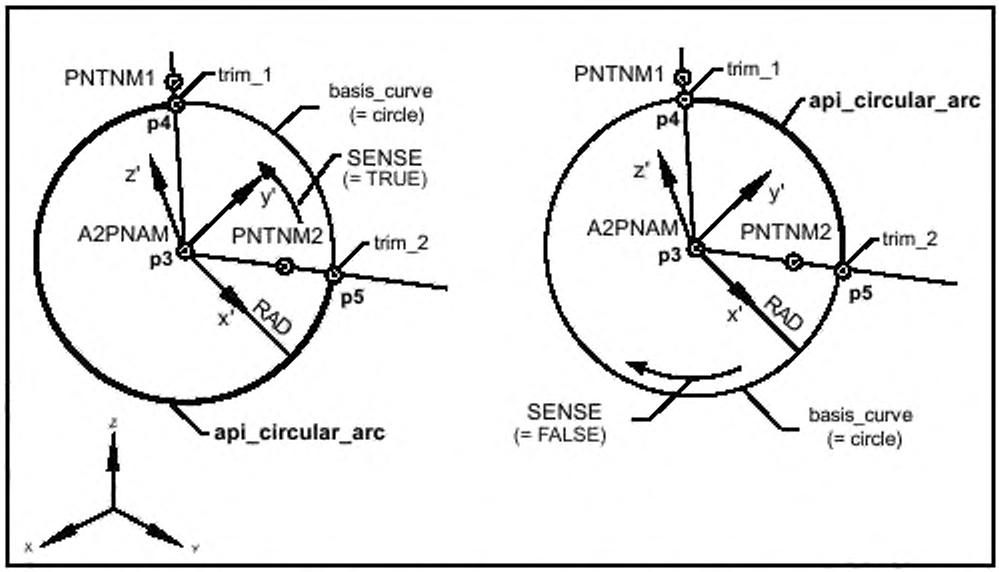
Расстояние между центром дуги окружности *api_circular_arc* и двумя заданными декартовыми точками должно быть не меньше допуска EPS. Кроме того, расстояние между двумя заданными декартовыми точками должно быть не меньше допуска EPS. Значение радиуса RAD лежит в диапазоне [EPS, MAX] и измеряется в единицах длины *OVC_length_unit*. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Если два неявно заданных направления *d1* и *d2* идентичны в диапазоне [*ZERO_value*, EPS], то обе точки вычленения *trim_1* и *trim_2* также идентичны. При этом созданная дуга образует полную окружность. В этом случае интерфейс гарантирует замкнутость созданной сущности *api_circular_arc* (декартовы координаты точки вычленения *trim_1* равны декартовым координатам точки вычленения *trim_2*).

2 Сущность *api_circular_arc* создается, если длина дуги от точки *trim_1* до точки *trim_2*, совместимая со значением атрибута *sense_agreement*, не меньше допуска EPS. В противном случае возникает ошибка.

3 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса не менее 2), то точки *P1* и *P2* должны лежать в плоскости (Oxy) локальной координатной системы (A2PNAM).



PNTNM1 — первая заданная точка; trim_1 — первая точка вычленения; basis_curve (= circle) — базовая кривая (окружность); api_circular_arc — построенная дуга окружности: SENSE (= TRUE) — обход кривой в положительном направлении; A2PNAM — начало локальной координатной системы, PNTNM2 — вторая заданная точка; trim_2 — вторая точка вычленения, RAD — радиус; SENSE (= FALSE) — обход кривой в отрицательном направлении

Рисунок А.26 — Функция Arc_Rad_2_Pnt_A2p

Внутренние ссылки: 6.1.9, 6.1.12, 6.1.12.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
105	Попытка создания вырожденного направления в процессе создания сущности	112	Попытка создания дуги длиной меньше EPS
119	Заданные сущности не лежат в одной плоскости	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

А.5.3.1.2.6 Построение дуги окружности как сопряжение двух сущностей

Имя функции:

Arc_Fillet_2_Ent

Уровень интерфейса:	1
Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNM1	N	Имя первой сущности	basic
Ввод	ENTNM1	N	Имя второй сущности	basic
Ввод	RAD	D	Радиус сопряжения	(EPS ≤ RAD ≤ MAX)
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_circular_arc*	arc

Привязка языка FORTRAN:
NAME = ARC_FILLET_2_ENT (ENTNM1, ENTNM2, RAD, KFIX)

Результат использования функции

Функция создает дугу окружности *api_circular_arc* как сопряжение двух базовых сущностей с радиусом RAD. Данные сущности могут пересекаться или не пересекаться. Указанные сущности с именами ENTNM1 и ENTNM2 вычленяются при построении, при этом их имена не изменяются. Они продолжают оставаться во временной базе данных до окончания построения. Вновь созданная сущность *api_circular_arc* начинается в новом конце первой сущности ENTNM1 (точка вычленения *trim_2* сущности ENTNM1) и заканчивается в новом начале второй сущности ENTNM2 (точка вычленения *trim_1* сущности ENTNM2). Направление обхода дуги окружности *api_circular_arc* совместимо с направлениями обхода указанных сущностей.

Интерфейс выполняет нижеследующие вычисления:

- все возможные геометрические варианты касания окружностей cir_i сущности *circle* с радиусом RAD, расположенным между заданными сущностями ENTNM1 и ENTNM2, рассчитываются виртуально. При этом центрами указанных окружностей являются экземпляры $pc1_i$ сущностей *cartesian_point* для точек касания ENTNM1 (как экземпляры $pt1_i$ сущностей *cartesian_point*) и для точек касания ENTNM2 (как экземпляры $pt2_i$ сущностей *cartesian_point*). Указанные точки имеют номер i (i = 1, ..., n), где n равно максимально возможному числу решений. Все указанные созданные экземпляры имеют нулевой стиль *null_style*.

Если в процессе вычислений происходит сбой, то геометрическое построение нецелесообразно (n = 0) или для касания дуг задан слишком большой (слишком маленький) радиус RAD. В результате:

- возникает ошибка;
- никакие сущности не создаются и никакие изменения существующих сущностей не производятся;
- функция возвращает нулевое имя элемента.

Если первая заданная базовая сущность ENTNM1 является экземпляром сущности *api_line*, то:

- создается n экземпляров сущности *axis2_placements* $a2p_i$ с началом локальной координатной системы $pc1_i$ и направлениями обеих локальных осей, расположенных в одной плоскости. Все указанные экземпляры имеют нулевой стиль;
- создается n экземпляров окружностей c_i сущности *circle* с центрами $a2p_i$ и радиусом RAD. Все указанные экземпляры имеют нулевой стиль;
- создается n экземпляров a_i сущности *api_circular_arc* в виде окружности как базовой кривой *basis_curve* и точками $pt1_i$ как точками вычленения *trim_1* и точками $pt2_i$ как точками вычленения *trim_2*. Значение атрибута направления обхода кривой *sense_agreement* равно «true», если направление обхода задается вектором *trim_2 — trim_1* сущности *api_line* (с именем ENTNM1), направление которого совпадает с направлением касательной в точке $pt1_i$ на базовой кривой c_i сущности *basis_curve*. В противном случае значение параметра SENSE равно «false». Значение атрибута главного представления *master_representation* зависит от реализации. Все указанные экземпляры имеют нулевой стиль.

Далее начинается процесс выбора единственного решения для вновь созданной сущности *api_circular_arc*. Выбирается экзмпляр a_i, для которого:

- направление касательной в точке $pt2_i$ на дуге окружности a_i сущности *api_circular_arc* совпадает с направлением обхода второй сущности ENTNM2 (с учетом значения атрибута обхода контура *sense_agreement_flag*);
- дуга окружности a_i сущности *api_circular_arc* имеет наименьший центральный угол;
- вычисленное расстояние между точкой вычленения *trim_1* первой сущности ENTNM1 и точкой вычленения *trim_1* вновь созданной дуги окружности a_i сущности *api_circular_arc* имеет наименьшую длину;
- выбранная сущность *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной дуги окружности *api_circular_arc*.

Если первая заданная базовая сущность ENTNM1 является экземпляром *api_circular_arc*, то:

- создается n экземпляров локальных координатных систем $a2p_i$ для сущностей *axis2_placement* с началом координат $pc1_i$ и направлением осей, определенным по положению базовой кривой для сущности ENTNM1. Все указанные экземпляры имеют нулевой стиль;
- создается n экземпляров окружностей c_i с центрами, определенными локальной координатной системой $a2p_i$, и радиусом RAD. Все указанные экземпляры имеют нулевой стиль;
- создается n экземпляров a_i сущности *api_circular_arc* дуги окружности как базовой кривой с точками $pt1_i$ как точками вычленения *trim_1* и точками $pt2_i$ как точками вычленения *trim_2*. Направление дуги вычисляется как разность $pt1_i$ — $pc1_i$, совпадает с направлением разности $pt1_i$ — *ENTNM1.basis_curve.position.location*. При этом направление обхода полученной дуги окружности *sense_agreement* совпадает с направлением обхода *sense_agreement* дуги окружности *api_circular_arc* сущности ENTNM1. В противном случае значение атрибута *sense_agreement* противоположно направлению обхода *sense_agreement* дуги окружности *api_circular_arc* сущности ENTNM1. Значение атрибута главного представления *master_representation* зависит от реализации. Все указанные экземпляры имеют нулевой стиль.

Далее начинается процесс выбора единственного решения для вновь созданной дуги окружности *api_circular_arc*. Выбирается один экземпляр a_i, для которого:

- направление касательной в точке $pt2_i$ дуги окружности a_i сущности *api_circular_arc* совпадает с направлением обхода второй сущности ENTNM2 (с учетом значения атрибута направления обхода *sense_agreement_flag*);
- дуга окружности a_i сущности *api_circular_arc* имеет наименьший центральный угол;
- вычисленная длина дуги окружности между точкой вычленения *trim_1* первой сущности ENTNM1 и точкой вычленения *trim_1* дуги окружности a_i сущности *api_circular_arc* (измеренная в направлении из точки вычленения *trim_1* в точку вычленения *trim_2* сущности ENTNM1) имеет наименьшую длину;
- выбранная дуга окружности *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя созданной дуги окружности *api_circular_arc* интерфейса прикладного программирования.

Заданные экземпляры ENTNM1 определяются повторно (с точкой вычленения *trim_2*, равной точке вычленения *trim_1*) для выбранной сущности *api_circular_arc*. Заданные экземпляры ENTNM2 определяются повторно (с точкой вычленения *trim_1*, равной точке вычленения *trim_2*) для выбранной сущности *api_circular_arc*.

Значение радиуса RAD лежит в диапазоне [EPS, MAX] и измеряется в единицах длины *OVC_length_unit*. При возникновении ошибки новая сущность не создается, существующие сущности не изменяются. Функция возвращает нулевое имя элемента.

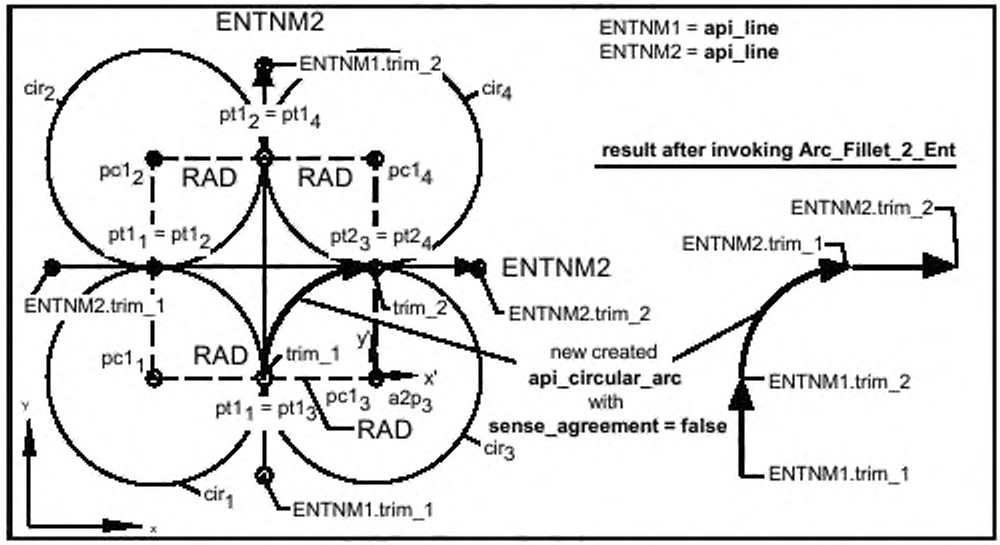
Примечания

1 Параллельные прямые *api_line* или концентрические окружности *api_circular_arc* не могут использоваться в качестве сущностей ENTNM1 и ENTNM2.

2 Созданная дуга окружности *api_circular_arc* не может быть больше полуокружности.

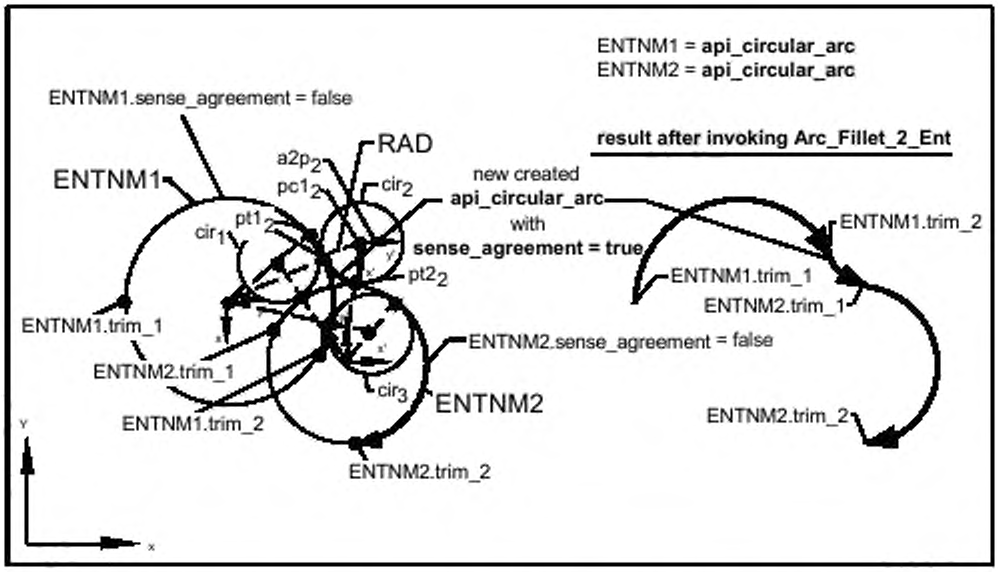
3 Сущность *api_circular_arc* создается, если длина дуги окружности, проведенной из точки вычленения *trim_1* в точку вычленения *trim_2* и совместимой со значением атрибута направления обхода кривой *sense_agreement*, не меньше допуска EPS. При этом длины сегментов обеих заданных сущностей ENTNM1 и ENTNM2 (после их вычленения) больше допуска EPS. В противном случае возникает ошибка.

4 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса не менее 2), то обе заданные сущности ENTNM1 и ENTNM2 должны лежать в одной плоскости.



ENTNM2 — вторая сущность сопряжения; ENTNM1 = api_line — имя сущности 1 api_line; ENTNM2 = api_line — имя сущности два api_line; ENTNM1.trim_2 — точка вычленения 2 первой сущности сопряжения, cir_2 — окружность cir_2; cir_4 — окружность cir_4; $pt1_2$ — точка $pt1_2$, $pt1_4$ — точка $pt1_4$; result after invoking Arc_Fillet_2_Ent — результат срабатывания функции Arc_Fillet_2_Ent (увеличено); $pc1_2$ — центр окружности $pc1_2$, RAD — радиус; $pc1_4$ — центр окружности $pc1_4$, ENTNM2.trim_2 — точка вычленения 2 второй сущности сопряжения; $pt1_1$ — точка $pt1_1$; $pt2_3$ — точка $pt2_3$; $pt2_4$ — точка $pt2_4$, ENTNM2.trim_1 — точка вычленения 1 второй сущности сопряжения; trim_2 — точка вычленения 2; $pc1_1$ — центр окружности $pc1_1$; trim_1 — точка отрезка 1; new created api_circular_arc with sense_agreement = false — вновь созданная дуга окружности с отрицательным направлением обхода; $pc1_3$ — центр окружности $pc1_3$; $a2p_3$ — положение $a2p_3$; $pt1_3$ — точка $pt1_3$; cir_3 — окружность cir_3; ENTNM1.trim_1 — точка вычленения 1 первой сущности сопряжения, cir_1 — окружность cir_1

Рисунок А.27 — Функция Arc_Fillet_2_Ent (сопряжение двух прямых)



ENTNM1 = api_circular_arc — имя сущности 1 api_circular_arc; ENTNM2 = api_circular_arc — имя сущности 2 api_circular_arc; ENTNM1.sense_agreement = false — первая исходная сущность с отрицательным направлением обхода: RAD — радиус; result after invoking Arc_Fillet_2_Ent — результат срабатывания функции Arc_Fillet_2_Ent (увеличено); $a2p_2$ — положение $a2p_2$, new created api_circular_arc with sense_agreement = true — вновь созданная дуга окружности с отрицательным направлением обхода, ENTNM1 — первая сущность сопряжения, $pc1_2$ — центр окружности pc12, cir_2 — окружность cir2; $pt1_2$ — точка $pt1_2$; ENTNM1.trim_2 — точка вычленения 2 первой сущности сопряжения, cir_1 — окружность cir_1; $pt2_2$ — точка $pt2_2$; ENTNM1.trim_1 — точка вычленения 1 первой сущности сопряжения, ENTNM2.tnm_1 — точка вычленения 1 второй сущности сопряжения; ENTNM2.sense_agreement = false — вторая исходная сущность с отрицательным направлением обхода; cir_3 — окружность cir_3; ENTNM2 — вторая сущность сопряжения; ENTNM2.trim_2 — точка вычленения 2 второй сущности сопряжения

Рисунок А.28 — Функция Arc_Fillet_2_Ent (сопряжение дуг двух окружностей)

Внутренние ссылки: 6.1.9, 6.1.12, 6.1.12.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
106	Попытка создания вырожденной локальной координатной системы *axis2_placement* в процессе создания сущности	108	Попытка создания вырожденной базовой кривой в процессе создания сущности
112	Попытка создания дуги длиной меньше EPS	115	Заданные сущности являются идентичными
118	Заданные сущности кривых являются параллельными (концентрическими)	119	Заданные сущности не лежат в одной плоскости
121	Радиус слишком большой (маленький)	127	Геометрическое построение нецелесообразно
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.3.1.2.7 Построение дуги окружности, касательной к двум сущностям

Имя функции:	Уровень интерфейса:	1
Arc_Tangential_2_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNM1	N	Имя первой сущности	basic
Ввод	ENTNM1	N	Имя второй сущности	basic
Ввод	RAD	D	Радиус касания сущности *api_circular_arc*	(EPS ≤ RAD ≤ MAX)
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_circular_arc*	arc

Привязка языка FORTRAN:
NAME = ARC_TANGENTIAL_2_ENT (ENTNM1, ENTNM2, RAD, KFIX)

Результат использования функции

Функция создает дугу окружности *api_circular_arc* с радиусом RAD, касательную к двум базовым сущностям. Заданные сущности ENTNM1 и ENTNM2 могут пересекаться или не пересекаться. Направление вновь созданной сущности *api_circular_arc* совместимо с направлениями заданных сущностей.

Интерфейс выполняет нижеследующие вычисления:

- сущности *E1* и *E2* рассматриваются как синонимы заданных сущностей ENTNM1 и ENTNM2 соответственно;

- виртуально вычисляются все геометрически возможные касательные к сущностям *E1* и *E2* окружностей cir_i сущности *circle* с радиусом RAD. Центры данных окружностей $pc1_j$ являются экземплярами сущности *cartesian_point*. Точки касания базовой кривой *basis_curve E1* являются экземплярами $pt1_j$ сущности *cartesian_point*. Точки касания базовой кривой *basis_curve E2* являются экземплярами $pt2_j$ сущности *cartesian_point*. Номера рассматриваемых точек *i* изменяются в пределах $i = 1, \ldots, n$, где *n* — максимальное количество возможных решений. Все указанные экземпляры имеют нулевой стиль *null_style*.

Если при вычислениях возникает сбой, то геометрическое построение нецелесообразно ($n = 0$): заданный радиус RAD касания дуг слишком велик или слишком мал. Затем:

- диагностируется ошибка;

- функция возвращает нулевое имя элемента.

Если первая базовая сущность *E1* является экземпляром прямой *api_line*, то:

- создается экземпляр $a2p_i$ локальной координатной системы *axis2_placements* с началом координат в точке $pc1_j$ и локальными осями, лежащими в одной общей плоскости. Все указанные экземпляры имеют нулевой стиль;

- создается *n* экземпляров c_i окружностей *circle* с центрами в точках $a2p_i$ и радиусом RAD. Все указанные экземпляры имеют нулевой стиль;

- создается *n* экземпляров дуг окружностей a_i для сущностей *api_circular_arc*. При этом сущности c_i являются базовыми кривыми *basis_curve*, точки $pt1_j$ являются точками вычленения *trim_1*, а точки $pt2_j$ являются точками вычленения *trim_2*. Значение атрибута направления обхода контура *sense_agreement* равно «true», если направление вектора *trim_2 — trim_1* вдоль прямой *api_line (E1)* совпадает с направлением касательной в точке $pt1_i$ базовой кривой *basis_curve* (окружности c_i). В противном случае значение атрибута обхода контура равно «false». Значение атрибута главного представления *master_representation* зависит от реализации. Все указанные экземпляры имеют нулевой стиль.

Далее из указанного множества возможных решений выбирается единственная дуга окружности *api_circular_arc*. Таким решением является экземпляр a_i, для которого:

- направление касательной в точке $pt2_i$ на дуге окружности a_i сущности *api_circular_arc* совпадает с направлением второй сущности *E2* (с учетом значения атрибута *sense_agreement_flag*);

- дуга окружности a_i сущности *api_circular_arc* имеет наименьший центральный угол;

- точки вычленения *trim_1* и *trim_2* дуги окружности a_i сущности *api_circular_arc* лежат внутри параметрического диапазона [*trim_1*, *trim_2*] для обеих заданных сущностей *E1* и *E2*;

- вычисленное расстояние между точкой вычленения *trim_1* первой сущности *E1* и точкой вычленения *trim_1* вновь созданной дуги окружности a_i сущности *api_circular_arc* является наименьшим;

- выбранная дуга окружности *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной сущности *api_circular_arc*.

Если первой базовой сущностью *E1* является экземпляр дуги окружности *api_circular_arc*, то:

- создается *n* экземпляров $a2p_i$ локальной координатной системы *axis2_placements* с началом координат в точке $pc1_i$ и осями координат, соответствующими базовой кривой *basis_curve* для сущности *E1*. Все указанные экземпляры имеют нулевой стиль;

- создается *n* экземпляров c_i сущности *circle* с центрами в точках $a2p_i$ и радиусом RAD. Все указанные экземпляры имеют нулевой стиль;

- создается *n* экземпляров дуг окружностей a_i сущности *api_circular_arc* с кривыми c_i в качестве базовых кривых, точками $pt1_i$ в качестве точек вычленения *trim_1* и точками $pt2_i$ в качестве точек вычленения *trim_2*. Если направление вектора $pt1_i$ — $pc1_i$ совпадает с направлением вектора $pt1_i$ — *ENTNM1.basis_curve.position.location*, то значение атрибута направления обхода кривой *sense_agreement* равно значению соответствующего атрибута *sense_agreement* для дуги окружности *api_circular_arc* (сущности *E1*). В противном случае направление обхода *sense_agreement* противоположно направлению обхода *sense_agreement* дуги окружности *api_circular_arc* (сущности *E1*). Значение атрибута главного представления *master_representation* зависит от реализации. Все указанные экземпляры имеют нулевой стиль.

Далее из множества возможных решений выбирается только одно решение для вновь созданной дуги окружности *api_circular_arc*. Выбирается такой экземпляр a_i, что:

- направление касательной в точке $pt2_i$ дуги окружности a_i сущности *api_circular_arc* совпадает с направлением второй сущности *E2* (с учетом значения атрибута *sense_agreement_flag*);

- дуга окружности a_i сущности *api_circular_arc* имеет наименьший центральный угол;

- точки вычленения *trim_1* и *trim_2* дуги окружности a_i сущности *api_circular_arc* лежат внутри параметрического диапазона [*trim_1*, *trim_2*] для обеих заданных сущностей *E1* и *E2*;

- вычисленная дуга окружности между точкой вычленения *trim_1* первой сущности *E1* и точкой вычленения *trim_1* дуги окружности a_i сущности *api_circular_arc* (измеренные от точки *trim_1* к точке *trim_2* сущности *E1*) имеет наименьшую длину;

- выбранная дуга окружности *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса.

Функция возвращает имя полученной дуги окружности *api_circular_arc* интерфейса прикладного программирования.

Значение радиуса RAD лежит в диапазоне [EPS, MAX]. Оно измеряется в единицах длины *OVC_length_unit*. При возникновении ошибки сущность не создается, существующие сущности не изменяются. Функция возвращает нулевое имя элемента.

Примечания

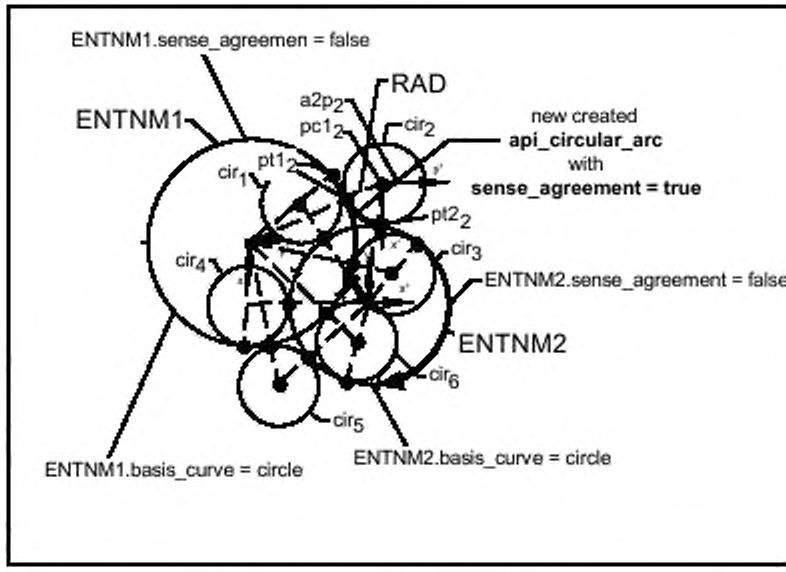
1 Параллельные прямые *api_line* или концентрические дуги окружностей *api_circular_arc* не допускаются при задании сущностей *E1* и *E2*.

2 Созданная дуга окружности *api_circular_arc* не может быть больше полуокружности.

3 Если значение расстояния между выбранными точками касания (используемыми для создания новой дуги окружности *api_circular_arc*) и одной из точек вычленения (*trim_1* или *trim_2*) соответствующей дуги окружности *api_circular_arc* лежит в диапазоне [*ZERO_value*, EPS], то ошибок нет и координаты точек вычленения используются вместо вычисленных координат.

4 Дуга окружности *api_circular_arc* создается, если длина дуги *trim_1* — *trim_2*, совместимая со значением атрибута *sense_agreement*, не меньше допуска EPS.

5 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса не менее 2), то обе заданные сущности *E1* и *E2* должны лежать в одной плоскости.



ENTNM1.sense_agreement = false — исходная первая сущность ENTNM1 (окружность) с отрицательным направлением обхода; RAD — радиус; $a2p_2$ — локальная координатная система. new created api_circular_arc with sense_agreement = true — вновь созданная дуга окружности с положительным направлением обхода; ENTNM1 — первая исходная окружность ENTNM1, $pc1_2$ — центр окружности сопряжения $pc1_2$; cir_2 — окружность cir_2, $pt1_2$ — точка $pt1_2$; cir_1 — окружность cir_1, $pt2_2$ — точка $pt2_2$; cir_3 — окружность cir_3; cir_4 — окружность cir_4; ENTNM2.sense_agreement = false — исходная вторая сущность ENTNM1 (окружность) с отрицательным направлением обхода; ENTNM2 — вторая исходная окружность ENTNM2: cir_6 — окружность cir_6; cir_5 — окружность cir_5; ENTNM2.basis_curve = circle — базовая кривая (окружность) для второй исходной сущности, ENTNM1. basis_curve = circle — базовая кривая (окружность) для первой исходной сущности

Рисунок А.29 — Функция Arc_Tangential_2_Ent

Внутренние ссылки: 6.1.9, 6.1.12, 6.1.12.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
106	Попытка создания вырожденной локальной координатной системы *axis2_placement* в процессе создания сущности	110	Попытка создания точки вне параметрического диапазона сущности кривой
112	Попытка создания дуги длиной меньше EPS	115	Заданные сущности являются идентичными
118	Заданные сущности кривой являются параллельными (концентрическими)	119	Заданные сущности не лежат в одной плоскости
121	Радиус является слишком большим (маленьким)	127	Геометрическое построение нецелесообразно
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.3.1.2.8 Построение дуги окружности по радиусу и двум сущностям

Имя функции:	Уровень интерфейса:	1
Arc_Rad_2_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	RAD	D	Радиус касания дуги окружности *api_circular_arc*	(EPS ≤ RAD ≤ MAX)
Ввод	ENTNM1	N	Имя первой исходной сущности	basic, pnt
Ввод	ENTNM1	N	Имя второй исходной сущности	basic, pnt
Ввод	IN1	E	Положение новой сущности относительно первой исходной сущности ENTNM1	[TRUE, FALSE]
Ввод	IN2	E	Положение новой сущности относительно второй исходной сущности ENTNM2	[TRUE, FALSE]
Ввод	MINLEN	E	Выбор длины дуги	[TRUE, FALSE]
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_circular_arc*	arc

Привязка языка FORTRAN:
NAME = ARC_RAD_2_ENT (RAD, ENTNM1, ENTNM2, IN1, IN2, MINLEN, KFIX)

Результат использования функции

К рассмотрению принимаются сущности *E1* и *E2* — синонимы исходных сущностей ENTNM1 и ENTNM2.

Функция создает дугу окружности *api_circular_arc*, определенную двумя ограничениями по двум исходным сущностям *E1* и *E2*. Указанные сущности могут быть декартовыми точками *cartesian_point*, прямыми *api_line* или дугами окружностей *api_circular_arc*. При этом сразу обе исходные сущности не могут быть декартовыми точками. Если одной из указанных сущностей является декартова точка, то эта точка должна лежать на результирующей дуге окружности *api_circular_arc*. В других случаях исходными сущностями могут быть сущность *api_line* или сущность *api_circular_arc*. При этом результирующая кривая должна касаться базовых кривых *basis_curve* (соответственно прямой или окружности) указанных сущностей. В данном случае имеет место «правило полиспаста» (одинаковое направление обхода отрезка кривой *trimmed_curve* в обеих точках касания). Три булева параметра MINLEN, IN1 и IN2 используются для разрешения неоднозначности. Эти параметры задают длину дуги и положение требуемого решения относительно (внутри или снаружи) каждого из возможных параметров сущности *api_circular_arc*.

Прежде всего плоскость геометрических построений ассоциируется с некоторым ссылочным направлением *R* сущности *ref_direction* и (в 3D-случае) с ортогональным направлением *z* сущности *axis_direction* (в 2D-случае направление *z* не существует). Указанные направления используются для параметризации требуемой дуги окружности *api_circular_arc*. Вычисления направлений производятся в следующем порядке:

- если текущий открытый вид определен как 2D-вид, то направление *R* является ссылочным направлением *ref_direction* базовой координатной системы OVC;

- в случае применения 3D-вида, если сущность *E1* или сущность *E2* являются дугой окружности *api_circular_arc*, то пусть сущность *E* будет первой сущностью (первой дугой окружности) *api_circular_arc* перечня (*E1*, *E2*) в указанном списке.

Пусть направление *R* = *E.basis_curve.position.ref_direction* соответствует установленному ссылочному направлению, а направление *Z* = *E.basis_curve.position.axis* соответствует указанной оси;

- в случае 3D-вида, когда среди исходных сущностей нет дуг окружностей *api_circular_arc* и если сущности *E1* и *E2* являются декартовыми точками *cartesian_point*, идентифицируется ошибка (плоскость не может быть определена);

- в случае 3D-вида, когда среди исходных сущностей нет дуг окружностей *api_circular_arc* и есть по крайней мере одна прямая *api_line*, пусть прямая *L* будет первой прямой *api_line* из перечня сущностей (*E1*, *E2*) в указанном порядке и *F* — оставшаяся сущность. Пусть *O* будет начальной точкой прямой *L*. Если *F* — декартова точка, то

пусть *G* = *F*. В противном случае *F* является прямой линией *api_line*. Если ее начало не принадлежит базовой кривой *L* сущности *basis_curve*, то пусть точка *P* является начальной для сущности *F*. В противном случае *P* является его крайней точкой.

Тогда *R* = *L.dir.orientation*, а *Z* = *L.dir.orientation* × *OP* (где × — символ векторного произведения). Пусть *S* — множество дуг окружностей *A* для сущности *api_circular_arc*:

- если сущность *E1* является декартовой точкой, то *p1* — ее дубликат. В противном случае *p1* есть точка касания сущности *A* и базовой кривой для сущности *E1*;

- если сущность *E2* является декартовой точкой, то *p2* — ее дубликат. В противном случае *p2* есть точка касания сущности *A* и базовой кривой для сущности *E2*;

- точки *p1*, *p2* вычисляются интерфейсом и имеют нулевой стиль *null_style*;

- для каждого *i* в диапазоне [1...2]: если сущность E_i является дугой окружности *api_circular_arc*, то элемент *A* находится внутри E_i (если значение параметра IN_i равно «true»); в противном случае (если значение параметра IN_i равно «false») элемент *A* находится вне E_i. Если сущность E_i не является дугой окружности *api_circular_arc*, то параметр IN_i не используется для определения элемента *A*;

- точка *p1* является начальной точкой отрезка (элементом *A.trim_1*[1]), а точка *p2* является концом точки отрезка (элементом *A.trim_2*[1]);

- изложенные далее постулаты используются для формального задания интуитивно воспринимаемых условий «правила полиспаста» для ориентированных дуг, касательных к сущностям *E1* и *E2* (касательные векторы должны быть когерентными).

Условие *A.sense_agreement*:

- пусть *BA* является базовой кривой для элемента *A*;

- пусть BE_i является базовой кривой для сущности E_i, если данная сущность не является декартовой точкой *cartesian_point*;

- пусть P_i является точкой касания кривых *BA* и BE_i;

- пусть Va_i является вектором касания кривой *BA* в точке P_i.

- пусть вектор VA_i равен вектору Va_i, если значение атрибута направления обхода контура *A.sense_agreement* равно «true», в противном случае пусть он будет равен Va_i;

- пусть Ve_i является вектором касания кривой BE_i в точке P_i;

- пусть вектор VE_i равен вектору Ve_i, если значение атрибута обхода контура E_i.*sense_agreement* равно «true», и в противном случае равен вектору $-Ve_i$.

Тогда для каждого *i*, для которого сущность E_i не является декартовой точкой, векторы VA_i и VE_i имеют одинаковые направления.

Множество *S* содержит не более двух элементов. Если множество *S* содержит только один элемент, то пусть этим элементом будет сущность NAME. В противном случае если множество *S* содержит два элемента, то (если параметр MINLEN равен «true») сущность NAME из них та, которая имеет меньшую дугу. В противном случае (если параметр MINLEN равен «false») сущностью NAME является оставшаяся сущность.

Далее создается экземпляр сущности NAME с точкой вычленения *trim_1*, точкой вычленения *trim_2*, значением атрибута направления обхода кривой *sense_agreement* и заданным положением базовой кривой *basis_curve.position.location*. При этом ссылочное направление для базовой кривой *basis_curve.position.ref_direction* равно *R*, и (при необходимости для 3D-вида) заданное направление оси базовой кривой *basis_curve.position*.axis, равное *Z*.

Если геометрическое построение нецелесообразно, то возникает ошибка.

Сущность NAME.*master_representation* зависит от реализации.

Созданная дуга окружности *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной дуги окружности *api_circular_arc* интерфейса прикладного программирования.

При возникновении ошибки сущность не создается, значение ее имени равно 0.

Примечания

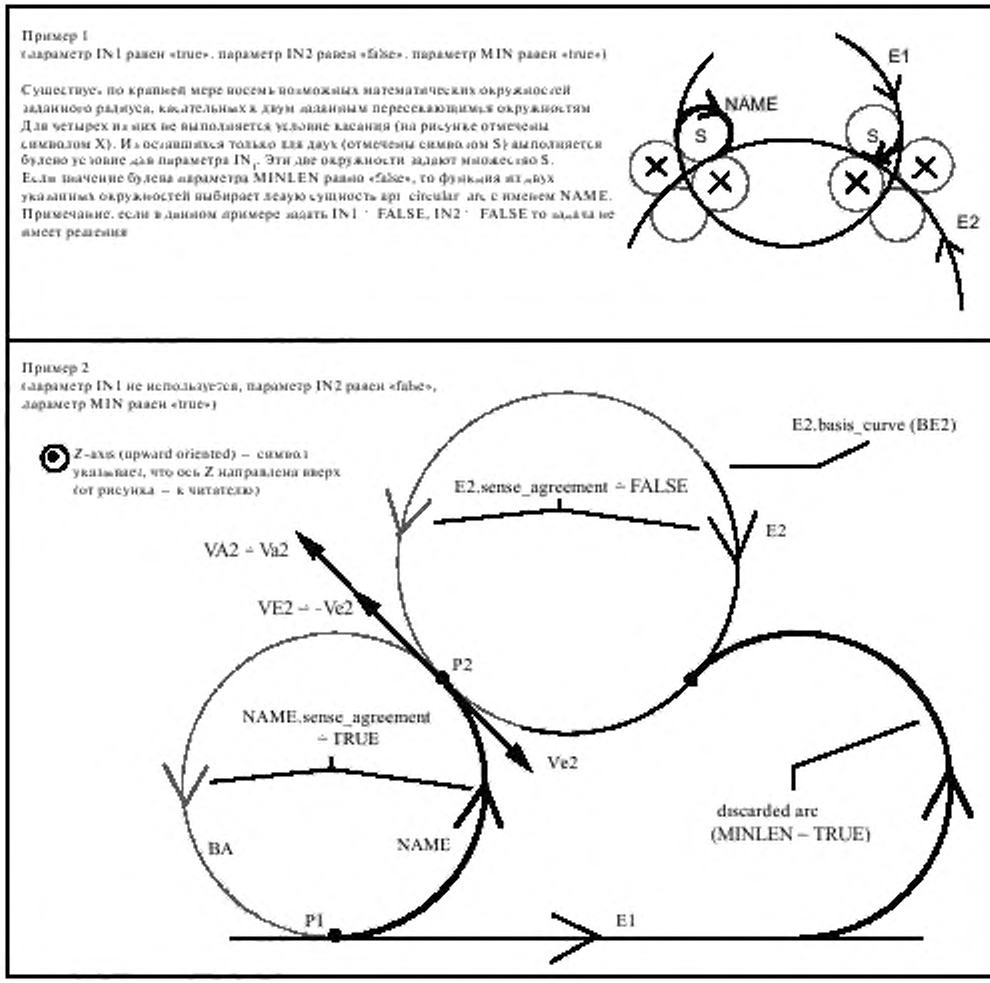
1 Если решения нет (принимая во внимание значение параметра IN_i), то возникает ошибка.

2 Параметр IN_i игнорируется, если соответствующие параметры $ENTNM_i$ не являются дугами окружностей *api_circular_arc*.

3 Параметр MINLEN игнорируется, если существует только одна дуга окружности *api_circular_arc*, вычисленная по заданному алгоритму.

4 Сущность *api_circular_arc* создается, если длина дуги, идущей из точки *trim_1* в точку *trim_2* и совместимой со значением атрибута *sense_agreement*, не меньше допуска EPS. В противном случае возникает ошибка.

5 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса не менее 2), то сущности *E1* и *E2* должны лежать в одной плоскости.



NAME — результирующая сущность; E2.basis_curve (BE2) — базовая кривая для исходной сущности E2 (BE2); E2.sense_agreement = FALSE — значение атрибута направления обхода сущности E2 равно «false»; NAME.sense_agreement = TRUE — значение атрибута обхода сущности равно «true»; discarded arc (MINLEN = TRUE) — исключенная дуга (значение параметра MINLEN равно «true»)

Рисунок А.30 — Функция Arc_Rad_2_Ent

Внутренние ссылки: 6.1.9.2, 6.1.12.1, 6.1.12.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
106	Попытка создания вырожденной локальной координатной системы *axis2_placement* в процессе создания сущности	110	Попытка создания точки вне параметрического диапазона сущности кривой
112	Попытка создания дуги, длиной меньше EPS	115	Заданные сущности являются идентичными

118	Заданные сущности кривой являются параллельными (концентрическими)	119	Заданные сущности не лежат в одной плоскости
121	Радиус слишком большой (маленький)	127	Геометрическое построение нецелесообразно
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.3.1.2.9 Построение дуги окружности по трем сущностям

Имя функции:	Уровень интерфейса:	1
Arc_3_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNM1	N	Имя первой сущности	basic, pnt
Ввод	ENTNM1	N	Имя второй сущности	basic, pnt
Ввод	ENTNM3	N	Имя третьей сущности	basic, pnt
Ввод	IN1	E	Положение новой сущности NAME относительно первой исходной сущности ENTNM1	[TRUE, FALSE]
Ввод	IN2	E	Положение новой сущности NAME относительно второй исходной сущности ENTNM2	[TRUE, FALSE]
Ввод	IN3	E	Положение новой сущности NAME относительно третьей исходной сущности ENTNM3	[TRUE, FALSE]
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_circular_arc*	arc

Привязка языка FORTRAN:
NAME = ARC_3_ENT (ENTNM1, ENTNM2, ENTNM3, IN1, IN2, IN3, KFIX)

Результат использования функции

К рассмотрению принимаются сущности *E1*, *E2* и *E3* как синонимы исходных сущностей ENTNM1, ENTNM2 и ENTNM3.

Функция создает дугу окружности *api_circular_arc* по трем ограничениям, задаваемым тремя исходными сущностями *E1*, *E2* и *E3*. Исходные сущности являются либо декартовыми точками *cartesian_point*, либо прямыми *api_line*, либо дугами окружности *api_circular_arc*. Если исходная сущность является декартовой точкой, то данная точка должна лежать на результирующей дуге окружности *api_circular_arc* с именем NAME. Если исходной сущностью является прямая или дуга окружности, то результирующая дуга окружности с именем NAME должна касаться базовой кривой *basis_curve* (прямой или окружности соответственно) исходной сущности. При этом выполняется «правило полиспаста» (обе отрезанные кривые *trimmed_curve* имеют одинаковое направление в точке касания). Функция имеет три булева параметра IN_i. Их задание устраняет неоднозначность решения путем задания относительного положения требуемой дуги окружности. Относительное положение (внутри или снаружи) задается для каждого параметра сущности *api_circular_arc*.

Если исходная сущность *E1* является декартовой точкой, то пусть точка *p1* является ее дубликатом. В противном случае точка *p1* является точкой касания результирующей сущности NAME и базовой кривой *basis_curve* для сущности *E1*.

Если исходная сущность *E2* является декартовой точкой, то пусть точка *p2* является ее дубликатом. В противном случае точка *p2* является точкой касания результирующей сущности NAME и базовой кривой для сущности *E2*.

Если исходная сущность *E3* является декартовой точкой, то пусть точка *p3* является ее дубликатом. В противном случае точка *p3* является точкой касания результирующей сущности NAME и базовой кривой для сущности *E3*.

Точки *p1*, *p2*, *p3* вычисляются интерфейсом, им назначается нулевой стиль *null_style*.

Требования к результирующей дуге окружности *api_circular_arc* с именем NAME:

- для каждого *i* в интервале [1...3], если сущность E_i является дугой окружности *api_circular_arc* и IN_i равно «true», то результирующая сущность NAME находится внутри E_i. В противном случае (когда IN_i равно «false»)

результирующая сущность NAME находится вне сущности E_i. Если же сущность E_i не является дугой окружности *api_circular_arc*, то параметр IN_i не используется для определения результирующей сущности NAME;

- точка *p1* является начальной точкой отрезка дуги NAME (обозначается *NAME.trim_1*[1]), точка *p3* является конечной точкой отрезка дуги (обозначается *NAME.trim_2*[1]), точка *p2* принадлежит параметрическому диапазону, определенному точками вычленения *p1*, *p3*. Расстояние от точки *p2* до точки *p1* (или до точки *p3*) может быть меньше 0 *ZERO_value*. Это означает идентичность двух точек;

- создается экземпляр положения результирующей сущности NAME (локальная координатная система *axis_2_placement*) с началом координат в вычисленном центре. При этом:

- если текущий открытый вид определен как 2D-вид, то ссылочным направлением базовой кривой *NAME.basis_curve.position.ref_directionis* является направление оси *X* OVC;

- если текущий открытый вид определен как 3D-вид, а одна из трех параметрических сущностей является дугой окружности *api_circular_arc*, то направления локальной координатной системы результирующей сущности *NAME.basis_curve.position* копируются по направлениям локальной координатной системы *axis_2_placement* первой дуги окружности *api_circular_arc* для перечня сущностей (*E1*, *E2*, *E3*) в указанном порядке;

- если текущий открытый вид определен как 3D-вид, когда исходные сущности *E1*, *E2* и *E3* являются декартовыми точками *cartesian_point*, то ссылочным направлением для результирующей сущности является направление *NAME.basis_curve.position.ref_direction= E1E2* (считается из начальной точки сущности *E1* до крайней точки сущности *E2*). Направлением оси результирующей сущности является направление *NAME.basis_curve.position.axis= E1E2* × *E1E3* (*где* × — символ векторного произведения). Указанные три точки не должны совпадать и лежать на одной прямой;

- если текущий открытый вид определен как 3D-вид, когда исходные сущности не являются дугами окружностей *api_circular_arc*, и по крайней мере одна сущность является прямой *api_line*, пусть сущность *L* является первой прямой из перечня (*E1*, *E2*, *E3*) в указанном порядке. Обозначим *F1* и *F2* как две оставшиеся сущности (в указанном порядке). Если данные сущности являются декартовыми точками, то точка *P* является первой из точек, не принадлежащих базовой кривой *basis_curve* сущности *L*. Если сущность *F1* или сущность *F2* являются прямыми *api_line*, то точка *G* является первой из точек, принадлежащих прямой *api_line* для сущностей (*F1*, *F2*) в указанном порядке. Пусть точка *P* является начальной точкой указанной прямой, если она не принадлежит базовой кривой *basis_curve* сущности *L* или в противном случае — конечной точкой. Пусть точка *O* является начальной точкой сущности *L*.

Тогда ссылочным направлением для результирующей дуги окружности является направление *NAME.basis_curve.position.ref_direction= L.dir.orientation*. Направлением координатной оси для результирующей сущности является направление *NAME.basis_curve.position.axis = L.dir.orientation* × *OP*.

Условия задания значения атрибута обхода кривой *sense_agreement*:

- пусть *BN* является базовой кривой для результирующей сущности NAME;
- пусть BE_i является базовой кривой для исходной сущности E_i, если E_i не является декартовой точкой;
- пусть P_i является точкой касания кривых *BN* и BE_i;
- пусть вектор Vn_i касается кривой *BN* в точке P_i;
- пусть вектор VN_i равен вектору Vn_i, если значение атрибута обхода контура *NAME.sense_agreement* равно «true», и в противном случае равен вектору $–Vn_i$;
- пусть вектор VE_i касается кривой BE_i в точке P_i;
- пусть вектор Ve_i равен вектору VE_i, если значение атрибута направления обхода контура E_i.*sense_agreement* равно «true», и в противном случае равен вектору $–VE_i$.

Тогда для каждого *i*, для которого исходная сущность E_i не является декартовой точкой, вектор Vn_i и вектор Ve_i имеют одинаковые направления. Если все три исходные сущности являются декартовыми точками, то значение атрибута *NAME.sense_agreement* для результирующей сущности равно «true».

Если геометрическое построение нецелесообразно, то возникает ошибка.

Значение радиуса результирующей сущности NAME лежит в диапазоне [EPS, MAX].

Значение атрибута *NAME.master_representation* результирующей сущности зависит от реализации.

Результирующая дуга окружности *api_circular_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной дуги окружности *api_circular_arc* интерфейса прикладного программирования.

Если возникает ошибка, то сущность не создается, значение ее имени равно 0.

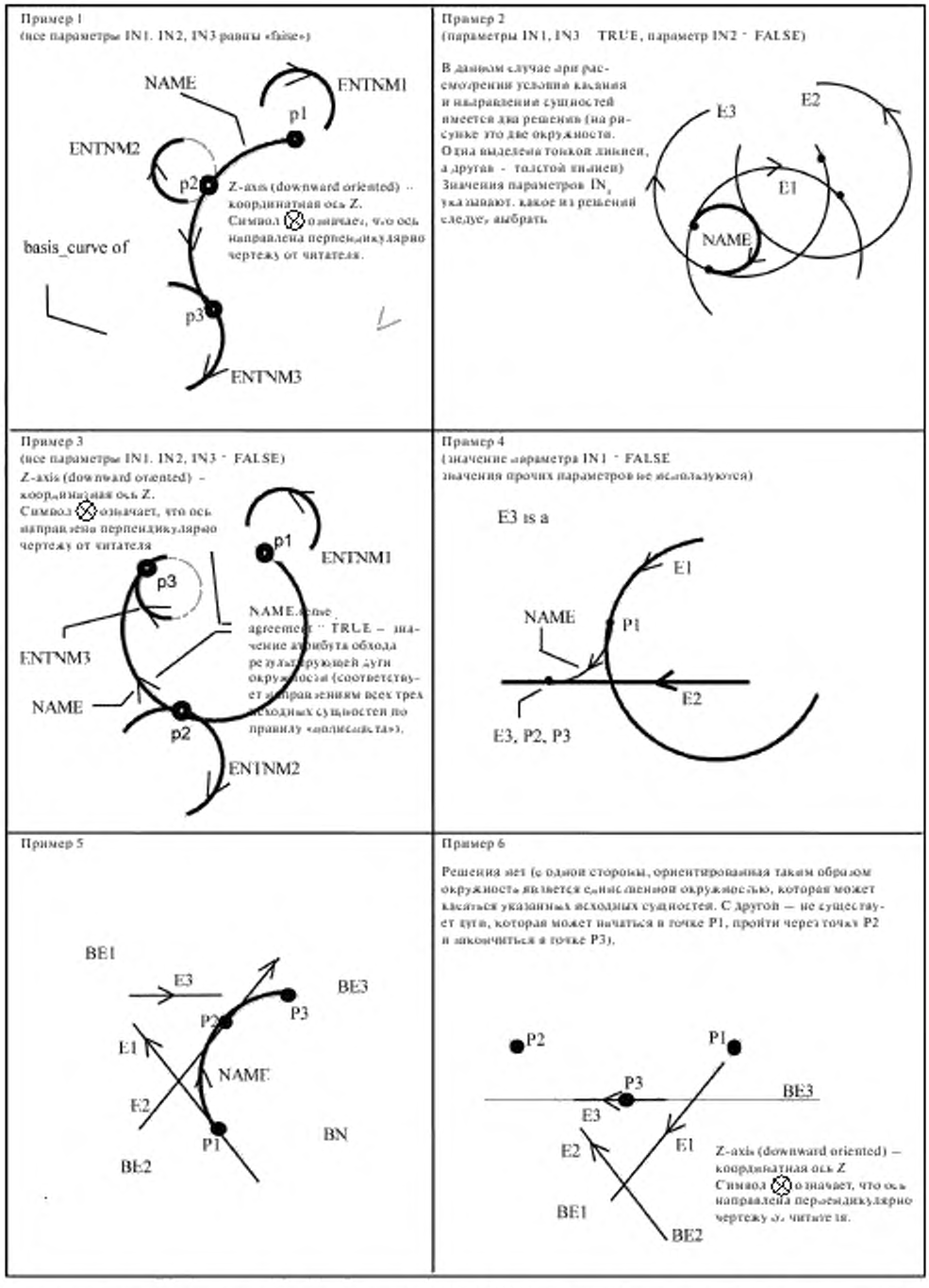
Примечания

1 Если решения нет (принимая во внимание значение параметра IN_i), то возникает ошибка.

2 Параметры IN_i игнорируются, если соответствующие параметры исходных сущностей $ENTNM_i$ не являются дугами окружности *api_circular_arc*.

3 Сущность *api_circular_arc* создается, если длина дуги, проведенной из точки вычленения *trim_1* в точку вычленения *trim_2*, совместимая с соответствующим значением атрибута направления обхода контура *sense_agreement*, не меньше допуска EPS. В противном случае возникает ошибка.

4 Если текущий открытый вид определен как 3D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса не менее 2), то исходные сущности *E1*, *E2* и *E3* должны лежать в одной плоскости.



NAME — результирующая сущность (дуга окружности) NAME, ENTNM1 — исходная сущность (дуга окружности) ENTNM1; ENTNM2 — исходная сущность (дуга окружности ENTNM2); ENTNM3 — исходная сущность (дуга окружности ENTNM3); BE1 — базовая кривая (прямая) для исходного примитива E1 (также прямая), BE3 — базовая кривая (прямая) для исходной сущности E3 (также прямая); BN – базовая кривая (прямая) имени; BE2 – базовая кривая (прямая) для исходной сущности E2 (также прямая); E3 is a — исходная сущность E3 является декартовой точкой

Рисунок А.31 — Функция Arc_3_Ent

Внутренние ссылки: 6.1.9.2, 6.1.12.1, 6.1.12.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
101	Попытка создания вырожденной сущности	106	Попытка создания вырожденной локальной координатной системы *axis2_placement* в процессе создания сущности
110	Попытка создания точки вне параметрического диапазона сущности кривой	112	Попытка создания дуги длиной меньше EPS
115	Заданные сущности являются идентичными	118	Заданные сущности кривой являются параллельными (концентрическими)
119	Заданные сущности не лежат в одной плоскости	127	Геометрическое построение нецелесообразно
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.3.2 Сущности дуг конических кривых

A.5.3.2.1 Эллипс и дуга эллипса (сущность api_elliptical_arc интерфейса прикладного программирования)

Построение эллипса по двум диаметрам и центру — Ellipse_Diameter_A2p

Построение дуги эллипса — Elc_Gen

A.5.3.2.1.1 Построение эллипса по двум диаметрам и центру

Имя функции:

Ellipse_2_Diameter_A2p

Уровень интерфейса:	1
Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	SEMI1	D	Большая полуось *semi_axis_1* эллипса *api_elliptical_arc*	(EPS ≤ SEMI1 ≤ MAX)
Ввод	SEMI2	D	Малая полуось *semi_axis_2* эллипса *api_elliptical_arc*	(EPS ≤ SEMI2 ≤ MAX)
Ввод	A2PNAM	N	Имя локальной координатной системы *axis2_placement*	a2p
Ввод	SENSE	E	Знак направления обхода кривой	[TRUE, FALSE]
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_elliptical_arc*	elc

Привязка языка FORTRAN:
NAME = ELLIPSE_2_DIAMETER_A2P (SEMI1, SEMI2, A2PNAM, SENSE, KFIX)

Результат использования функции

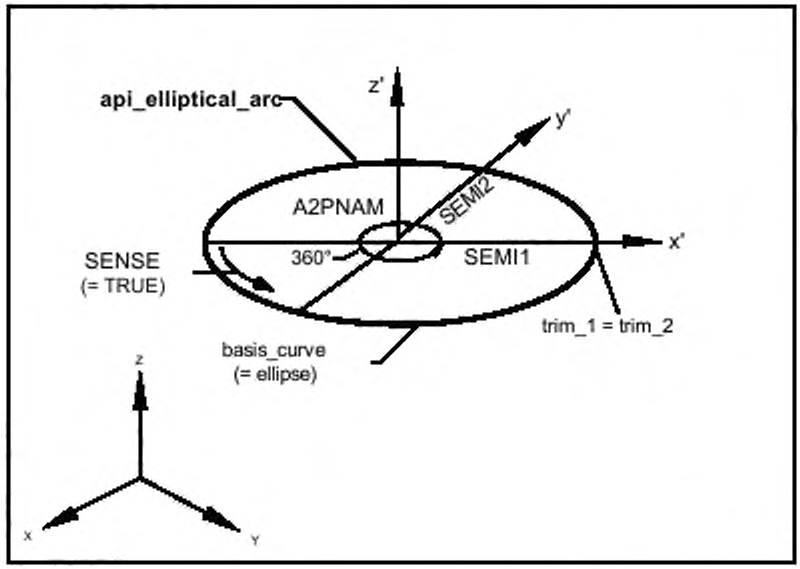
Функция создает полный эллипс как сущность *api_elliptical_arc* по большой полуоси эллипса (SEMI1), по малой полуоси эллипса (SEMI2), положению локальной координатной системы *axis2_placement* (A2PNAM) и значению атрибута направления обхода (SENSE) вновь созданной дуги *api_elliptical_arc* в совокупности с сущностью *ellipse* как базовой кривой *basis_curve*. Направление оси (Ox) локальной координатной системы A2PNAM определяет направление большой полуоси эллипса.

Сущность локальной координатной системы *axis2_placement* (A2PNAM) дублируется как сущность *a2p1*, имеющая нулевой стиль *null_style*. Затем:

- создается экземпляр *e* сущности *ellipse* с центром *position* локальной координатной системы *a2p1*, большая полуось *semi_axis_1* равна SEMI1 и малая полуось *semi_axis_2* равна SEMI2. Данный эллипс имеет нулевой стиль;
- создается экземпляр *api_elliptical_arc* с сущностью *e* в качестве базовой кривой *basis_curve*. Значение параметра кривой равно 0° в точке вычленения *trim_1*, значение параметра кривой равно 360° в точке вычленения *trim_2*. Значение атрибута направления обхода кривой *sense_agreement* равно SENSE, значение атрибута главного представления *master_representation* зависит от реализации. Полученная кривая *api_elliptical_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученного эллипса *api_elliptical_arc*.

Значения параметров MAJA и MINA должны лежать в диапазоне [EPS, MAX], они измеряются в единицах длины *OVC_length_unit*. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Интерфейс гарантирует замкнутость созданной сущности *api_elliptical_arc* (декартовы координаты точки вычленения *trim_1* равны декартовым координатам точки вычленения *trim_2*).



Api_elliptical_arc — результирующий эллипс; SEMI2 — малая полуось эллипса; A2PNAM — локальная координатная система; SENSE (= TRUE) - направление обхода -- положительное (против часовой стрелки), SEMI1 — большая полуось эллипса; trim_1 = trim_2 — условие замкнутости кривой (начальная точка вычленения совпадает с конечной точкой вычленения); basis_curve (= ellipse) — базовая кривая (эллипс)

Рисунок А.32 — Функция Ellipse_2_Diameter_A2p

Внутренние ссылки: 6.1.9, 6.1.13.1, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.3.2.1.2 Построение дуги эллипса

Имя функции:	Уровень интерфейса:	1
Elc Gen	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	SEMI1	D	Большая полуось *semi_axis_1* эллипса *api_elliptical_arc*	(EPS ≤ SEMI1 ≤ MAX)
Ввод	SEMI2	D	Малая полуось *semi_axis_2* эллипса *api_elliptical_arc*	(EPS ≤ SEMI2 ≤ MAX)
Ввод	STAANG	D	Начальный угол в плоскости (Oxy) относительно оси (Ox) заданной локальной координатной системы *a2p*	(0° ≤ STAANG ≤ 360°)
Ввод	ENDANG	D	Конечный угол в плоскости (Oxy) относительно оси (Ox) заданной локальной координатной системы *a2p*	(0° ≤ ENDANG ≤ 360°)
Ввод	A2PNAM	N	Имя локальной координатной системы *axis2_placement*	a2p
Ввод	SENSE	E	Знак направления обхода кривой	[TRUE, FALSE]
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_elliptical_arc*	elc

Привязка языка FORTRAN:
NAME = ELC_GEN (SEMI1, SEMI2, STAANG, ENDANG, A2PNAM, SENSE, KFIX)

Результат использования функции

Функция создает дугу эллипса *api_elliptical_arc* по большой полуоси (SEMI1), малой полуоси (SEMI2), локальной координатной системе *axis2_placement* (A2PNAM) и заданному направлению обхода контура (SENSE) вновь созданной сущности *api_elliptical_arc* в совокупности с сущностью *ellipse* в качестве базовой кривой *basis_curve*. Начальная и конечная точки дуги неявно определяются углами STAANG и ENDANG соответственно. Направление оси (Ox) локальной координатной системы A2PNAM определяет направление большой полуоси эллипса.

Локальная координатная система *axis2_placement* (A2PNAM) дублируется как сущность *a2p1*, имеющая нулевой стиль *null_style*. Затем:

- создается экземпляр *e* эллипса с центром *position* в начале координат *a2p1*, большая полуось *semi_axis_1* равна SEMI1, малая полуось *semi_axis_2* равна SEMI2. Полученный эллипс имеет нулевой стиль;

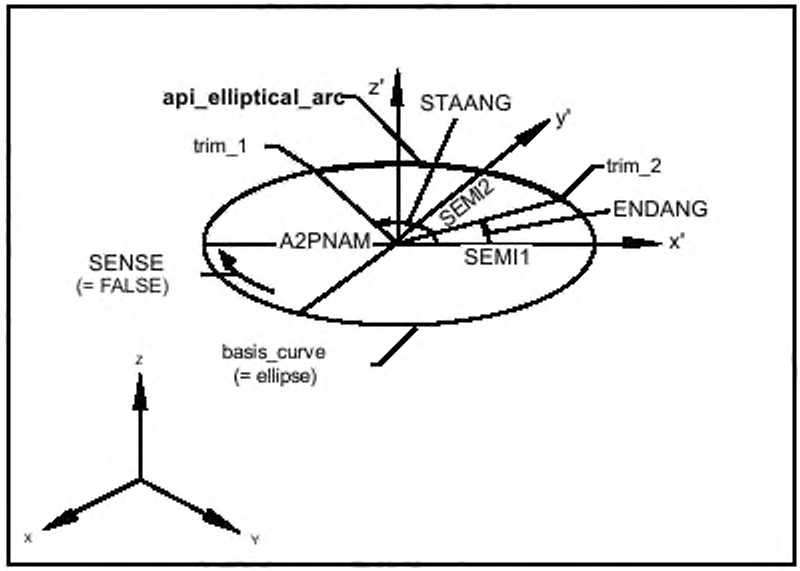
- создается экземпляр дуги эллипса *api_elliptical_arc* с кривой *e* в качестве базовой кривой *basis_curve*. Значение параметра STAANG соответствует точке вычленения *trim_1*, значение параметра ENDANG соответствует точке вычленения *trim_2*. Значение атрибута направления обхода кривой *sense_agreement* равно параметру SENSE, значение атрибута главного представления *master_representation* зависит от реализации. Полученная дуга эллипса *api_elliptical_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя построенной дуги эллипса *api_elliptical_arc*.

Значения параметров SEMI1 и SEMI2 лежат в диапазоне [EPS, MAX] и измеряются в единицах длины *OVC_length_unit*. Углы измеряются в единицах угла *OVC_angle_unit* и вычисляются в плоскости (Oxy) заданной локальной координатной системы A2PNAM. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Если два указанных параметра (STAANG и ENDANG) определяют одну и ту же точку в диапазоне [*ZERO_value*, EPS], то точки вычленения *trim_1* и *trim_2* совпадают. При этом созданная дуга окружности является полным эллипсом. Интерфейс гарантирует замкнутость созданной сущности *api_elliptical_arc*.

2 Сущность *api_elliptical_arc* создается, если длина дуги, проведенной из точки вычленения *trim_1* в точку вычленения *trim_2* и совместимой с соответствующим значением атрибута *sense_agreement*, больше допуска EPS. В противном случае возникает ошибка.



Api_elliptical_arc — результирующая дуга эллипса; STAANG — начальный угол дуги; trim_1 — первая точка вычленения; trim_2 — вторая точка вычленения: SEMI2 — малая полуось эллипса SEMI2; ENDANG — конечный угол дуги; A2PNAM — локальная координатная система; SEMI1 — большая полуось эллипса SEMI1; SENSE (= FALSE) — направление обхода дуги — отрицательное (по часовой стрелке); basis_curve (= ellipse) — базовая кривая (эллипс)

Рисунок А.33 — Функция Elc_Gen

Внутренние ссылки: 6.1.9, 6.1.13.1, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
101	Попытка создания вырожденной сущности	112	Попытка создания дуги длиной меньше EPS
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.3.2.2 Построение дуги гиперболы (api_hyperbolic_arc)

Построение дуги гиперболы Hyp_Gen

A.5.3.2.2.1 Построение дуги гиперболы

Имя функции:	Уровень интерфейса:	1
Hyp_Gen	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	SEMAXI	D	Действительная полуось гиперболы *hyperbola*	(EPS ≤ SEMAXI ≤ MAX)
Ввод	SEMIMG	D	Мнимая полуось гиперболы *hyperbola*	(EPS ≤ SEMIMG ≤ MAX)
Ввод	STAANG	D	Начальный угол в плоскости (Oxy) локальной координатной системы *a2p*, отсчитывается вокруг фокусной точки от оси (Ox)	(0° ≤ STAANG ≤ 360°)
Ввод	ENDANG	D	Конечный угол в плоскости (Oxy) заданной локальной координатной системы *a2p*, отсчитывается вокруг фокусной точки от оси (Ox)	(0° ≤ ENDANG ≤ 360°)
Ввод	A2PNAM	N	Имя локальной координатной системы *axis2_placement*	a2p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_hyperbolic_arc*	hyp

Привязка языка FORTRAN:
NAME = HYP_GEN (SEMAXI, SEMIMG, STAANG, ENDANG, A2PNAM, KFIX)

Результат использования функции

Функция создает сущность *api_hyperbolic_arc*, которая является дугой гиперболы *hyperbola*, определенной по действительной полуоси (SEMAXI), мнимой полуоси (SEMIMG) и по локальной координатной системе *axis2_placement* (A2PNAM), задающей расположение и направление осей гиперболы. Ветви гиперболы расположены вдоль оси (Ox) локальной координатной системы A2PNAM. Направление обхода вновь созданной дуги гиперболы *api_hyperbolic_arc* неявно задается начальной точкой дуги и конечной точкой дуги, соответствующими начальному углу STAANG и конечному углу ENDANG. Направление оси (Ox) локальной координатной системы A2PNAM определяет направление действительной полуоси гиперболы.

Локальная координатная система *axis2_placement* (A2PNAM) дублируется как сущность *a2p1*, имеющая нулевой стиль *null_style*. Затем:

- создается экземпляр *h* гиперболы с центром локальной координатной системы *a2p1*, действительная полуось *semi_axis_1* равна SEMAXI, мнимая полуось *semi_imag_axis_2* равна SEMIMG. Данная гипербола имеет нулевой стиль;

- создается экземпляр *p1* декартовой точки *cartesian_point*, являющейся точкой пересечения прямой, проведенной из фокуса гиперболы *h* под углом STAANG, и самой гиперболой *h*. Данная точка имеет нулевой стиль;

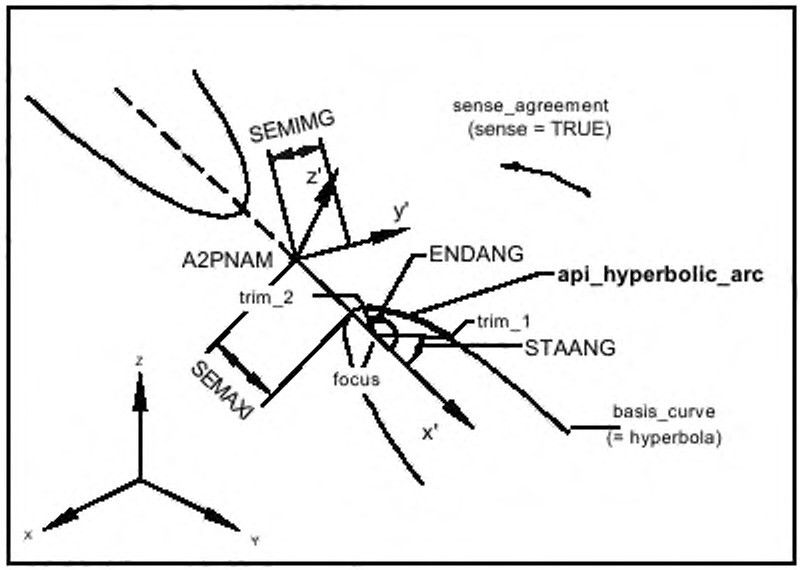
- создается экземпляр *p2* декартовой точки, являющейся точкой пересечения прямой, проведенной из фокуса гиперболы *h* под углом ENDANG, и самой гиперболы *h*. Данная точка имеет нулевой стиль;

- создается экземпляр сущности *api_hyperbolic_arc* с гиперболой *h* как базовой кривой *basis_curve*, точкой *p1* как точкой вычленения *trim_1* и точкой *p2* как точкой вычленения *trim_2*. При этом значение атрибута направления обхода контура *sense_agreement* равно «true», если для начального и конечного углов справедливо соотношение STAANG < ENDANG (в противном случае значение атрибута *sense_agreement* равно «false»), значение атрибута *master_representation* зависит от реализации. Полученная дуга гиперболы *api_hyperbolic_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей

уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя построенной сущности *api_hyperbolic_arc*.

Значения SEMAXI и SEMIMG лежат в диапазоне [EPS, MAX] и измеряются в единицах длины *OVC_length_unit*. Заданные углы измеряются в единицах угла *OVC_angle_unit*. Углы вычисляются в плоскости (Oxy) заданной локальной координатной системы A2PNAM. Они откладываются вокруг фокусной точки гиперболы от оси (Ox). При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Если начальное STAANG и конечное ENDANG значения угла определяют одну и ту же точку в диапазоне [*ZERO_value*, EPS] (длина дуги гиперболы *api_hyperbolic_arc* меньше допуска EPS) или если процесс вычислений для двух декартовых точек *p1* и *p2* не дает решения, то дуга гиперболы *api_hyperbolic_arc* не создается и возникает ошибка.



Sense_agreement (sense = TRUE) — положительное направление обхода гиперболы (против часовой стрелки); SEMIMG — минимая полуось гиперболы, ENDANG — конечный угол дуги; A2PNAM — локальная координатная система: api_hyperbolic_arc — результирующая дуга гиперболы; trim_2 — вторая точка вычленения: trim_1 — первая точка вычленения; STAANG — начальный угол дуги, SEMAXI — действительная полуось гиперболы; focus — фокус; basis_curve (= hyperbola) — базовая кривая (гипербола)

Рисунок А.34 — Функция Hyp_Gen

Внутренние ссылки: 6.1.9, 6.1.13.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
101	Попытка создания вырожденной сущности	112	Попытка создания дуги длиной меньше EPS
128	Неустойчивость процесса вычисления конической дуги	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.3.2.3 Построение дуги параболы (сущность api_parabolic_arc)

Построение дуги параболы Par_Gen

A.5.3.2.3.1 Построение дуги параболы

Имя функции:	Уровень интерфейса:	1
Par Gen	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	FOCLEN	D	Фокусное расстояние параболы	(EPS ≤ FOCLEN ≤ MAX)
Ввод	STAANG	D	Начальный угол в плоскости (Oxy) в локальной координатной системе *a2p*, отложенный вокруг фокусной точки от оси (Ox)	(0° ≤ STAANG ≤ 360°)
Ввод	ENDANG	D	Конечный угол в плоскости (Oxy) в локальной координатной системе *a2p*, отложенный вокруг фокусной точки от оси (Ox)	(0° ≤ ENDANG ≤ 360°)
Ввод	A2PNAM	N	Локальная координатная система *axis2_placement*	a2p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_hyperbolic_arc*	par

Привязка языка FORTRAN:
NAME = PAR_GEN (FOCLEN, STAANG, ENDANG, A2PNAM, KFIX)

Результат использования функции

Функция создает сущность *api_parabolic_arc*, которая является дугой параболы *parabola*, определенной фокусным расстоянием (FOCLEN), локальной координатной системой *axis2_placement* (A2PNAM), определяющей положение вершины и оси параболы. Значение параметра FOCLEN определяет расстояние от вершины параболы до ее фокуса в направлении оси (Ox) локальной координатной системы A2PNAM. Направление вновь созданной дуги параболы *api_parabolic_arc* неявно определяется начальной и конечной точками, соответствующими начальному STAANG и конечному ENDANG углам соответственно. Ось (Ox) локальной координатной системы A2PNAM определяет ось симметрии параболы.

Локальная координатная система *axis2_placement* (A2PNAM) дублируется сущностью *a2p1*, имеющей нулевой стиль *null_style*. Затем:

- создается экземпляр *p* параболы с началом координат локальной координатной системы *a2p1*, фокусное расстояние *focal_dist* равно FOCLEN. Построенная кривая имеет нулевой стиль;

- создается экземпляр *p1* декартовой точки *cartesian_point*, являющейся точкой пересечения прямой, проведенной из фокуса параболы *p* под углом STAANG, и самой параболы *p*. Данная декартова точка *cartesian_point* имеет нулевой стиль;

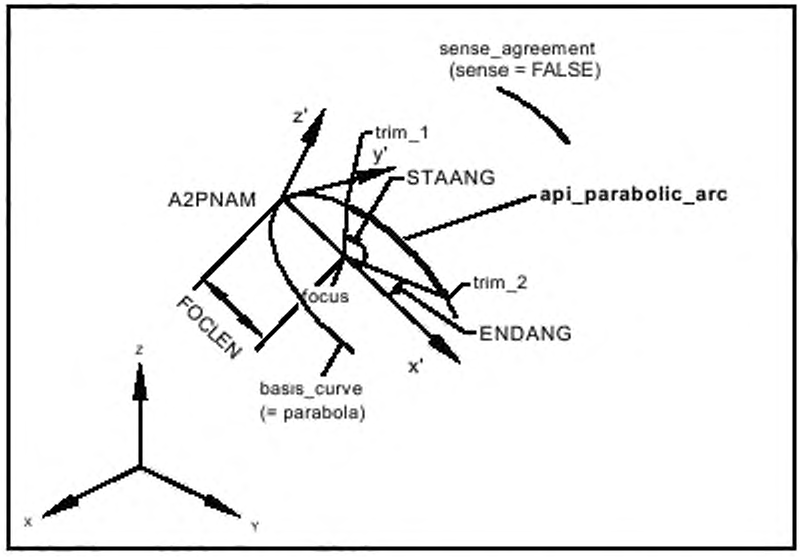
- создается экземпляр *p2* декартовой точки, являющейся точкой пересечения прямой, проведенной из фокуса параболы *p* под углом ENDANG, и самой параболы *p*. Данная декартова точка имеет нулевой стиль;

- создается экземпляр сущности *api_parabolic_arc* с параболой *p* в качестве базовой кривой *basis_curve*, точкой *p1* как точкой вычленения *trim_1* и точкой *p2* как точкой вычленения *trim_2*. Значение атрибута направления обхода кривой *sense_agreement* равно «true», если выполняется соотношение STAANG < ENDANG (в противном случае значение атрибута *sense_agreement* равно «false»). Значение атрибута *master_representation* зависит от реализации. Результирующая дуга параболы *api_parabolic_arc* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение

записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя построенной дуги параболы *api_parabolic_arc*.

Значение фокусного расстояния FOCLEN лежит в диапазоне [EPS, MAX] и измеряется в единицах длины *OVC_length_unit*. Углы измеряются в единицах угла *OVC_angle_unit* и откладываются в плоскости (Oxy) заданной локальной координатной системы A2PNAM вокруг фокусной точки от оси (Ox). При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Если начальное и конечное значения угла STAANG и ENDANG определяют одну и ту же точку в диапазоне [*ZERO_value*, EPS] (длина созданной дуги параболы *api_parabolic_arc* меньше допуска EPS) или если используемый процесс вычислений декартовых точек *p1* и *p2* не дает решения, то дуга параболы *api_parabolic_arc* не создается и возникает ошибка.



Sense_agreement (sense = FALSE) — направление обхода дуги отрицательное (по часовой стрелке); trim_1 — первая точка вычленения; STAANG — начальный угол дуги параболы; A2PNAM — локальная координатная система; api_parabolic_arc — результирующая дуга параболы; trim_2 — вторая точка вычленения, focus — фокус, FOCLEN — фокусное расстояние; ENDANG — конечный угол дуги параболы; basis_curve (= parabola) — базовая кривая (парабола)

Рисунок А.35 — Функция Par_Gen

Внутренние ссылки: 6.1.9, 6.1.13.3, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
101	Попытка создания вырожденной сущности	112	Попытка создания дуги длиной меньше EPS
128	Неустойчивость процесса вычисления конической дуги	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.3.3 Общие сущности кривых

A.5.3.3.1 Построение полилинии

Построение полилинии по декартовым координатам	Pln_Cartesian_Coordinate
Построение полилинии по заданному перечню точек	Pln_Pnt_List

A.5.3.3.1.1 Построение полилинии по декартовым координатам

Имя функции:	Уровень интерфейса:	1
Pln_Cartesian_Coordinate	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	N	1	Длина каждого из перечней координат XLST, YLST и ZLST, определяющих количество (*n*) декартовых точек *cartesian_point* в перечне точек полилинии *polyline*	≥ 2
Ввод	XLST	nxD	Перечень координат *X* декартовых точек *cartesian_point* p_i [см. примечание (1)]	(0.0 или (EPS ≤ |XLST(i)| ≤ MAX))
Ввод	YLST	nxD	Перечень координат *Y* декартовых точек *cartesian_point* p_i [см. примечание (1)]	(0.0 или (EPS ≤ |YLST(i)| ≤ MAX))
Ввод	ZLST	nxD	Перечень координат *Z* декартовых точек *cartesian_point* p_i [см. примечания (1) и (2)]	(0.0 или (EPS ≤ |ZLST(i)| ≤ MAX))
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *polyline*	pln

Привязка языка FORTRAN:
NAME = PLN_CARTESIAN_COORDINATE (N, XLST, YLST, ZLST, KFIX)

Результат использования функции

Функция создает сущность *polyline*, определенную тремя перечнями декартовых координат (XLST, YLST, ZLST). Каждая тройка координат *X*, *Y* и *Z* (XLST(i), YLST(i), ZLST(i) для каждого $1 < i < n$) указанных перечней определяет декартову точку *cartesian_point* из перечня декартовых точек, являющуюся атрибутом полилинии. Затем:

- создается *n* наборов декартовых точек p_i с координатами *X*, *Y* и *Z*, равными $XLST_i$, $YLST_i$ и $ZLST_i$, где $i = 1, \ldots, n$, соответственно. Все указанные декартовы точки имеют нулевой стиль *null_style*;

- создается экземпляр полилинии с точками из перечня декартовых точек p_i полилинии, где $i = 1, \ldots, n$. Полученная полилиния имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя построенной полилинии.

Длина каждого линейного сегмента полилинии, построенного по двум декартовым точкам p_i и p_{i+1}, где $i = 1, \ldots, n - 1$, лежит в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Длины (*n*) перечней декартовых координат XLST, YLST и ZLST, определяющих количество точек p_i, необходимых для построения полилинии с линейными сегментами, должны быть одинаковыми для всех перечней.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то перечень с декартовыми координатами *Z* (ZLST) игнорируется интерфейсом.

Внутренние ссылки: 6.1.9, 6.1.14.1, 6.2.4, 8.2.

Ошибки

3	Значение меры длины находится вне допустимого диапазона	5	Целочисленное значение находится вне допустимого диапазона
101	Попытка создания вырожденной сущности	103	Значение расстояния между двумя точками находится вне установленного диапазона [EPS, MAX]
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	205	Превышено максимально допустимое количество точек полилинии
1001	Перечислимое значение находится вне установленного диапазона		

A.5.3.3.1.2 Построение полилинии по заданному перечню точек

Имя функции:

Pln_Pnt_List

Уровень интерфейса:	1
Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	N	I	Длина перечня PNTLST, определяющего количество точек полилинии	≥ 2
Ввод	PNTLST	nxN	Перечень имен декартовых точек *cartesian_point* (см. ниже)	pnt
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *polyline*	pln

Привязка языка FORTRAN:
NAME = PLN_PNT_LIST (N, PNTLST, KFIX)

Результат использования функции

Функция создает сущность *polyline*, определенную перечнем заданных декартовых точек *cartesian_point* (PNTLST). Длина указанного перечня PNTLST, равная *N* (*n*), определяет количество точек, необходимых для построения линейных сегментов полилинии.

Перечень *list1* декартовых точек, обозначенных p_i, где i = 1, ..., *n*, создан путем дублирования *n* декартовых точек перечня PNTLST. Все указанные декартовы точки имеют нулевой стиль *null_style*. Затем:

- создается экземпляр полилинии с точками перечня *list1*. Указанная полилиния имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной полилинии.

Длина каждого линейного сегмента, построенного по двум декартовым точкам, обозначенным p_i и p_{i+1}, где $i = 1, ..., n - 1$, лежит в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.

Внутренние ссылки: 6.1.9, 6.1.14.1, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
5	Целочисленное значение находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
103	Расстояние между двумя точками находится вне установленного диапазона [EPS, MAX]	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
205	Превышение максимально допустимого количества точек полилинии	1001	Перечислимое значение находится вне установленного диапазона

A.5.3.3.2 Плоский контур (сущность api_contour)

Построение плоского контура Ctr_Gen

A.5.3.3.2.1 Построение плоского контура

Имя функции:	Уровень интерфейса:	1
Ctr_Gen	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	N	I	Длина перечня ENTLST, определяющего количество элементов	≥ 1
Ввод	ENTLST	nxN	Перечень имен сущностей, определяющих контур *api_contour*	basic, conic_arc, pln, grp
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_contour*	ctr

Привязка языка FORTRAN:
NAME = CTR_GEN (N, ENTLST, KFIX)

Результат использования функции

Функция создает сущность *api_contour* несамопересекающейся ориентированной плоской замкнутой комбинированной кривой *composite_curve*, построенной с помощью перечня заданных сущностей (ENTLST). Данный перечень сущностей кривых может быть упорядоченным или неупорядоченным. Все экземпляры указанных сущностей, определенные перечнем ENTLST, дублируются. Они имеют нулевой стиль *null_style*. Затем:

- интерфейсом вычисляется перечень *list1* сегментов комбинированной кривой *composite_curve_segment* с помощью точной копии заданной сущности. Процесс вычислений, установленный в подразделе 6.1.14.2, гарантирует замкнутость полученного контура. Данный процесс доставляет необходимое количество *n* сегментов комбинированной кривой *composite_curve_segment*, используемых атрибутом *n_segment*;

- создается экземпляр контура *api_contour* с сегментами, определенными перечнем *list1*. Значение атрибута *self_intersect* равно «false», значение атрибута *n_segment* равно *n* и значение атрибута *closed_curve* равно «true». Полученный контур *api_contour* имеет текущую запись *curve_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя полученной сущности *api_contour*.

При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Внутри результирующего контура *api_contour* должна помещаться окружность диаметром EPS.

2 Если экземпляр сущности, заданной перечнем ENTLST, имеет тип *api_group*, то все экземпляры сущностей (содержащиеся внутри данной группы и существующие в параметрическом диапазоне настоящей функ-

ции, например, *basic*, *conic_arc* и *pln*) дублируются для создания перечня *list1* сегментов комбинированной кривой *composite_curve_segment*.

3 Сущности, используемые для представления контура, имеют соответствующие записи *contour_entities* в таблице описаний интерфейса. Если существуют допустимые сущности кривых, используемые настоящей функцией, но не попадающие в диапазон *contour_entities*, то данные сущности моделируются особым образом, определенным интерфейсом для указанных сущностей.

4 Если текущий открытый вид определен как 3D-вид (значение записи *geometrial_power_level* в таблице статуса интерфейса не менее 2), то все элементы перечня ENTLST должны лежать в одной плоскости.

Внутренние ссылки: 6.1.12, 6.1.13, 6.1.14, 6.1.14.2, 6.1.19, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
5	Целочисленное значение находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
113	Попытка создания самопересекающейся сущности контура	115	Заданные сущности являются идентичными
119	Заданные сущности не лежат в одной плоскости	129	Сбой процесса аппроксимации, гарантирующего замыкание контура
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	206	Превышение максимально допустимого количества сущностей контура
1001	Перечислимое значение находится вне установленного диапазона		

А.5.4 Сущности создания заполненной области

Построение заполненной области комментариев *annotation_fill_area*	Afa_Gen
Задание стиля штриховки заполненной области *fill_area_style_hatching*	Fsh_Gen
Штриховка заполненной области комментариев *annotation_fill_area*	Hatch_Afa

А.5.4.1 Построение заполненной области комментариев annotation_fill_area

Имя функции:	Уровень интерфейса:	1
Afa_Gen	Уровень геометрической мощности:	1

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/ значение
Ввод	CTRNAM	N	Имя контура *api_contour* наружной границы	ctr
Ввод	N	I	Длина перечня CTRLST, определяющего количество контуров *api_contour* внутренних границ	≥ 0
Ввод	CTRLST	nxN	Перечень имен внутренних контуров *api_contour*	ctr
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности заполненной области комментариев *annotation_fill_area*	afa

Привязка языка FORTRAN:
NAME = AFA_GEN (CTRNAM, N, CTRLST, KFIX)

Результат использования функции

Функция создает сущность *annotation_fill_area*, представляющую собой множество кривых, область между которыми может быть заполнена штриховкой, затенением, цветом или мелким узором. Сущность *annotation_fill_area* описывается путем задания границ, представляющих собой непересекающиеся и несамопересекающиеся замкнутые кривые. Эти кривые задают границы плоской области, заполненной в соответствии с установленным стилем. Сущность *annotation_fill_area* представляет собой плоскую двусвязную область, внешняя граница которой является заданным контуром *api_contour* с именем CTRNAM. Данная область может иметь внутреннюю границу, определенную перечнем контуров *api_contour* с именем CTRLST.

Все контуры лежат в одной плоскости. Они не должны пересекать друг друга. Все (возможные) контуры внутренних границ заполненной области должны находиться внутри наружного контура. Внутренние контуры не пересекаются.

Заданные сущности *api_contour* интерфейса прикладного программирования дублируются как контуры a_1,... a_n. Указанные сущности имеют нулевой стиль *null_style*. Затем:

- создается экземпляр сущности *annotation_fill_area* с границами в виде множества контуров *api_contour* a_1, ..., a_n. Данная сущность имеет текущую запись *fill_area_style* в таблице статуса интерфейса. В случае открытого 2D-вида (если значение записи *hidden_line* равно «on» (включено) и значение записи *hidden_line_involved* равно «true») полученная сущность приобретает предварительно установленный стиль затенения *api_pre_defined_occlusion_style* с текущими значениями записей уровня вида *view_level* и аспекта невидимых линий *hidden_line_aspect* таблицы статуса интерфейса. Функция возвращает имя построенной заполненной области *annotation_fill_area*. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 В контексте схемы *api_abstract_schema* использование сущностей *annotation_fill_area* допустимо только для 2D-видов (уровень геометрической мощности *geometrical_power_level* интерфейса равен 1).

2 Если внутренних границ нет (длина перечня внутренних контуров CTRLST равна 0), то параметр CTRLST игнорируется.

3 Если существует два и более контура *api_contour* в описании заполненной области комментариев *annotation_fill_area*, то расстояние между двумя контурами *api_contour* не лежит в диапазоне [*ZERO_value*, EPS].

Внутренние ссылки: 6.1.14, 6.1.15.1, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
5	Целочисленное значение находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
104	Расстояние между двумя контурами меньше EPS	119	Заданные сущности не лежат в одной плоскости
123	Выявлено пересечение заданных контуров	125	Выявлено перекрытие заданных контуров
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	207	Превышено максимально допустимое количество внутренних границ
1001	Перечислимое значение находится вне установленного диапазона		

A.5.4.2 Задание стиля штриховки заполненной области fill_area_style_hatching

Имя функции:	Уровень интерфейса:	1
Fsh_Gen	Уровень геометрической мощности:	1

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	REFPNT	N	*Имя декартовой точки cartesian_point* начала отображения	pnt
Ввод	DIST1	D	Расстояние между параллельными линиями штриховки	(EPS ≤ DIST1 ≤ MAX)
Ввод	DIST2	D	Расстояние от точки REFPNT, задающей начальную точку шаблона	(0.0 или (EPS ≤ DIST2 ≤ MAX))
Ввод	ANGLE	D	Угол штриховки относительно оси (Ox) текущей OVC	(0° ≤ ANGLE ≤ 180°)
Вывод	NAME	N	*Имя созданной сущности fill_area_style_hatching*	fsh

Привязка языка FORTRAN:
NAME = FSH_GEN (REFPNT, DIST1, DIST2, ANGLE)

Результат использования функции

Функция создает сущность *fill_area_style_hatching*, определяющую стилизованный шаблон для линий штриховки видимых заполненных областей.

Заданная декартова точка *cartesian_point* с именем REFPNT, определяющая начальную точку отображения сущности *fill_area_style_hatching* с именем NAME, дублируется как точка *p1* с нулевым стилем *null_style*. Пусть прямая *l* является виртуальной ссылочной линией штриховки, проходящей через точку *p1* под заданным углом ANGLE. Затем:

- создается экземпляр *v* вектора, перпендикулярного ссылочной линии штриховки *l*. Модуль данного вектора равен DIST1. Вектор имеет нулевой стиль;
- создается экземпляр *o* фактора повторения заданного направления *one_direction_repeat_factor* в виде фактора повторения *repeat_factor* вектора *v*;
- создается экземпляр *p2* декартовой точки либо путем дублирования ссылочной точки REFPNT (если заданное расстояние DIST2 равно 0), либо путем вычисления точки на ссылочной линии штриховки *l* на расстоянии DIST2 от заданной ссылочной точки REFPNT. Полученная декартова точка имеет нулевой стиль;
- стиль кривой *curve_style* определяется значениями записей таблицы статуса интерфейса: толщина линий штриховки *hatch_width*, тип линий штриховки *hatch_curve_font* и цвет штриховки *hatch_colour*;
- создается экземпляр стиля штриховки заполненной области *fill_area_style_hatching* со значением атрибута вида линий штриховки *hatch_line_appearance* равным *c*, значением атрибута начала следующей линии штриховки *start_of_next_hatching_line*, равным *o*, значением атрибута линии штриховки, проходящей через ссылочную точку *point_of_reference_hatching_line*, равным *p1*, значением атрибута начала шаблона *pattern_start*, равным *p2*, и значением атрибута угла наклона линий штриховки *hatch_line_angle* равным ANGLE.

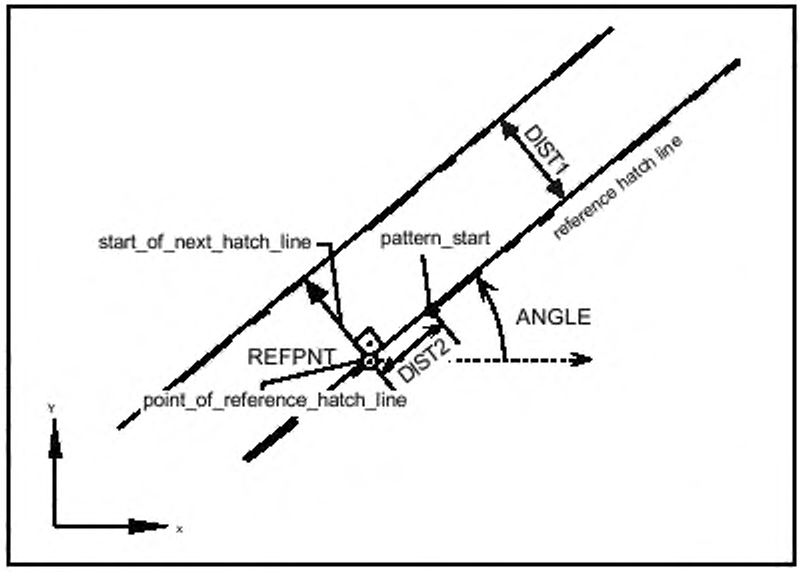
Оба заданных расстояния DIST1 и DIST2 измеряются в единицах длины *OVC_length_unit*. Значение DIST1 лежит в диапазоне [EPS, MAX]. Значение DIST2 либо равно 0, либо лежит в диапазоне [EPS, MAX]. Значение угла наклона ANGLE вычисляется в единицах угла *OVC_angle_unit*. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 В контексте схемы *api_abstract_schema* сущность *fill_area_style_hatching* используется только для стилизации штриховки заполненной области *annotation_fill_area*. Следовательно, указанная сущность стиля штриховки хранится только во временной базе данных.

2 В контексте схемы *api_abstract_schema* вид линий штриховки *hatch_line_appearance* задается предварительно определенными элементами *pre_defined_item*.

3 Создание сущности *fill_area_style_hatching* для 2D-вида допускается, если уровень геометрической мощности интерфейса *geometrical_power_level* равен 1.



DIST1 — расстояние между линиями штриховки; reference hatch line — ссылочная линия штриховки; start_of_next_hatch_line — начало следующей линии штриховки, pattern_start — начало шаблона; ANGLE — угол наклона штриховки; REFPNT — ссылочная точка; DIST2 — расстояние от ссылочной точки до начала шаблона; point_of_reference_hatch_line — точка ссылочной линии штриховки

Рисунок А.36 — Функция Fsh_Gen

Внутренние ссылки: 6.1.9, 6.1.15, 6.2.3.6, 6.2.5, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
101	Попытка создания вырожденной сущности	201	Переполнение временной базы данных
204	Функция несовместима с текущим уровнем мощности		

А.5.4.3 Штриховка заполненной области комментариев annotation_fill_area

Имя функции:	Уровень интерфейса:	1
Hatch_Afa	Уровень геометрической мощности:	1

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	HATCH	N	Имя отображаемой сущности *fill_area_style_hatching*	fsh
Ввод	AFA	N	Имя заполненной области *annotation_fill_area*	afa
Ввод	TARPNT	N	Имя декартовой точки *cartesian_point*, задающей исходное положение для штриховки	pnt

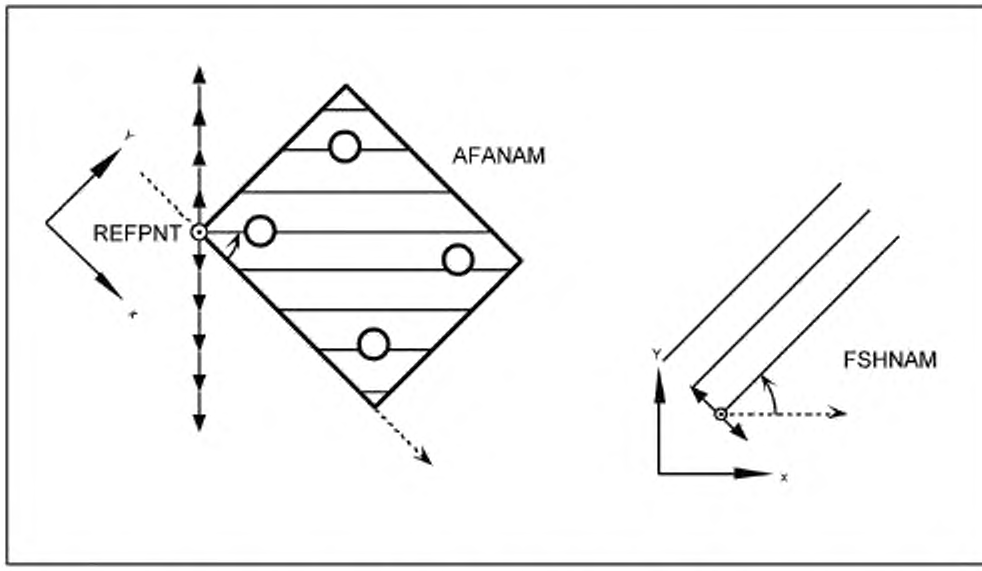
Привязка языка FORTRAN:
CALL HATCH_AFA (HATCH, AFA, REFPNT)

Результат использования функции

Функция назначает стиль штриховки *fill_area_style_hatching* (HATCH) заполненной области *annotation_fill_area* (AFA). Целевая точка задается сущностью *cartesian_point* (TARPNT). Целевая точка описывает начальную точку стиля заполнения области. Указанная точка с именем TARPNT дублируется как точка *p1*, она имеет нулевой стиль *null_style*. Затем создается экземпляр сущности *annotation_fill_area_occurrence*. При этом целевая точка задания стиля *fill_style_target* равна *p1*, значение атрибута стиля заполнения *SELF.styles* равно HATCH, элемент представления *SELF.item* равен AFA.

Назначение выполняется с помощью преобразования, выполняемого для начальной точки ссылочной линии штриховки HATCH (как начала координат) и целевой точки стиля заполнения *fill_style_target* (с именем *p1*). При этом производится неявное отображение оси *X* ссылочной координатной системы OVC, которой принадлежит сущность HATCH, на ось *X* ссылочной координатной системы OVC, которой принадлежит целевая точка REFPNT. При возникновении ошибки никаких назначений не производится.

Примечание — В контексте схемы *api_abstract_schema* назначение указанного стиля допустимо только для 2D-видов (геометрическая мощность интерфейса *geometrical_power_level* равна 1).



AFANAM — область штриховки; REFPNT — ссылочная точка; FSHNAM — стиль штриховки

Рисунок А.37 — Функция Hatch_Afa

Внутренние ссылки: 6.1.9, 6.1.15, 6.2.3.13, 6.2.3.6, 6.2.5, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	204	Функция несовместима с текущим уровнем мощности
403	Сбой при назначении стиля штриховки		

А.5.5 Сущности поверхности

Построение плоской поверхности *api_planar_surface* Aps_Gen

A.5.5.1 Построение плоской поверхности api_planar_surface

Имя функции:	Уровень интерфейса:	2
Aps_Gen	Уровень геометрической мощности:	2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	CTRNAM	N	Имя наружной границы *api_contour*	ctr
Ввод	N	I	Длина перечня CTRLST, определяющего количество контуров *api_contour* внутренних границ	≥ 0
Ввод	CTRLST	n × N	Перечень имен внутренних контуров *api_contour*	ctr
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *api_planar_surface*	aps

Привязка языка FORTRAN:
NAME = APS_GEN (CTRNAM, N, CTRLST, KFIX)

Результат использования функции

Сущность *api_planar_surface* определяется контуром *api_contour* (с именем CTRNAM), соответствующим внешней границе поверхности, и перечнем контуров *api_contour* (с именем CTRLST), соответствующим внутренним (возможным) границам поверхности. Если внутренних границ нет, то параметр N (определяющий длину перечня CTRLST) равен 0. Все контуры должны лежать в одной плоскости и не пересекаться. Контуры, соответствующие внутренним границам, должны принадлежать ограниченной поверхности, определенной контуром *api_contour* (с именем CTRNAM), соответствующим внешней границе, ни один из них не должен принадлежать ограниченной поверхности, определенной другим контуром *api_contour*. Вышесказанное означает, что плоская поверхность *api_planar_surface* образована дугами. Если указанные условия выполняются, то плоская поверхность *api_planar_surface* вычисляется интерфейсом. Все сущности *api_contour* дублируются и имеют нулевой стиль *null_style*. Затем:

1) плоскость указанной поверхности вычисляется по ее положению. Положение плоскости *position.location* определяется первой точкой первого сегмента комбинированной кривой *composite_curve_segmentof* контура *api_contour* с именем CTRNAM. Положение плоскости определяется также осью *X* (прямой *position.p[1]*), касательной к сегменту данной комбинированной кривой *composite_curve_segment*, определенному атрибутом *same_sense*. Ось *Z* положения плоскости (прямая *position.p[3]*) ортогональна плоскости, содержащей данный контур *api_contour* CTRNAM. Направление обхода контура *api_contour* положительно, если относительно оси *Z* обход производится против часовой стрелки;

2) для каждого контура *api_contour*, определяющего плоскую поверхность *api_planar_surface*, создается экземпляр кривой на поверхности *surface_curve*, ссылающийся на настоящий контур как на пространственную кривую *curve_3d*. Атрибут *associated_geometry* данной кривой на поверхности содержит только один элемент, который является плоскостью поверхности *api_planar_surface*, вычисленной на этапе 1. Значение атрибута главного представления *master_representation* настоящей кривой на поверхности равно *curve_3d*;

3) для каждой вычисленной кривой на поверхности создается экземпляр замкнутого сегмента комбинированной кривой *composite_curve_segment*, который:

- относится к соответствующей кривой на поверхности как к первоначальной (исходной) кривой *parent_curve*;
- содержит значение атрибута перехода для последнего сегмента комбинированной кривой *composite_curve_segment* контура *api_contour*, который является сущностью *curve_3d* соответствующей кривой на поверхности;
- содержит атрибут *same_sense*, значение которого равно «true» для сегмента комбинированной кривой *composite_curve_segment*, соответствующей внешней границе. Это значение гарантирует, что все прочие замкнутые сегменты комбинированной кривой ориентированы по часовой стрелке относительно оси *Z* для плоскости поверхности *api_planar_surface* (прямая *position.p[3]*, описанная на этапе 1);

4) создается экземпляр внешней границы *outer_boundary_curve*, сегменты которой содержат только один элемент: сегмент комбинированной кривой *composite_curve_segment*, первоначальная (исходная) кривая которого *parent_curve* относится как пространственная кривая *curve_3d* к контуру *api_contour* соответствующей внешней границы плоской поверхности *api_planar_surface*;

5) для каждого сегмента комбинированной кривой *composite_curve_segment* создается экземпляр граничной кривой *boundary_curve*, сегменты которой содержат только данный сегмент *composite_curve_segment*;

6) создается экземпляр плоской поверхности *api_planar_surface*. Ее базовой поверхностью *basis_surface* является плоскость для указанной сущности *api_planar_surface*. Границами являются (возможные) граничные кривые *boundary_curve*. Кривая наружной границы *outer_boundary_curve* вычисляется на этапах 4 и 5. Значение ее атрибута *implicit_outer* равно «false». Полученная плоская поверхность *api_planar_surface* имеет текущую запись *surface_style* в таблице статуса интерфейса. Функция возвращает имя полученной плоской поверхности. Все сущности имеют нулевой стиль *null_style*.

Примечания

1 Создание плоской поверхности *api_planar_surface* для 3D-видов допустимо, если уровень геометрической мощности интерфейса *geometrical_power_level* больше 2.

2 Если внутренних границ нет (длина перечня CTRLST равна 0), то параметр CTRLST игнорируется.

3 Если существует два и более контура *api_contour* в спецификации плоской поверхности *api_planar_surface*, то расстояние между двумя контурами расположено вне диапазона [*ZERO_value*, EPS].

Внутренние ссылки: 6.1.15, 6.1.16, 6.1.17.1, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
5	Целочисленное значение находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
104	Расстояние между двумя контурами меньше EPS	119	Заданные сущности не лежат в одной плоскости
123	Выявлено пересечение заданных контуров	125	Выявлено перекрытие заданных контуров
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
207	Превышение максимально допустимого количества внутренних границ	1001	Перечислимое значение находится вне установленного диапазона

А.5.6 Сущности геометрических тел

А.5.6.1 Сущности конструктивной блочной геометрии (CSG-сущности)

Построение сферы	Sph_Gen
Построение конуса	Con_Gen
Построение цилиндра	Cyl_Gen
Построение тора	Tor_Gen
Построение блока	Blk_Gen
Построение клина	Wdg_Gen

А.5.6.1.1 Построение сферы

Имя функции:	Уровень интерфейса:	3
Sph_Gen	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	RAD	D	Радиус сферы	(EPS ≤ RAD ≤ MAX)
Ввод	CNTPNT	N	Имя декартовой точки *cartesian_point*	pnt
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *sphere*	sph

Привязка языка FORTRAN:
NAME = SPH_GEN (RAD, CNTPNT, KFIX)

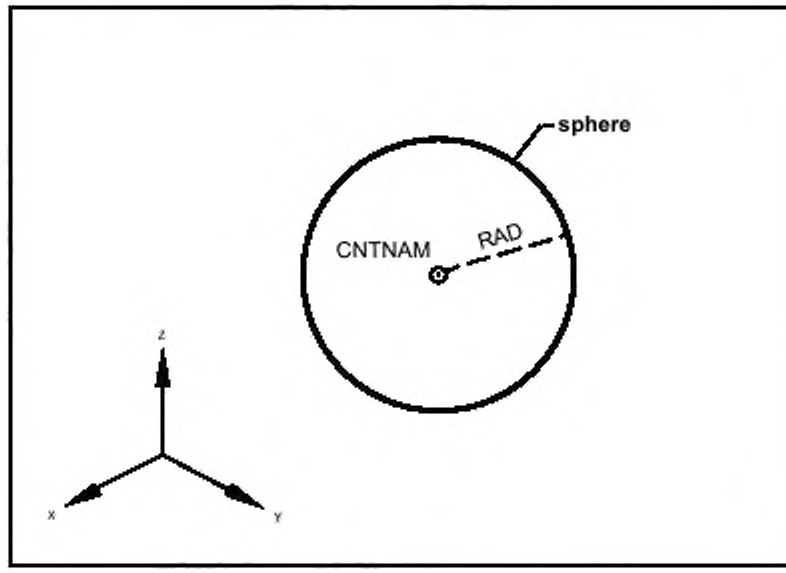
Результат использования функции

Сущность конструктивной блочной геометрии *sphere* задает сферу. Она определяется центром (декартовой точкой *cartesian_point* с именем CNTPNT) и радиусом RAD. Центр сферы CNTPNT дублируется как точка *p1*, имеющая нулевой стиль *null_style*.

Затем создается экземпляр сферы с радиусом RAD и центром в точке *p1*. Настоящая сфера имеет один стиль представления *presentation_style_assignment*, содержащий текущие записи таблицы статуса интерфейса для стиля поверхности *surface_style* и стиля кривой *curve_style*. Функция возвращает имя полученной сущности *sphere*.

Радиус RAD измеряется в единицах длины *OVC_length_unit*, его значение находится в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



Sphere — сфера; RAD — радиус; CNTNAM — центр

Рисунок А.38 — Функция Sph_Gen

Внутренние ссылки: 6.1.9, 6.1.18.4.1, 6.2.4, 6.2.3.2, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	203	Функция несовместима с текущим уровнем интерфейса
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.6.1.2 Построение конуса

Имя функции:	Уровень интерфейса:	3
Con_Gen	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ANGLE	D	Угол полураствора конуса (между центральной осью и образующей конической поверхности)	(0° ≤ ANGLE ≤ 90°)
Ввод	HEIGHT	D	Высота конуса	(EPS ≤ HEIGHT ≤ MAX)
Ввод	RAD	D	Радиус конуса в заданной точке на оси конуса	(0.0 или (EPS ≤ RAD ≤ MAX))
Ввод	A1PNAM	N	Имя локальной координатной системы *axis1_placement*	a1p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *right_circular_cone*	con

Привязка языка FORTRAN:
NAME = CON_GEN (ANGLE, HEIGHT, RAD, A1PNAM, KFIX)

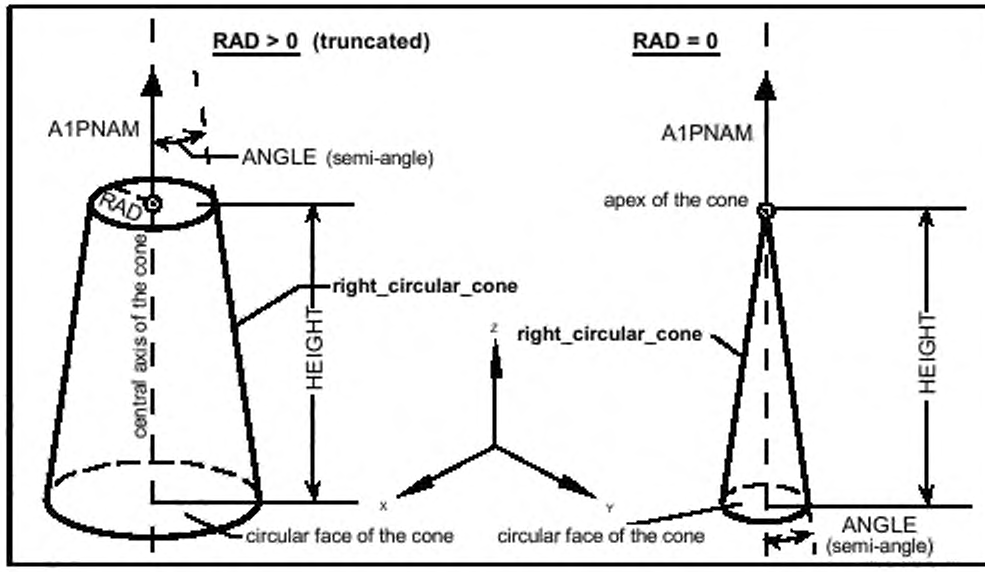
Результат использования функции

Сущность конструктивной блочной геометрии *right_circular_cone* задает прямой круговой конус. Конус может быть усеченным. Конус задается локальной координатной системой *axis1_placement* (с именем A1PNAM), углом полураствора конуса ANGLE и высотой конуса HEIGHT. Локальная координатная система *axis1_placement* определяет направление центральной оси симметрии конуса и точку на координатной оси, лежащую на одном из плоских круговых оснований (усеченного) конуса. Если радиус основания равен 0, то точка попадает в вершину конуса. Если радиус основания задан и не равен 0, то указанный радиус задает размеры и расположение малого основания усеченного конуса. Если радиус RAD больше допуска EPS, то высота конуса HEIGHT задает расстояние между плоскими круговыми основаниями (усеченного) конуса. Если радиус равен 0, то высота конуса есть расстояние от основания конуса до его вершины. Заданная локальная координатная система *axis1_placement* дублируется как сущность *a1p1*. Данная сущность имеет нулевой стиль *null_style*.

Затем создается экземпляр прямого кругового конуса *right_circular_cone* с локальной координатной системой *a1p1*, высотой, равной HEIGHT, радиусом, равным RAD, и углом полураствора конуса *semi_angle*, равным ANGLE. Созданному прямому круговому конусу *right_circular_cone* назначен единственный стиль презентации *presentation_style_assignment*, содержащий записи в таблице статуса интерфейса для стиля поверхности *surface_style* и стиля кривой *curve_style*. Функция возвращает имя полученного прямого кругового конуса *right_circular_cone*.

Высота HEIGHT и радиус RAD измеряются в единицах длины *OVC_length_unit*. Радиус RAD либо равен 0, либо находится в диапазоне [EPS, MAX]. Угол полураствора конуса вычисляется в единицах угла *OVC_angle_unit*. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



RAD > 0 (truncated) — радиус больше 0 (усеченный конус); RAD = 0 — радиус равен 0; A1PNAM — локальная координатная система; ANGLE (semi-angle) — угол полураствора конуса, RAD — радиус; apex of the cone — вершина конуса; central axis of the cone — центральная ось конуса; right_circular_cone — прямой круговой конус; HEIGHT — высота; circular face of the cone — основание конуса

Рисунок А.39 — Функция Con_Gen

Внутренние ссылки: 6.1.9, 6.1.18.4.2, 6.2.4, 6.2.3.2, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

А.5.6.1.3 Построение цилиндра

Имя функции:	Уровень интерфейса:	3
Cyl_Gen	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	RAD	D	Радиус цилиндра	(EPS ≤ RAD ≤ MAX)
Ввод	HEIGHT	D	Высота цилиндра	(EPS ≤ HEIGHT ≤ MAX)
Ввод	A1PNAM	N	Имя локальной координатной системы *axis1_placement*	a1p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *right_circular_cylinder*	cyl

Привязка языка FORTRAN:
NAME = CYL_GEN (RAD, HEIGHT, A1PNAM, KFIX)

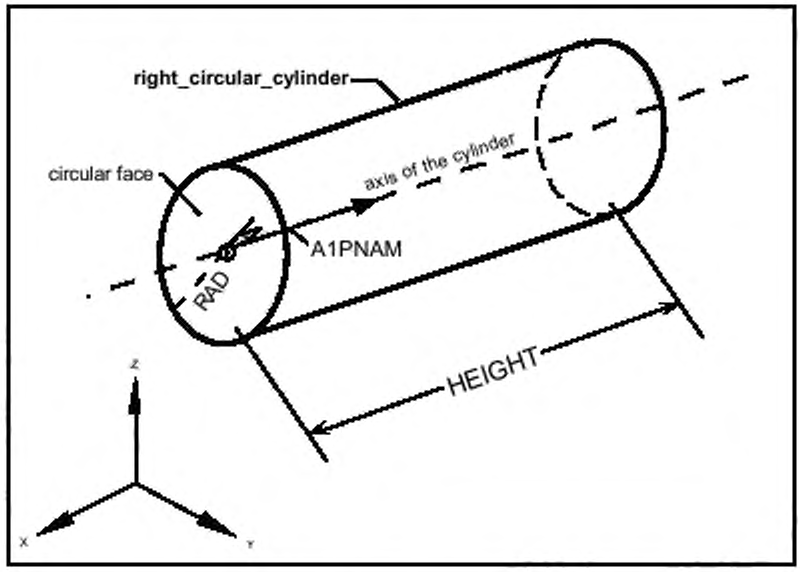
Результат использования функции

Сущность конструктивной блочной геометрии *right_circular_cylinder* представляет собой прямой круговой цилиндр. Он строится по радиусу RAD, высоте HEIGHT и локальной координатной системе *axis1_placement* с именем A1PNAM. Сущность *axis1_placement* определяет ось цилиндра и центр одного из оснований цилиндра. Высота цилиндра — это расстояние от центра первого основания до центра второго основания, измеренное вдоль оси в положительном направлении. Сущность *axis1_placement* дублируется сущностью *a1p1*, имеющей нулевой стиль *null_style*.

Затем создается экземпляр прямого кругового цилиндра *right_circular_cylinder* с локальной координатной системой *a1p1*, высотой, равной HEIGHT, и радиусом, равным RAD. Настоящему цилиндру *right_circular_cylinder* назначен единственный стиль представления *presentation_style_assignment*, который содержит текущие записи в таблице статуса интерфейса для стиля поверхности *surface_style* и стиля кривой *curve_style*. Функция возвращает имя построенной сущности *right_circular_cylinder*.

Высота цилиндра HEIGHT и радиус RAD измеряются в единицах длины *OVC_length_unit*. Значения параметров RAD и HEIGHT располагаются в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



Right_circular_cylinder — прямой круговой цилиндр; axis of the cylinder — ось цилиндра; circular face — основание цилиндра; A1PNAM — локальная координатная система; RAD — радиус; HEIGHT — высота

Рисунок А.40 — Функция Cyl_Gen

Внутренние ссылки: 6.1.9, 6.1.18.4.3, 6.2.4, 6.2.3.2, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	201	Переполнение временной базы данных

202	Ошибка при отправке сущности в CAD	203	Функция несовместима с текущим уровнем интерфейса
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.6.1.4 Построение тора

Имя функции:

Tor_Gen

Уровень интерфейса:	3
Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	MAJOR	D	Радиус направляющей окружности тора	(MINOR ≤ MAJOR ≤ MAX)
Ввод	MINOR	D	Радиус образующей окружности тора	(EPS ≤ MINOR ≤ MAX)
Ввод	A1PNAM	N	Имя локальной координатной системы *axis1_placement*	a1p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *torus*	tor

Привязка языка FORTRAN:
NAME = TOR_GEN (MAJOR, MINOR, A1PNAM, KFIX)

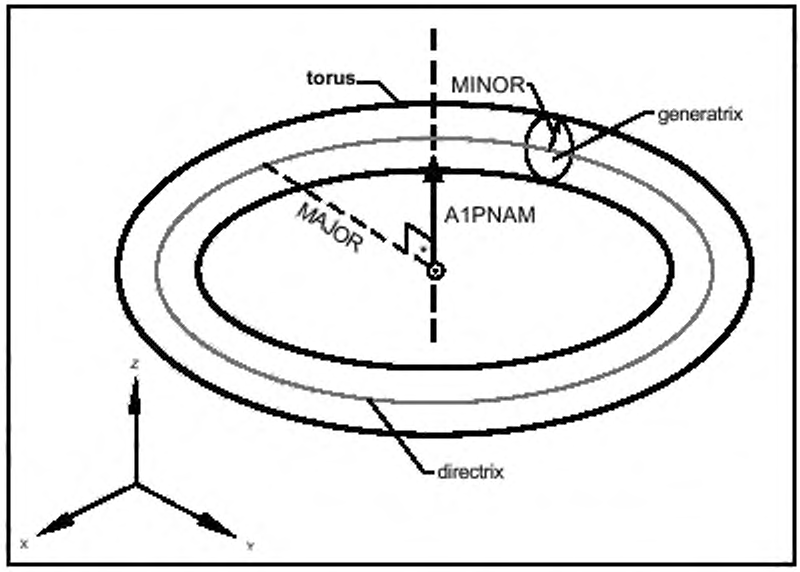
Результат использования функции

Функция создает сущность конструктивной блочной геометрии *torus*, представляющую собой тор. Он получается путем очерчивания образующей окружности радиусом MINOR вдоль направляющей окружности радиусом MAJOR. Центр и плоскость направляющей окружности определяются заданием локальной координатной системы *axis1_placement* с именем A1PNAM. Данная локальная координатная система дублируется как сущность *a1p1*, имеющая нулевой стиль *null_style*.

Затем создается экземпляр сущности *torus* с локальной координатной системой *a1p1*, радиусом направляющей окружности *major_radius*, равным MAJOR, и радиусом образующей окружности *minor_radius*, равным MINOR. Полученной сущности назначается единственный стиль представления *presentation_style_assignment*, содержащий текущие записи в таблице статуса интерфейса для стиля поверхности *surface_style* и стиля кривой *curve_style*. Функция возвращает имя полученной сущности *torus*.

Радиусы окружностей измеряются в единицах длины *OVC_length_unit* в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



Torus — тор; MINOR — радиус образующей окружности; generatrix — образующая линия; A1PNAM — локальная координатная система; MAJOR — радиус направляющей окружности; directrix — направляющая линия

Рисунок А.41 — Функция Tor_Gen

Внутренние ссылки: 6.1.9, 6.1.18.4.4, 6.2.4, 6.2.3.2, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	203	Функция несовместима с текущим уровнем интерфейса
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.6.1.5 Построение блока

Имя функции:	Уровень интерфейса:	3
Blk_Gen	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	LENX	D	Длина блока вдоль оси *X*	(EPS ≤ LENX ≤ MAX)
Ввод	LENY	D	Длина блока вдоль оси *Y*	(EPS ≤ LENY ≤ MAX)
Ввод	LENZ	D	Длина блока вдоль оси *Z*	(EPS ≤ LENZ ≤ MAX)
Ввод	A2PNAM	N	Имя локальной координатной системы *axis2_placement_3d*	a2p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *block*	blk

Привязка языка FORTRAN:
NAME = BLK_GEN (LENX, LENY, LENZ, A2PNAM, KFIX)

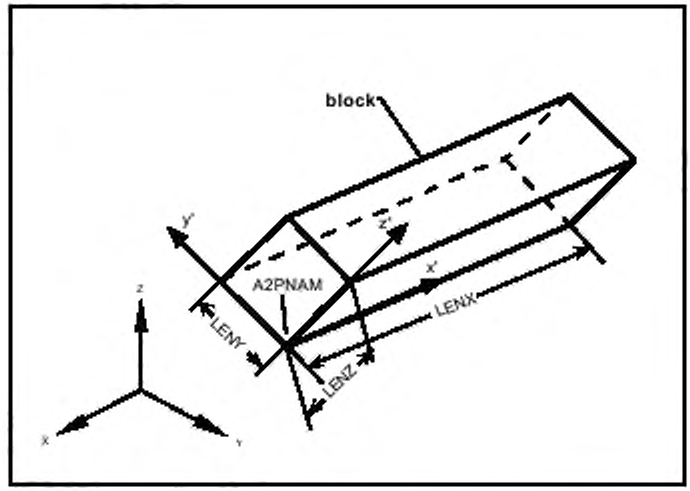
Результат использования функции

Функция создает сущность конструктивной блочной геометрии *block*, представляющую собой прямоугольный параллелепипед (кирпич). Он определяется начальной точкой и локальной координатной системой. Блок описывается положительными размерами LENX, LENY и LENZ, отложенными вдоль осей локальной координатной системы *axis2_placement_3d* с именем A2PNAM. Блок имеет одну вершину в начале локальной координатной системы. Заданная сущность *axis2_placement_3d* дублируется как сущность *a2p1*, имеющая нулевой стиль *null_style*.

Затем создается экземпляр сущности *block* с локальной координатной системой *a2p1* и длинами блока *x*, *y* и *z*, равными LENX, LENY и LENZ соответственно. Для блока назначен стиль представления *presentation_style_assignment*, содержащий текущие записи в таблице статуса интерфейса для стиля поверхности *surface_style* и стиля кривой *curve_style*. Функция возвращает имя полученной сущности *block*.

Значения длин блока измеряются в единицах длины *OVC_length_unit*. Они расположены в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



Block — блок; A2PNAM — имя локальной координатной системы, LENX — длина блока по оси X; LENY — длина блока по оси Y; LENZ — длина блока по оси Z

Рисунок А.42 — Функция Blk_Gen

Внутренние ссылки: 6.1.9, 6.1.18.4.5, 6.2.4, 6.2.3.2, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	203	Функция несовместима с текущим уровнем интерфейса
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.5.6.1.6 Построение клина

Имя функции:	Уровень интерфейса:	3
Wdg_Gen	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	LENX	D	Размер клина по оси *X*	(EPS ≤ LENX ≤ MAX)
Ввод	LENY	D	Размер клина по оси *Y*	(EPS ≤ LENY ≤ MAX)
Ввод	LENZ	D	Размер клина по оси *Z*	(EPS ≤ LENZ ≤ MAX)
Ввод	LTX	D	Вспомогательный размер клина	(0.0 или (EPS ≤ LTX ≤ MAX))
Ввод	A2PNAM	N	Имя локальной координатной системы *axis2_placement_3d*	a2p
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *right_angular_wedge*	wdg

Привязка языка FORTRAN:
NAME = WDG_GEN (LENX, LENY, LENZ, LTX, A2PNAM, KFIX)

Результат использования функции

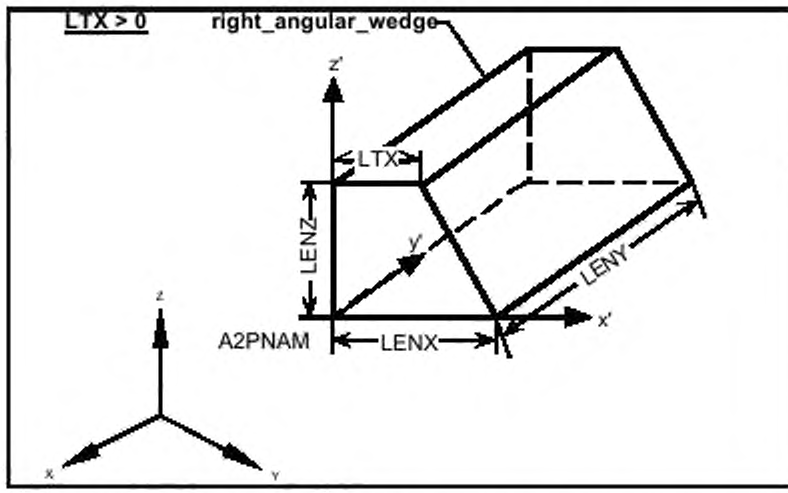
Функция создает сущность конструктивной блочной геометрии *right_angular_wedge*, представляющую собой прямой клин. Это тело получено сечением блока плоскостью, перпендикулярной одной из его граней. Грань клина получается треугольной (трапецеидальной), если размеры клина LENX и LENY вдоль осей *X* и *Y* являются положительными, если имеется вспомогательный (ненулевой) размер LTX, отложенный параллельно оси (Ox) на расстоянии LENY от начала координат, и если имеется линия, соединяющая концы сегментов LENX и LTX. Положительный размер LENZ определяет расстояние экструдирования полученного (полученной) в основании треугольника (трапеции) вдоль оси (Oz).

Локальная координатная система *axis2_placement_3d* с именем A2PNAM дублируется как сущность *a2p1*, имеющая нулевой стиль *null_style*.

Затем создается экземпляр прямого клина *right_angular_wedge* с локальной координатной системой *a2p1* и размерами LENX, LENY, LENZ и LTX. Для данной сущности *right_angular_wedge* назначен стиль представления *presentation_style_assignment*, содержащий текущие записи в таблице статуса интерфейса для стиля поверхности *surface_style* и стиля кривой *curve_style*. Функция возвращает имя полученной сущности *right_angular_wedge*.

Значения размеров LENX, LENY и LENZ определяются в единицах длины *OVC_length_unit* и находятся в диапазоне [EPS, MAX]. Вспомогательный размер LTX может быть равен 0. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Если вспомогательный размер LTX равен 0.0, то клин имеет пять граней и треугольник в основании. В противном случае он имеет шесть граней и трапецию в основании.



LTX > 0 -- вспомогательный размер больше 0, right_angular_wedge — прямой клин; LTX — вспомогательный размер, LENZ — размер клина по оси Z; LENY -- размер клина по оси Y; A2PNAM — локальная координатная система: LENX — размер клина по оси X

Рисунок А.43 — Функция Wdg_Gen

Внутренние ссылки: 6.1.9, 6.1.18.4.6, 6.2.4, 6.2.3.2, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	203	Функция несовместима с текущим уровнем интерфейса
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

A.5.6.2 Регуляризованные булевы операции конструктивной блочной геометрии

Объединение тел	Union_Sld
Пересечение тел	Intersection_Sld
Вычитание тел	Difference_Sld

A.5.6.2.1 Объединение тел

Имя функции:	Уровень интерфейса:	3
Union_Sld	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	BOPNM1	N	Имя первого операнда *boolean_operand*	solids
Ввод	BOPNM2	N	Имя второго операнда *boolean_operand*	solids
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *boolean_result*	brs

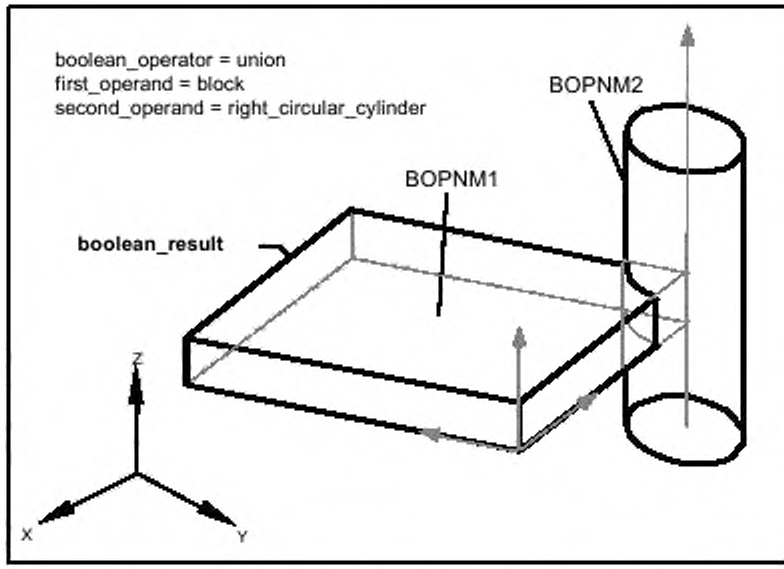
Привязка языка FORTRAN:
NAME = UNION_SLD (BOPNM1, BOPNM2, KFIX)

Результат использования функции

Функция выполняет регуляризованную булеву операцию объединения тел. Булевы операнды *boolean_operand* с именами BOPNM1 и BOPNM2 дублируются как сущности *b1* и *b2*. Указанные сущности имеют нулевой стиль *null_style*. Создается экземпляр *o* булева оператора *boolean_operator*, выполняющего объединение тел. Затем создается сущность результата выполненной булевой операции *boolean_result*, созданная из первого операнда *first_operand b1*, второго операнда *second_operand b2* и оператора объединения *o*. Для сущности *boolean_result* назначается стиль представления *presentation_style_assignment*, содержащий текущую запись в таблице статуса интерфейса для стиля поверхности *surface_style* и для стиля кривой *curve_style*. Функция возвращает имя полученной сущности *boolean_result*.

Сущность *boolean_operand* не должна содержать несоединенных частей. Объединение тел не должно быть пустым. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



Boolean_operator = union — булев оператор объединения тел; first_operand = block — первый операнд — блок; second_operand = right_circular_cylinder — второй операнд - - прямой круговой цилиндр; BOPNM2 — обозначение второго операнда; boolean_result — результат выполнения булевой операции объединения геометрических тел; BOPNM1 — обозначение первого операнда

Рисунок А.44 — Функция Union_Sld

Внутренние ссылки: 6.1.18, 6.1.18.3, 6.2.3.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
101	Попытка создания вырожденной сущности	130	Сбой при выполнении булевой операции
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

А.5.6.2.2 Пересечение тел

Имя функции:	Уровень интерфейса:	3
Intersection_Sld	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	BOPNM1	N	Имя первого операнда *boolean_operand*	solids
Ввод	BOPNM2	N	Имя второго операнда *boolean_operand*	solids
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *boolean_result*	brs

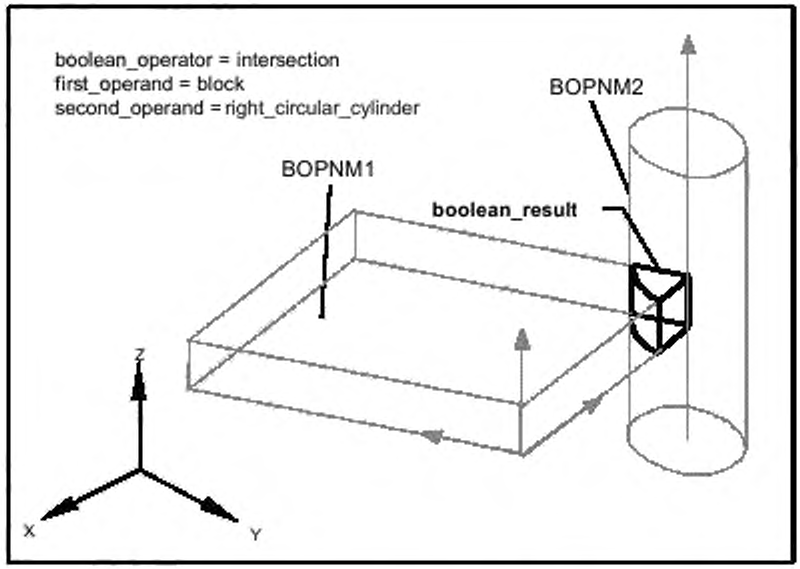
Привязка языка FORTRAN:
NAME = INTERSECTION_SLD (BOPNM1, BOPNM2, KFIX)

Результат использования функции

Функция выполняет булеву операцию регуляризованного пересечения двух тел. Булевы операнды *boolean_operand* с именами BOPNM1 и BOPNM2 дублируются как сущности *b1* и *b2*. Данные сущности имеют нулевой стиль *null_style*. Создается экземпляр *o* булева оператора *boolean_operator* пересечения тел. Далее создается сущность результата выполнения булевой операции *boolean_result*, созданная с помощью первого операнда *first_operand b1*, второго операнда *second_operand b2* и оператора их пересечения *o*. Сущности *boolean_result* назначен стиль представления *presentation_style_assignment*, содержащий текущую запись в таблице статуса интерфейса для стиля поверхности *surface_style* для стиля кривой *curve_style*. Функция возвращает имя полученного результата выполнения булевой операции *boolean_result*.

Результат булевой операции может быть получен для нескольких отдельных частей. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



Boolean_operator = intersection - - булев оператор пересечения тел; first_operand = block — первый операнд - - блок; second_operand = right_circular_cylinder — второй операнд — прямой круговой цилиндр, BOPNM2 — обозначение второго операнда, BOPNM1 — обозначение первого операнда; boolean_result — результат выполнения булевой операции объединения геометрических тел

Рисунок А.45 — Функция Intersection_Sld

Внутренние ссылки: 6.1.18, 6.1.18.3, 6.2.3.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
101	Попытка создания вырожденной сущности	130	Сбой при выполнении булевой операции
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.6.2.3 Вычитание тел

Имя функции:	Уровень интерфейса:	3
Difference_Sld	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	BOPNM1	N	Имя первого операнда *boolean_operand*	sollids
Ввод	BOPNM2	N	Имя второго операнда *boolean_operand*	solids
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *boolean_result*	brs

Привязка языка FORTRAN:
NAME = DIFFERENCE_SLD (BOPNM1, BOPNM2, KFIX)

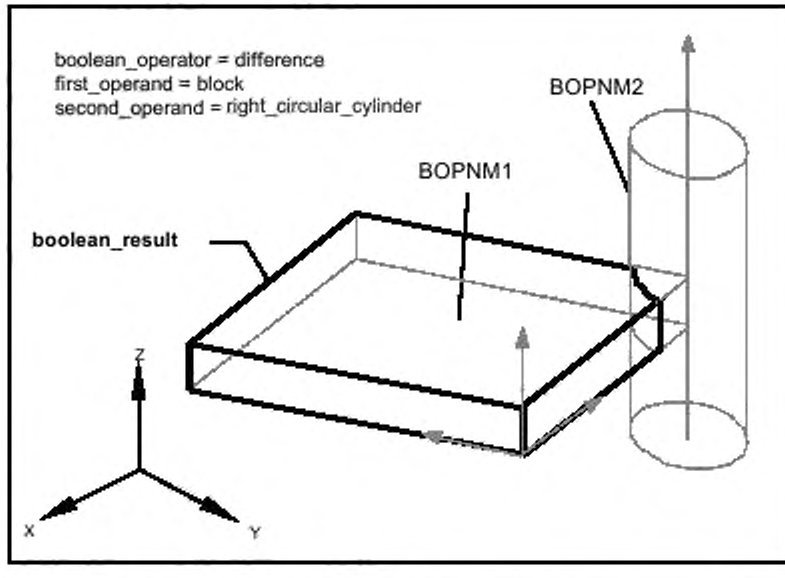
Результат использования функции

Функция выполняет булеву операцию *boolean_operand* регуляризованного вычитания двух геометрических тел. Булевы операнды *boolean_operand* с именами BOPNAM1 и BOPNM2 дублируются как сущности *b1* и *b2*. Данные сущности имеют нулевой стиль *null_style*.

Создается экземпляр *o* булева оператора *boolean_operator* вычитания геометрических тел. Затем создается сущность результата выполнения булевой операции *boolean_result* с помощью первого операнда *first_operand b1*, второго операнда *second_operand b2* и булева оператора *o*. Для сущности *boolean_result* назначен стиль представления *presentation_style_assignment*, содержащий текущую запись в таблице статуса интерфейса для стиля поверхности *surface_style* и стиля кривой *curve_style*. Функция возвращает имя результата выполнения булевой операции *boolean_result*.

Результат выполнения булевой операции может быть получен из нескольких отдельных частей. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



Boolean_operator = difference — булев оператор вычитания тел; first_operand = block — первый операнд — блок, second_operand = right_circular_cylinder — второй операнд — прямой круговой цилиндр, BOPNM2 — обозначение второго операнда; boolean_result — результат выполнения булевой операции объединения геометрических тел; BOPNM1 — обозначение первого операнда

Рисунок А.46 — Функция Difference_Sld

Внутренние ссылки: 6.1.18, 6.1.18.3, 6.2.3.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
101	Попытка создания вырожденной сущности	130	Сбой при выполнении булевой операции
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.6.3 Сущности тел, полученных путем очерчивания (сущности swept_area)

Экструдирование	Sld_Extrusion
Вращение	Sld_Revolution

A.5.6.3.1 Экструдирование

Имя функции:	Уровень интерфейса:	3
Sld_Extrusion	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	SRFNAM	N	Имя переносимой плоской поверхности *api_planar_surface*	aps
Ввод	DIRNAM	N	Направление параллельного переноса	dir
Ввод	DEPTH	D	Расстояние параллельного переноса	(EPS ≤ DEPTH ≤ MAX)
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *extruded_area_solid*	eas

Привязка языка FORTRAN:
NAME = SLD_EXTRUSION (SRFNAM, DIRNAM, DEPTH, KFIX)

Результат использования функции

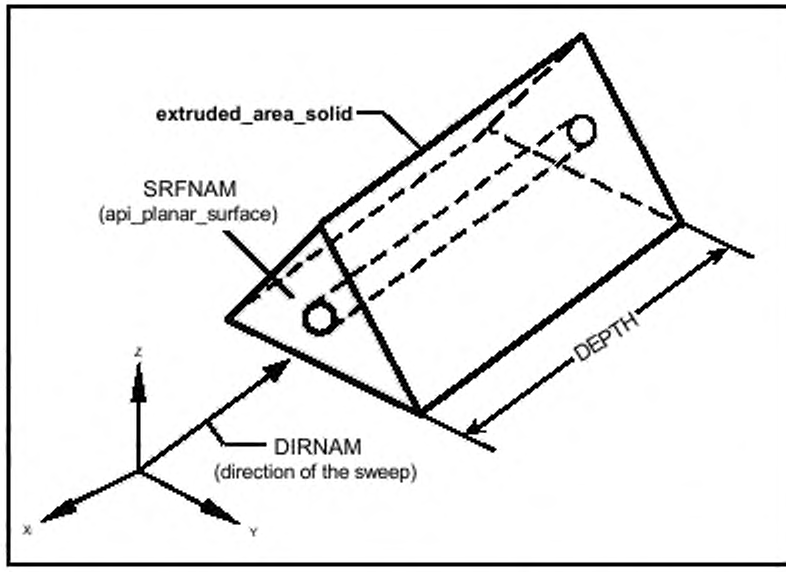
Функция создает геометрическое тело *extruded_area_solid* путем параллельного переноса и очерчивания ограниченной плоской поверхности *api_planar_surface* с именем SRFNAM заданной формы вдоль прямой. Направление параллельного переноса задается вектором направления с именем DIRNAM. Расстояние параллельного переноса определяется параметром DEPTH.

Направление DIRNAM дублируется как вектор *d*, имеющий нулевой стиль *null_style*. Затем:

- плоская поверхность *api_planar_surface* SRFNAM дублируется как сущность *s*, имеющая нулевой стиль;
- создается экземпляр экструдированного геометрического тела *extruded_area_solid* с заданной формой *s* плоской поверхности переноса *swept_area* для заданного направления *d* параллельного переноса *extruded_direction* и расстояния DEPTH параллельного переноса. Для экструдированного геометрического тела назначен стиль представления *presentation_style_assignment*, содержащий текущую запись в таблице статуса интерфейса для стиля поверхности *surface_style* и стиля кривой *curve_style*. Функция возвращает имя геометрического тела *extruded_area_solid*, полученного экструдированием.

Значение расстояния параллельного переноса DEPTH лежит в диапазоне [EPS, MAX]. Плоская поверхность переноса может иметь вырезы, которые в результате очерчивания (экструдирования, параллельного переноса) превращаются в отверстия объемного тела. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



Extruded_area_solid — геометрическое тело, полученное экструдированием, SRFNAM (api_planar_surface) — плоская поверхность с вырезом, переносимая параллельно самой себе, DEPTH — расстояние параллельного переноса; DIRNAM (direction of the sweep) — направление параллельного переноса

Рисунок А.47 — Функция Sld_Extrusion

Внутренние ссылки: 6.1.9, 6.1.17, 6.1.18.6, 6.2.3.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

А.5.6.3.2 Вращение

Имя функции:	Уровень интерфейса:	3
Sld_Revolution	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	SRFNAM	N	Имя вращаемой плоской ограниченной поверхности *api_planar_surface*	aps
Ввод	ANG	D	Угол поворота	($0° \le ANG \le 360°$)
Ввод	A1PNAM	N	Имя локальной координатной системы *axis1_placement*	a1p

Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *revolved_area_solid*	ras

Привязка языка FORTRAN:
NAME = SLD_REVOLUTION (SRFNAM, ANG, A1PNAM, KFIX)

Результат использования функции

Создание геометрического тела *revolved_area_solid* путем вращения плоской ограниченной поверхности *api_planar_surface* с именем SRFNAM вокруг заданной оси. Указанная ограниченная плоская поверхность поворачивается по часовой стрелке относительно оси A1PNAM на угол ANG.

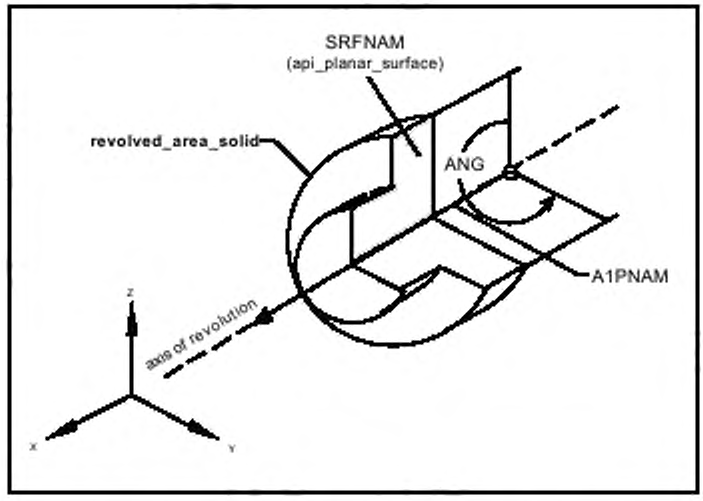
Локальная координатная система *axis1_placement* дублируется сущностью *a*. Она имеет нулевой стиль *null_style*. Затем:

- плоская поверхность *api_planar_surface* с именем SRFNAM дублируется сущностью *s*, имеющей нулевой стиль;

- создается экземпляр геометрического тела *revolved_area_solid* путем поворота плоской поверхности *swept_area* сущности *s* относительно оси *a* на угол ANG. Для полученного тела назначен стиль представления *presentation_style_assignment*, содержащий текущую запись в таблице статуса интерфейса для стиля поверхности *surface_style* и стиля кривой *curve_style*. Функция возвращает имя геометрического тела *revolved_area_solid*, полученного очерчиванием.

Угол поворота измеряется в единицах угла *OVC_angle_unit*. Он вычисляется по часовой стрелке, если смотреть вдоль оси в положительном направлении. Отсчет ведется от исходного положения вращаемой плоской области *api_planar_surface*. Вращаемая плоская область может иметь вырезы. В результате очерчивания вырезы превращаются в объемные отверстия. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Нет.



SRFNAM (api_planar_surface) — вращаемая ограниченная плоская поверхность SRFNAM (api_planar_surface); revolved_area_solid — тело, полученное вращением ограниченной плоской поверхности вокруг оси, ANG — угол поворота плоской поверхности; A1PNAM — локальная координатная система, axis of revolution — ось вращения

Рисунок А.48 — Функция Sld_Revolution

Внутренние ссылки: 6.1.9, 6.1.17, 6.1.18.7, 6.2.3.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
4	Значение меры плоского угла находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
124	Выявлено пересечение осей вращаемой плоской поверхности	126	Ось вращения не лежит во вращаемой плоской поверхности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несоместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

A.5.6.4 Построение «труб» с помощью сущностей конструктивной блочной геометрии

Построение «трубы» Sld_Pipe

A.5.6.4.1 Построение «трубы»

Имя функции:

Sld_Pipe

Уровень интерфейса:	3
Уровень геометрической мощности:	3

Параметры:

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PLNNAM	N	Имя сущности *polyline*	pln
Ввод	SRFNAM	N	Имя локальной координатной системы *api_planar_surface*	aps
Ввод	RAD	D	Радиус скругления	(EPS ≤ RAD ≤ MAX)
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя созданной сущности *boolean_result*	brs

Привязка языка FORTRAN:
NAME = SLD_PIPE (PLNNAM, SRFNAM, RAD, KFIX)

Результат использования функции

«Труба» формируется с помощью булевой операции *boolean_result* путем вращения заданной ограниченной поверхности *api_planar_surface* с именем SRFNAM вокруг направляющей, которая представляет собой скругление в виде полилинии радиусом RAD. Скругление производится автоматически. При этом проверяется возможность появления самопересечений.

Плоская поверхность *api_planar_surface* с именем SRFNAM дублируется как сущность *s*, имеющая нулевой стиль *null_style*. Полилиния с $n + 1$ точками делится на несколько прямолинейных сегментов *api_line*, обозначенных l_1, ..., l_n и соединенных дугами окружностей *api_circular_arc*, обозначенных a_1, ..., a_{n-1}, радиусом RAD. Все указанные экземпляры имеют нулевой стиль *null_style*. Затем:

- выполняется экструдирование тела e_1 (для сущности *extruded_area_solid*) с помощью плоской поверхности *s*, заданного направления параллельного переноса *extruded_direction* (для сущности *api_line.basis_curve.dir*) и расстояния параллельного переноса, полученного в результате вычитания параметра первой точки вычленения *api_line.trim[1]* и параметра второй точки вычленения *api_line.trim[2]* из сегмента l_1 (для сущности *api_line*);

- создается экземпляр *a1p1* локальной координатной системы *axis1_placement*, определяющей положение кривой a_1 (для сущности *api_circular_arc.basis_curve.position.p[3]*) и угол поворота *ang1*, полученный путем вычитания параметра первой точки вычленения *api_circular_arc.trim[1]* и параметра второй точки вычленения

api_circular_arc.trim[2] из сегмента a_1 сущности *api_circular_arc*. Сущность локальной координатной системы *axis1_placement* имеет нулевой стиль. Создается экземпляр r_1 тела, очерченного вращением *revolved_area_solid* плоской поверхности *s* относительно оси *a1p1* под углом *ang1*;

- создается экземпляр b_1 в результате выполнения булевой операции объединения *boolean_operator* первого операнда e_1 сущности *first_operand* и второго операнда r_1 сущности *second_operand*;

- для следующих сегментов прямых l_j , j = 2, …, n (сущностей *api_line*), для дуг окружности a_k (сущностей *api_circular_arc*), k = 2, …, n – 1 и i = 2, …, 2n – 2 выполняются нижеследующие операции:

1) геометрические тела e_j (сущности *extruded_area_solid*), построенные параллельным переносом плоских поверхностей *s* в направлении *extruded_direction* (сущность *api_line.basis_curve.dir*) на расстояние, равное разности значения атрибута точки вычленения 1 *api_line.trim[1]* и атрибута точки вычленения 2 *api_line.trim[2]* для сегмента l_j сущности *api_line*;

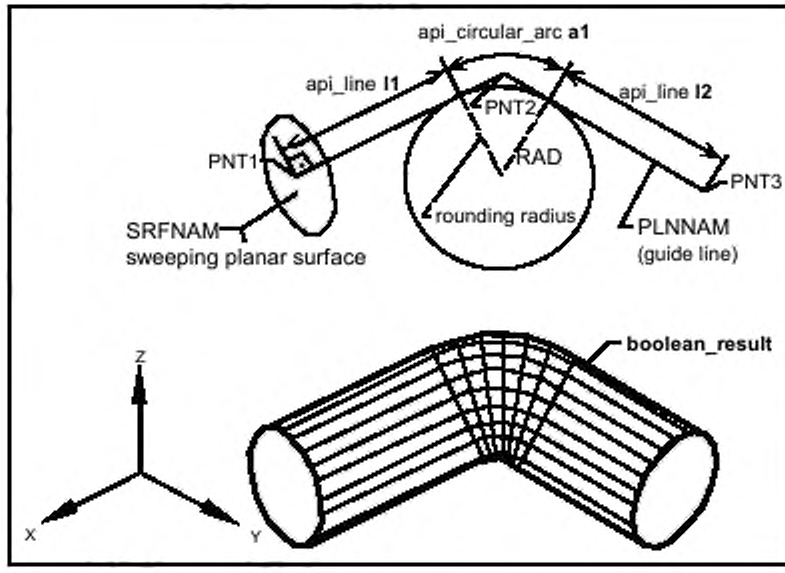
2) создается экземпляр b_i в результате выполнения булевой операции *boolean_operator* объединения первого операнда $b_{(i-1)}$ сущности *first_operand* и второго операнда e_j сущности *second_operand*;

3) создается экземпляр $a1p_k$ сущности *axis1_placement* для базовой кривой a_k (сущности *api_circular_arc.basis_curve.position.p[3]*). Угол поворота ang_k получается путем вычитания параметра первой точки вычленения *api_circular_arc.trim[1]* и параметра второй точки вычленения *api_circular_arc.trim[2]* из дуги окружности a_k (сущность *api_circular_arc*). Локальная координатная система *axis1_placement* имеет нулевой стиль. Создается экземпляр r_k тела *revolution_area_solid*, очерчиваемого путем поворота плоской поверхности *s* вокруг оси $a1p_k$ на угол ang_k;

4) создается экземпляр $b_{(i+1)}$ булевой операции *boolean_operator* объединения (с результатом *boolean_result*) первого операнда b_i сущности *first_operand* и второго операнда r_k сущности *second_operand*. Для булева результата *boolean_result* назначен стиль представления *presentation_style_assignment*, содержащий текущую запись в таблице статуса интерфейса для стиля поверхности *surface_style* и стиля кривой *curve_style*. Функция возвращает имя полученного результата булевой операции *boolean_result* (приращение индексов *i* и *j* равно 1, приращение индекса *k* равно 2).

Радиус скругления измеряется в единицах длины *OVC_length_unit*. Он расположен в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Если плоская поверхность *api_planar_surface* с именем SRFNAM (очерчиваемая вокруг направляющей) имеет форму окружности (кольца), то результатом выполнения функции интерфейса *boolean_result* является тело, называемое «труба».



Api_circular_arc a1 — дуга окружности a_1; api_line l1 — расстояние параллельного переноса l_1; api_line l2 — расстояние параллельного переноса l_2; PNT2 — точка 2; PNT1 — точка 1; RAD — радиус; PNT3 — точка 3; rounding radius — радиус скругления; SRFNAM — вращаемая ограниченная плоская поверхность SRFNAM; PLNNAM — имя кривой линии PLNNAM; sweeping planar surface — очерчивающая плоская поверхность; guide line — направляющая очерчивания; boolean_result — булев результат

Рисунок А.49 — Функция Sld_Pipe

Внутренние ссылки: 6.1.14, 6.1.17, 6.1.18, 6.2.3.2, 6.2.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
107	Попытка создания вырожденной локальной координатной системы *axis1_placement* в процессе создания сущности	109	Попытка создания вырожденного тела в процессе создания сущности
114	Попытка создания тела с перекрытием	130	Сбой при выполнении булевой операции
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
203	Функция несовместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности
1001	Перечислимое значение находится вне установленного диапазона		

А.5.6.5 Построение тела в полупространстве

Построение тела в полупространстве *half_space_solid* Hss_Gen

А.5.6.5.1 Построение тела в полупространстве (сущность half_space_solid)

Имя функции:	Уровень интерфейса:	3
Hss_Gen	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	A2PNAM	N	Имя локальной координатной системы *axis2_placement_3d*	a2p
Ввод	AGREMF	E	Значение сущности обхода контура *agreement_flag*	[TRUE, FALSE]
Вывод	NAME	N	Имя созданной сущности *half_space_solid*	hss

Привязка языка FORTRAN:
NAME = HSS_GEN (A2PNAM, AGREMF)

Результат использования функции

Сущность *half_space_solid* определяется заданным полупространством, которое является регулярным подмножеством области, лежащей с одной стороны неограниченной поверхности. Сторона поверхности, находящаяся в указанном полупространстве, определяется перпендикуляром к поверхности и значением флажка перпендикуляра. Если значение флажка равно «true», то перпендикуляр направлен от заданного полупространства. Если значение флажка равно «false», перпендикуляр направлен в заданное полупространство.

Локальная координатная система *axis2_placement_3d* дублируется как сущность *a*, имеющая нулевой стиль *null_style*. Затем:

- создается экземпляр *p* плоскости с локальной координатной системой *a*. Плоскости назначен нулевой стиль;
- создается экземпляр тела в полупространстве *half_space_solid* с базовой поверхностью *base_surface*, равной *p*, и флажком перпендикуляра *agreement_flag*, равным AGREMF. Данная сущность *half_space_solid* имеет нулевой стиль;
- функция возвращает имя полученной сущности. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечание — Геометрическая сущность тела в полупространстве *half_space_solid* не является подтипом твердотельной модели *solid_model*. Тело в полупространстве можно использовать только как операнд булева выражения *boolean_expression*. Следовательно, тело в полупространстве может храниться только в TDB.

Внутренние ссылки: 6.1.9, 6.1.16, 6.1.17.1, 6.1.18.8, 6.2.1.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	203	Функция несовместима с текущим уровнем интерфейса
204	Функция несовместима с текущим уровнем мощности	1001	Перечислимое значение находится вне установленного диапазона

А.6 Функции структурных сущностей

А.6.1 Структурные сущности в TDB

Создание группы	Create_Grp
Закрытие группы	Close_Grp
Повторное открытие группы	Reopen_Grp
Удаление сущности из группы	Remove_Ent_Grp
Сборка сущностей в новую группу	Gather_Ent_Grp
Добавление сущности в группу	Add_Ent_Grp

А.6.1.1 Создание группы

Имя функции:

Create_Grp

Уровень интерфейса:	1
Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	NAME	N	Имя созданной группы	grp

Привязка языка FORTRAN:
NAME = CREATE_GRP ( )

Результат использования функции

Функция создает новую группу *api_group* со значением атрибута имени *name*, равным пустой строке. Вновь созданная сущность *api_group* принадлежит текущей открытой группе. Она помещается в вершину стека группы и становится текущей открытой группой. Функция возвращает имя NAME вновь созданной группе *api_group*.

При возникновении ошибки группа не создается, имя группы равно 0.

Примечание — Нет.

Внутренние ссылки: 5.4.1, 6.1.19, 6.1.19.1.

Ошибки

201	Переполнение временной базы данных	204	Функция несовместима с текущим уровнем мощности
208	Превышено максимально допустимое количество групп	210	Переполнение стека группы

А.6.1.2 Закрытие группы

Имя функции:	Уровень интерфейса:	1
Close_Grp	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
—	—	—	—	—

Привязка языка FORTRAN:
CALL CLOSE_GRP ( )

Результат использования функции

Закрывается группа *api_group*, находящаяся на вершине стека группы. Настоящая группа удаляется из стека группы, при этом новой вершиной стека группы становится текущая открытая группа. При возникновении ошибки закрытие группы не происходит.

Примечание — Если текущая открытая группа является корневой группой, то возникает ошибка и никакая группа не закрывается.

Внутренние ссылки: 5.4.1, 6.1.19, 6.1.19.1.

Ошибки

204	Функция несовместима с текущим уровнем мощности	301	Попытка закрыть корневую группу

А.6.1.3 Повторное открытие группы

Имя функции:	Уровень интерфейса:	1
Reopen_Grp	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	GRPNAM	N	Имя группы *api_group*	grp

Привязка языка FORTRAN:
CALL REOPEN_GRP (GRPNAM)

Результат использования функции

Функция повторно открывает существующую группу *api_group*. Настоящая группа не должна быть текущей открытой группой одной из групп стека открытых групп. Группа *api_group* с именем GRPNAM помещается в вершину стека группы и становится текущей открытой группой. Настоящий процесс не изменяет групповую структуру. Все сущности, созданные в TDB после повторного открытия группы, принадлежат указанной группе до ее закрытия. При возникновении ошибки группа повторно не открывается.

Примечание — Нет.

Внутренние ссылки: 5.4.1,6.1.19, 6.1.19.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	210	Переполнение стека группы
302	Попытка повторно открыть уже открытую группу		

А.6.1.4 Удаление сущности из группы

Имя функции:	Уровень интерфейса:	1
Remove_Ent_Grp	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности, удаляемой из группы, *api_group*	Все графические сущности, grp

Привязка языка FORTRAN:
CALL REMOVE_ENT_GRP (ENTNAM)

Результат использования функции

Удаляется сущность (геометрическая или структурированная) из группы *api_group*, которой она принадлежит. Удаляемая сущность становится членом корневой группы. Сущность ENTNAM удаляется из множества *items* назначенной сущности *api_group_assignment* его группы *api_group*. Затем сущность размещается для хранения на множестве сущности *api_group_assignment*, принадлежащей корневой группе. При возникновении ошибки удаление сущности не происходит.

Примечание — Удаление сущности из корневой группы не допускается.

Внутренние ссылки: 5.4.1, 6.1.19, 6.1.19.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	204	Функция несовместима с текущим уровнем мощности
303	Сущность является членом корневой группы		

А.6.1.5 Сборка сущностей в новую группу

Имя функции:	Уровень интерфейса:	1
Gather_Ent_Grp	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	N	I	Длина ENTLST	≥ 1
Ввод	ENTLST	n × N	Перечень сущностей для формирования группы	Все графические сущности, grp
Вывод	NAME	N	Имя созданной сущности *api_group*	grp

Привязка языка FORTRAN:
NAME = GATHER_ENT_GRP (N, ENTLST)

Результат использования функции

Функция производит сборку перечня сущностей (геометрических или структурированных) в новую группу. Все указанные сущности удаляются из группы, которой они принадлежат, и помещаются в новую группу. Указанная новая группа должна принадлежать текущей открытой группе. Она закрывается, когда заканчивается процесс создания. Перечень сущностей, собираемых в новую группу, не должен содержать группы, включающие текущую открытую группу.

Создается экземпляр новой группы *api_group* с атрибутом имени *name*, равным пустой строке. Указанный экземпляр принадлежит текущей открытой группе. Заданные сущности удаляются из группы, которой они принадлежат, и помещаются на хранение в множество *items* сущностей *api_group_assignment*. Функция возвращает имя GRPNAM вновь созданной группе *api_group*. При возникновении ошибки никакая группа не создается, возвращаемое имя равно 0.

Примечание — Нет.

Внутренние ссылки: 5.4.1,6.1.19, 6.1.19.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
5	Целочисленное значение находится вне допустимого диапазона	201	Переполнение временной базы данных
204	Функция несовместима с текущим уровнем мощности	208	Превышение максимально допустимого количества групп
210	Переполнение стека группы	304	Сущность содержит текущую открытую группу

A.6.1.6 Добавление сущности в группу

Имя функции:	Уровень интерфейса:	1
Add_Ent_Grp	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	GRPNAM	N	Имя группы *api_group*	grp
Ввод	ENTNAM	N	Имя добавляемой сущности	Все графические сущности, grp

Привязка языка FORTRAN:
CALL ADD_ENT_GRP (GRPNAM, ENTNAM)

Результат использования функции

Функция добавляет сущность (геометрическую или структурированную) в группу GRPNAM. Сущность удаляется из множества *items* сущности *api_group_assignment*, принадлежащей ее группе *api_group*. Затем сущность ENTNAM помещается для хранения в множество *items* сущности *api_group_assignment*, принадлежащей группе *api_group* с именем GRPNAM.

Если сущность является группой, то данная группа не должна содержать группы с указанным именем. При возникновении ошибки никаких изменений не вносится.

Примечание — Нет.

Внутренние ссылки: 5.4.1, 6.1.19, 6.1.19.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	304	Сущность содержит текущую открытую группу
305	Попытка создания цикличной групповой структуры		

А.6.2 Структурные сущности, отправляемые в CAD

Открытие множества	Open_Set
Закрытие множества	Close_Set

А.6.2.1 Открытие множества

Имя функции:	Уровень интерфейса:	1
Open_Set	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	SETNAM	S	Имя множества	

Привязка языка FORTRAN:
CALL OPEN_SET (SETNAM)

Результат использования функции

Функция создает новую сущность *api_set* с атрибутом имени *name*, полученную из множества SETNAM в качестве уникального идентификатора и принадлежащую текущему открытому множеству. Сущность помещается на вершине стека множества и становится текущим открытым множеством. При возникновении ошибки множество не создается.

Примечание — Интерфейс должен гарантировать уникальность имени сущности *api_set*.

Внутренние ссылки: 5.4.2, 6.1.19, 6.1.19.3.

Ошибки

204	Функция несовместима с текущим уровнем мощности	209	Превышение максимально допустимого количества символов в строке
211	Переполнение стека множества	306	Имя множества не является уникальным

А.6.2.2 Закрытие множества

Имя функции:	Уровень интерфейса:	1
Close_Set	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
—				

Привязка языка FORTRAN:
CALL CLOSE_SET ( )

Результат использования функции

Сущность *api_set* на вершине стека множества закрывается. Данное множество удаляется из стека, новой вершиной стека становится текущее открытое множество. При возникновении ошибки множество не закрывается.

Примечания

1 Если текущая открытая сущность *api_set* является корневым множеством, то возникает ошибка и множество не закрывается.

2 Закрытое множество *api_set* не может быть открыто повторно с помощью прикладного программирования.

Внутренние ссылки: 5.4.2, 6.1.19, 6.1.19.3.

Ошибки

204	Функция несовместима с текущим уровнем мощности	307	Попытка закрыть корневое множество

А.7 Функции геометрических манипуляций с сущностями

А.7.1 Дублирующие сущности

Дублирование сущности — Dup_Ent

А.7.1.1 Дублирование сущности

Имя функции:	Уровень интерфейса:	1
Dup_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	Все графические сущности, grp
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя дублированной сущности	Тип сущности ENTNAM

Привязка языка FORTRAN:
NAME = DUP_ENT (ENTNAM, KFIX)

Результат использования функции

Функция создает новую сущность путем дублирования заданной сущности ENTNAM. Дублированная сущность содержит копии всех внутренних атрибутов исходной сущности ENTNAM и назначает тот же стиль воспроизведения. Функция возвращает имя дублированной сущности. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Внутри TDB допускаются манипуляции только с сущностями, которые являются экземплярами типа *fill_area_style_hatching* или типа *half_space_solid*. Дублированная сущность остается внутри TDB. Значение параметра KFIX, равное CAD (размещение в CAD), игнорируется.

2 Дублем сущности группы *group* является новая группа. Если сущность *group* не является пустой группой, то содержание группы дублируется один к одному.

Внутренние ссылки: 6.1, 6.2.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
208	Превышение максимально допустимого количества групп	212	Сущность может использоваться только внутри временной базы данных
1001	Перечислимое значение находится вне установленного диапазона		

А.7.2 Зеркальное отражение сущностей

Зеркальное отражение	Mirror_Ent
Дублирование и зеркальное отражение	Duplicate_Mirror_Ent

А.7.2.1 Зеркальное отражение

Имя функции:	Уровень интерфейса:	1
Mirror_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	Все графические сущности, grp
Ввод	LINNAM	N	Имя сущности прямой *api_line*	lin

Привязка языка FORTRAN:
CALL MIRROR ENT (ENTNAM, LINNAM)

Результат использования функции

Результатом использования функции является зеркальное отражение сущности относительно оси. Прямая *api_line* с именем LINNAM задает ось зеркального отражения с помощью атрибута *pnt*, атрибута *dir* и сущности *line* в качестве базовой кривой *basis_cirve*.

Если зеркально отражаемая сущность ENTNAM является сущностью конической кривой *conic* (в контексте схемы *api_abstract_schema* это может быть дуга окружности *api_circular_arc*, дуга эллипса *api_elliptical_arc*, дуга гиперболы *api_hyperbolic_arc* или дуга параболы *api_parabolic_arc*), то выполняется следующая процедура.

В случае 2D-вида:

- для сущностей конических кривых прежде всего производится зеркальное отражение положения базовой кривой *SELF\basis_curve.position*. Затем создается коническая кривая с параметрами базовой кривой с помощью зеркально отраженной локальной координатной системы *axis2_placement_2d*;
- создается экземпляр отрезка кривой *trimmed_curve* с направлением обхода *sense_agreement*, противоположным направлению обхода исходной дуги конической кривой. Параметры вычленения кривой вычисляются интерфейсом. Они гарантируют, что зеркально отраженная сущность состоит из зеркально отраженных точек исходной дуги конической кривой (см. примечание 1).

В случае 3D-вида:

- для сущностей конических кривых *conic* производится зеркальное отражение правосторонней локальной координатной системы *axis2_placement_3d* сущности *SELF\basis_curve.positionis*. Затем создается дуга конической кривой с параметрами базовой кривой *basis_curve* с помощью локальной координатной системы *axis2_placement_3d*;
- создается экземпляр отрезка кривой *trimmed_curve* с направлением обхода *sense_agreement*, совпадающим с направлением *sense_agreement* исходной дуги конической кривой. Параметры вычленения дуги вычисляются интерфейсом. При этом зеркальное отражение сущности состоит из зеркально отраженных точек исходной дуги конической кривой (см. примечание 2).

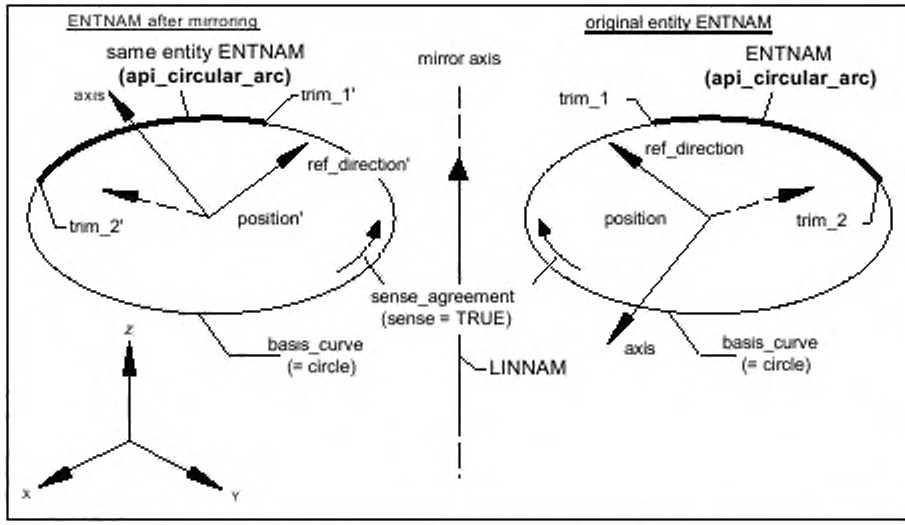
Имя зеркально отраженной сущности ENTNAM, имена характеристик визуальных представлений (например, стиля воспроизведения, уровня вида *view_level*), структуры групп и множеств сущностей не изменяются. При возникновении ошибки никакие изменения не производятся.

Примечания

1 Если точки вычленения *trim_1* и *trim_2* являются декартовыми точками *cartesian_point*, то параметры вычленения зеркально отраженных конических дуг также являются декартовыми точками, вычисляемыми по указанным точкам *trim_1* и *trim_2* в установленном порядке.

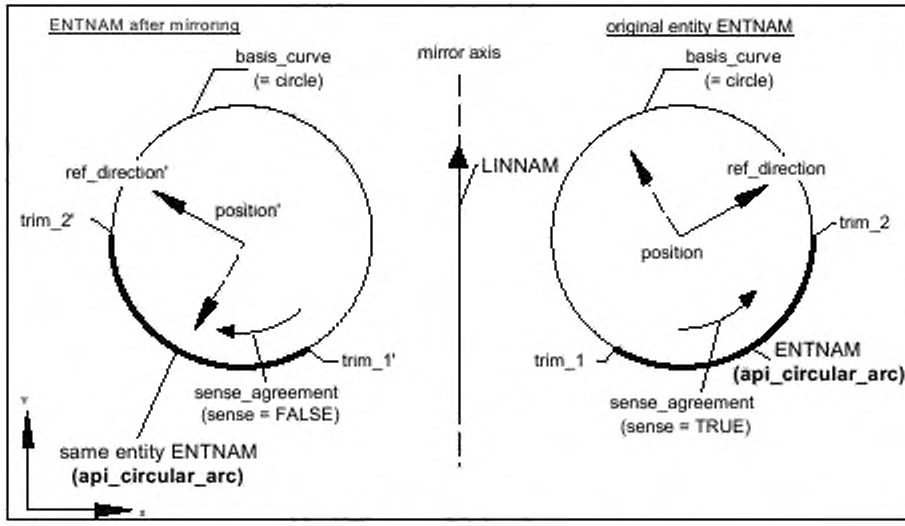
2 Направление обхода базовой кривой изменяется. Значение атрибута *sense_agreement* и параметризация кривой не изменяются.

3 Если заданная зеркально отражаемая сущность ENTNAM является экземпляром группы *api_group*, то зеркально отражаются все сущности, ссылающиеся на данную группу.



ENTNAM after mirroring — зеркально отраженная сущность; original entity ENTNAM — прообраз исходной сущности; same entity ENTNAM — зеркально отраженная дуга окружности ENTNAM; mirror axis — ось зеркального отражения; ENTNAM — исходная сущность; api_circular_arc — примитив дуги окружности; trim_1' — первая точка вычленения, trim_1 — первая точка вычленения, ref_direction' — ссылочное направление; ref_direction — ссылочное направление; position' — зеркально отраженное начало координат; position — зеркально отраженное начало координат; trim_2' — вторая точка вычленения; trim_2 — вторая точка вычленения; sense_agreement (sense = TRUE) — положительное направление обхода кривой (значение атрибута sense равно «true»), basis_curve (= circle) — базовая кривая (окружность), axis — ось LINNAM — прямая с именем LINNAM

Рисунок А.50 — Функция Mirror_Ent (3D-вид)



ENTNAM after mirroring — зеркально отраженная сущность; original entity ENTNAM — прообраз исходной сущности, basis_curve (= circle) — базовая кривая (окружность), mirror axis — ось зеркального отражения; ref_direction — ссылочное направление; LINNAM — прямая с именем LINNAM; ref_direction — ссылочное направление; trim_2' — вторая точка вычленения; position' — зеркально отраженное начало координат, trim_2 — вторая точка вычленения, position — зеркально отраженное начало координат; ENTNAM — дуга окружности ENTNAM; trim_1' — первая точка вычленения; trim_1 — первая точка вычленения; api_circular_arc — сущность дуги окружности; sense_agreement (sense = FALSE) — отрицательное направление обхода кривой (значение атрибута sense равно «false»); sense_agreement (sense = TRUE) — положительное направление обхода кривой (значение атрибута sense равно «true»), same entity ENTNAM — зеркально отраженная дуга окружности ENTNAM

Рисунок А.51 — Функция Mirror_Ent (2D-вид)

Внутренние ссылки: 6.1, 6.2.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

А.7.2.2 Дублирование и зеркальное отражение

Имя функции:	Уровень интерфейса:	1
Dup_Mirror_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	Все графические сущности, grp
Ввод	LINNAM	N	Имя сущности прямой *api_line*	lin
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя дублированной и зеркально отраженной сущности	Тип сущности ENTNAM

Привязка языка FORTRAN:
NAME = DUP_MIRROR_ENT (ENTNAM, LINNAM, KFIX)

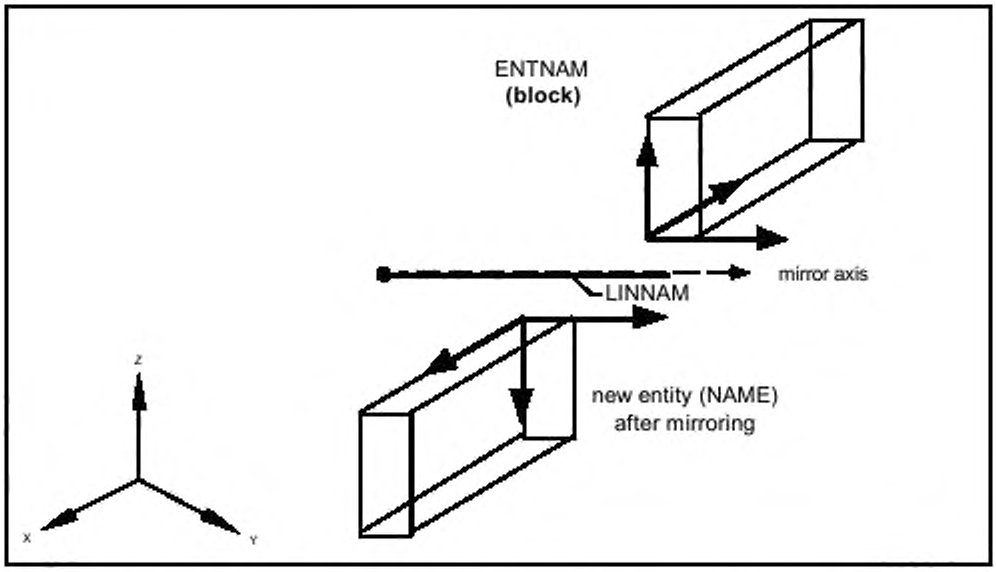
Результат использования функции

Настоящая функция осуществляется в два этапа: сначала исходная сущность дублируется функцией *Dup_Ent*, полученная сущность зеркально отображается функцией *Mirror_Ent*. Рассматриваемая функция *Dup_Mirror_Ent* возвращает имя полученной сущности. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Манипуляции с сущностями, являющимися экземплярами типа *fill_area_style_hatching* и *half_space_solid*, допустимы только внутри TDB. При этом дубликат сущности остается внутри TDB. Значение параметра KFIX, равное CAD (при этом результат отправляется в CAD), игнорируется.

2 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все сущности, содержащие ссылки на данную группу, также отражаются зеркально.



ENTNAM (block) — исходная сущность (блок); mirror axis — ось зеркального отражения; LINNAM — прямая, new entity (NAME) after mirroring — результирующая сущность с именем NAME после зеркального отражения

Рисунок А.52 — Функция Dup_Mirror_Ent

Внутренние ссылки: 6.1, 6.2.1, A.7.1.1, A.7.2.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	208	Превышение максимально допустимого количества групп
212	Сущность может использоваться только внутри временной базы данных	1001	Перечислимое значение находится вне установленного диапазона

А.7.3 Сущности сдвига

Сущность сдвига в заданном направлении	Shift_Dir_Ent
Сущность сдвига по вектору перемещения	Shift_Displacement_Ent
Сущность дублирования и сдвига в заданном направлении	Dup_Shift_Dir_Ent
Сущность дублирования и сдвига по вектору перемещения	Dup_Shift_Displacement_Ent

А.7.3.1 Сущность сдвига в заданном направлении

Имя функции:	Уровень интерфейса: 1
Shift_Dir_Ent	Уровень геометрической мощности: 1, 2, 3

Параметры

Ввод/вывод	Имя	тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	Все графические сущности, grp
Ввод	DIRNAM	N	Имя направления *direction*	dir
Ввод	SHFLEN	D	Величина сдвига	(0.0 или (EPS ≤ SHFLEN ≤ MAX))

Привязка языка FORTRAN:
CALL SHIFT_DIR_ENT (ENTNAM, DIRNAM, SHFLEN)

Результат использования функции

Функция выполняет поступательное смещение исходной сущности ENTNAM. Поступательное смещение задается направлением с именем DIRNAM и величиной сдвига SHFLEN. При этом имя сущности ENTNAM, все характеристики визуального представления (например, стиль воспроизведения, уровень вида *view_level*), структура группы и множества сущностей не изменяются. Величина сдвига SHFLEN измеряется в единицах длины *OVC_length_unit*. Она либо равна 0, либо находится в диапазоне [EPS, MAX]. При возникновении ошибки никакие изменения сущности не производятся.

Примечание — Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все сущности, ссылающиеся на данную группу, также сдвигаются.

Внутренние ссылки: 6.1, 6.2.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	204	Функция несовместима с текущим уровнем мощности

A.7.3.2 Сущность сдвига по вектору перемещения

Имя функции:	Уровень интерфейса:	1
Shift_Displacement_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	Все графические сущности, grp
Ввод	DX	D	Составляющая вектора перемещения по оси *X*	(0.0 или (EPS ≤ |DX| ≤ MAX))
Ввод	DY	D	Составляющая вектора перемещения по оси *Y*	(0.0 или (EPS ≤ |DY| ≤ MAX))
Ввод	DZ	D	Составляющая вектора перемещения по оси *Z*	(0.0 или (EPS ≤ |DZ| ≤ MAX))

Привязка языка FORTRAN:
CALL SHIFT_DISPLACEMENT_ENT (ENTNAM, DX, DY, DZ)

Результат использования функции

Функция сдвигает сущность ENTNAM из исходного положения по заданному вектору перемещения. Заданные составляющие вектора перемещения по осям измеряются в единицах длины *OVC_length_unit*. Они либо равны 0, либо находятся в диапазоне [EPS, MAX]. Имя сущности ENTNAM, все характеристики визуального представления (например, стиль воспроизведения, уровень вида *view_level*), структура группы и множества сущностей не изменяются. При возникновении ошибки никакие изменения сущности не производятся.

Примечания

1 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все сущности, ссылающиеся на данную группу, также сдвигаются.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то заданная составляющая вектора смещения *DZ* игнорируется интерфейсом.

Внутренние ссылки: 6.1, 6.2.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	204	Функция несовместима с текущим уровнем мощности

A.7.3.3 Сущность дублирования и сдвига в заданном направлении

Имя функции:	Уровень интерфейса:	1
Dup_Shift_Dir_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя исходной сущности	Все графические сущности, grp
Ввод	DIRNAM	N	Имя направления *direction*	dir
Ввод	SHFLEN	D	Величина сдвига	(0.0 или (EPS ≤ SHFLEN ≤ MAX))
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя дублированной и сдвинутой сущности	Тип сущности ENTNAM

Привязка языка FORTRAN:
NAME = DUP_SHIFT_DIR_ENT (ENTNAM, DIRNAM, SHFLEN, KFIX)

Результат использования функции

Настоящая функция осуществляется в два этапа: сначала производится дублирование сущности с помощью функции Dup_Ent, затем полученная сущность сдвигается в заданном направлении с помощью функции Shift_Dir_Ent. Рассматриваемая функция возвращает имя полученной сущности. Величина сдвига SHFLEN измеряется в единицах длины *OVC_length_unit*. Она либо равна 0, либо находится в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Если заданная величина сдвига равна 0, то в результате выполнения функции Dup_Ent получается только копия сущности ENTNAM.

2 Манипуляции с сущностями допустимы только внутри TDB, если они являются экземплярами типа *fill_area_style_hatching* или *half_space_solid*. Дублированнная сущность остается внутри TDB. Параметр KFIX, равный CAD (результат функции направляется в CAD), игнорируется.

3 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все сущности, ссылающиеся на данную группу, также сдвигаются.

Внутренние ссылки: 6.1, 6.2.1, A.7.1.1, A.7.3.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
3	Значение меры длины находится вне допустимого диапазона	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
208	Превышение максимально допустимого количества групп	212	Сущность может быть использована только внутри временной базы данных
1001	Перечислимое значение находится вне установленного диапазона		

А.7.3.4 Сущность дублирования и сдвига по вектору перемещения

Имя функции:	Уровень интерфейса:	1
Dup_Shift_Displacement_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя исходной сущности	Все графические сущности, grp
Ввод	DX	D	Составляющая вектора перемещения по оси *X*	(0.0 или (EPS ≤ |DX| ≤ MAX))
Ввод	DY	D	Составляющая вектора перемещения по оси *Y*	(0.0 или (EPS ≤ |DY| ≤ MAX))
Ввод	DZ	D	Составляющая вектора перемещения по оси *Z*	(0.0 или (EPS ≤ |DZ| ≤ MAX))
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя дублированной и сдвинутой сущности	Тип сущности ENTNAM

Привязка языка FORTRAN:
NAME = DUP_SHIFT_DISPLACEMENT_ENT (ENTNAM, DX, DY, DZ, KFIX)

Результат использования функции

Настоящая функция осуществляется в два этапа: сначала выполняется дублирование сущности с помощью функции Dup_Ent, затем полученный дубликат сдвигается с помощью функции *Shift_Displacement_Ent*. Рассматриваемая функция возвращает имя сдвинутой сущности. Заданное значение сдвига измеряется в *OVC_length_unit*. Оно либо равно 0, либо находится в диапазоне [EPS, MAX]. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Если евклидова норма сдвига равна 0, то результатом использования функции Dup_Ent будет только копия сущности ENTNAM.

2 Манипуляции с сущностями, являющимися экземплярами типа *fill_area_style_hatching* или *half_space_solid*, допускаются только внутри TDB. Дублированная сущность остается внутри TDB. Значение параметра KFIX, равное CAD (результат вычисления функции направляется в CAD), игнорируется интерфейсом.

3 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все сущности, ссылающиеся на данную группу, также дублируются и сдвигаются.

4 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то заданная составляющая смещения *DZ* игнорируется интерфейсом.

Внутренние ссылки: 6.1, 6.2.1, А.7.1.1, А.7.3.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	3	Значение меры длины находится вне допустимого диапазона
201	Переполнение временной базы данных	202	Ошибка при отправке сущности в CAD
204	Функция несовместима с текущим уровнем мощности	208	Превышение максимально допустимого количества групп
212	Сущность может быть использована только внутри временной базы данных	1001	Перечислимое значение находится вне установленного диапазона

A.7.4 Сущности вращения

Вращение — Rotate_Ent

Дублирование и вращение — Dup_Rotate_Ent

A.7.4.1 Вращение

Имя функции:	Уровень интерфейса:	1
Rotate_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя исходной сущности	Все графические сущности, grp
Ввод	PNTNAM	N	Имя указателя *cartesian_point* для вращения	pnt
Ввод	ANG1	D	Угол поворота в плоскости (Oxy) вокруг оси (Oz)	(–360° ≤ ANG1 ≤ 360°)
Ввод	ANG2	D	Угол поворота в плоскости (Oyz) вокруг оси (Ox)	(–360° ≤ ANG2 ≤ 360°)
Ввод	ANG3	D	Угол поворота в плоскости (Ozx) вокруг оси (Oy)	(–360° ≤ ANG3 ≤ 360°)

Привязка языка FORTRAN:
CALL ROTATE_ENT (ENTNAM, PNTNAM, ANG1, ANG2, ANG3)

Результат использования функции

Производится поворот сущности ENTNAM. Указатель *cartesian_point* начала координат виртуальной координатной системы, в которой производится вращение. Данная виртуальная координатная система определена путем параллельного переноса текущей ссылочной координатной системы OVC в PNTNAM. Вращение производится в следующем порядке:

1) поворот вокруг оси (Oz);
2) поворот вокруг оси (Ox);
3) поворот вокруг оси (Oy).

Углы измеряются в единицах угла *OVC_angle_unit*. Имя сущности ENTNAM, все характеристики визуального представления (например, стиль воспроизведения, уровень вида *view_level*), структура группы и множества сущностей не изменяются. При возникновении ошибки никакие изменения сущностей не производятся.

Примечания

1 Если вычисленное смещение сущности из исходного положения по указателю вращения находится в диапазоне [*ZERO_value*, EPS], то никакие модификации не производятся и ошибки не возникают.

2 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все сущности, ссылающиеся на данную группу, также должны быть повернуты.

3 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то заданные углы ANG2 и ANG3 поворота относительно осей игнорируются интерфейсом. Производится только вращение в плоскостях *x*, *y* вокруг PNTNAM.

Внутренние ссылки: 6.1, 6.2.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
4	Значение меры плоского угла находится вне допустимого диапазона	204	Функция несовместима с текущим уровнем мощности

A.7.4.2 Дублирование и вращение

Имя функции:	Уровень интерфейса:	1
Dup_Rotate_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	Все графические сущности, grp
Ввод	PNTNAM	N	Имя указателя *cartesian_point* для вращения	pnt
Ввод	ANG1	D	Угол поворота в плоскости (Oxy) вокруг оси (Oz)	(–360° ≤ ANG1 ≤ 360°)
Ввод	ANG2	D	Угол поворота в плоскости (Oyz) вокруг оси (Ox)	(–360° ≤ ANG2 ≤ 360°)
Ввод	ANG3	D	Угол поворота в плоскости (Ozx) вокруг оси (Oy)	(–360° ≤ ANG3 ≤ 360°)
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]
Вывод	NAME	N	Имя дублированной и повернутой сущности	Тип сущности ENTNAM

Привязка языка FORTRAN:
NAME = DUP_ROTATE_ENT (ENTNAM, PNTNAM, ANG1, ANG2, ANG3, KFIX)

Результат использования функции

Настоящая функция осуществляется в два этапа: сначала выполняется параллельный перенос с помощью функции *Dup_Ent*. Затем полученная сущность поворачивается с помощью функции *Rotate_Ent*. Функция возвращает имя сущности, полученной параллельным переносом. При возникновении ошибки сущность не создается, функция возвращает нулевое имя элемента.

Примечания

1 Если вычисленное смещение из исходного положения сущности по указателю вращения находится в диапазоне [*ZERO_value*, EPS], то в результате выполнения функции *Dup_Ent* создается только копия сущности ENTNAM.

2 Манипуляции с сущностями, являющимися экземплярами типа *fill_area_style_hatching* или *half_space_solid*, допускаются только внутри TDB. Дублированная сущность остается внутри TDB. Значение параметра KFIX, равное CAD (результат вычисления функции отправляется в CAD), игнорируется интерфейсом.

3 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все сущности, ссылающиеся на данную группу, также дублируются и сдвигаются.

4 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то заданные углы ANG2 и ANG3 поворота относительно осей игнорируются интерфейсом. Производится только вращение в плоскостях *x*, *y* вокруг PNTNAM.

Внутренние ссылки: 6.1,.6.2.1, A.7.1.1, A.7.4.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
4	Значение меры плоского угла находится вне допустимого диапазона	201	Переполнение временной базы данных
202	Ошибка при отправке сущности в CAD	204	Функция несовместима с текущим уровнем мощности
208	Превышение максимально допустимого количества групп	212	Сущность может быть использована только внутри временной базы данных
1001	Перечислимое значение находится вне установленного диапазона		

A.7.5 Изменения сущностей

Изменение направления сущностей кривой	Chg_orientation_Ent
Изменение положительного направления обхода круговой или эллиптической сущности	Chg_Sense_Ent
Сущность гомотетии	Homotetia_Ent

A.7.5.1 Изменение направления сущностей кривой

Имя функции:	Уровень интерфейса:	1
Chg_Orientation_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	curves

Привязка языка FORTRAN:
CALL CHG_ORIENTATION_ENT (ENTNAM)

Результат использования функции

Функция изменяет направление сущности кривой ENTNAM. Если заданная сущность лежит в диапазоне базовых или конических кривых, то выполняется следующая процедура:

- меняются местами две точки вычленения *trim_1* и *trim_2*;
- изменяется положительное направление обхода кривой *sense_agreement*.

Если заданная сущность кривой ENTNAM является экземпляром типа полилинии *polyline*, то выполняется следующая процедура:

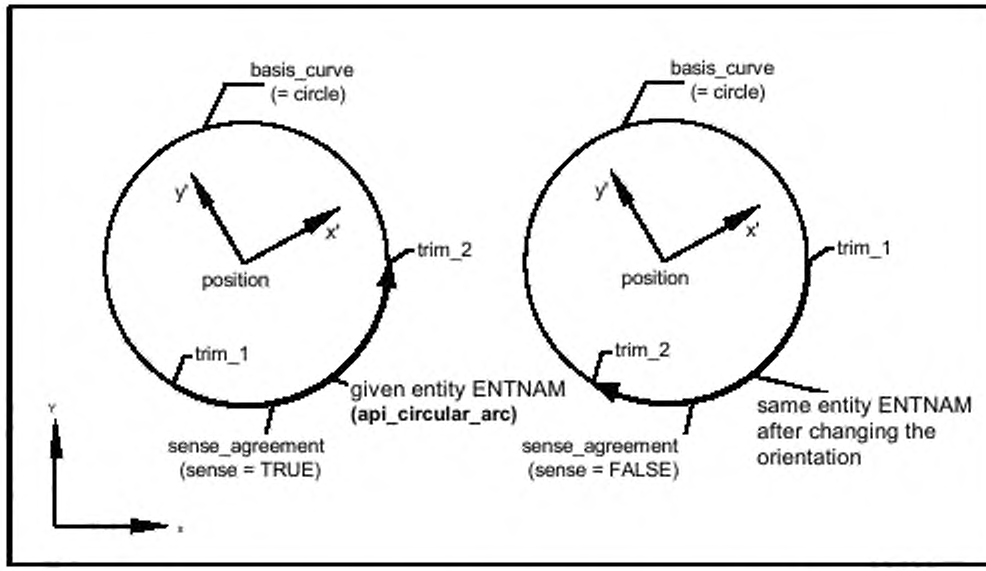
- порядок перечня точек полилинии *cartesian_point* изменяется на противоположный.

Если заданная сущность кривой ENTNAM является экземпляром типа контур *api_contour*, то выполняется нижеследующая процедура:

- каждый элемент перечня сегментов комбинированной кривой *composite_curve_segment* изменяется в соответствии с процедурой, описанной выше;
- порядок перечня сегментов изменяется на противоположный.

Имя сущности ENTNAM, все характеристики визуального представления (например, стиль воспроизведения, уровень вида *view_level*), структура группы и множества сущностей не изменяются. При возникновении ошибки изменения сущностей не производятся.

Примечание — Нет.



Basis_curve (= circle) — базовая кривая (окружность); trim_2 — вторая точка вычленения; trim_1 — первая точка вычленения; position — центр окружности, given entity ENTNAM (api_circular_arc) — исходная сущность ENTNAM (дуга окружности); same entity ENTNAM after changing the orientation — исходная сущность ENTNAM (дуга окружности) после смены положения; sense_agreement (sense = TRUE) — положительное направление обхода кривой; sense_agreement (sense = FALSE) — отрицательное направление обхода кривой

Рисунок А.53 — Функция Chg_Orientation_Ent

Внутренние ссылки: 6.1.12, 6.1.13, 6.1.14, 6.2.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

А.7.5.2 Изменение положительного направления обхода круговой или эллиптической сущности

Имя функции:	Уровень интерфейса:	1
Chg_Sense_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

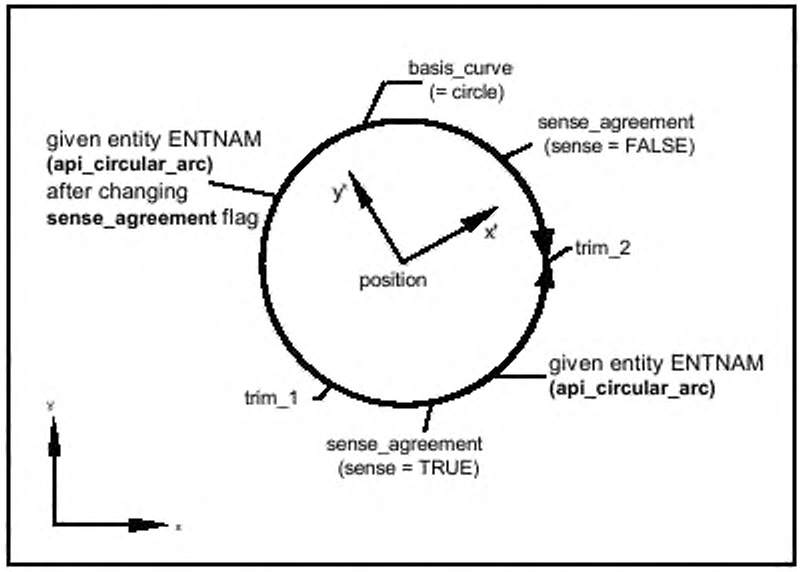
Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	arc, elc

Привязка языка FORTRAN:
CALL CHG_SENSE_ENT (ENTNAM)

Результат использования функции

Функция изменяет значение флажка *sense_agreement* (направления обхода дуги окружности *api_circular_arc* или дуги эллипса *api_elliptical_arc* с именем ENTNAM) на противоположное. Имя сущности ENTNAM, все характеристики визуального представления (например, стиля воспроизведения, уровня вида *view_level*), структуры группы и множества сущностей не изменяются. При возникновении ошибки сущности не изменяются.

Примечание — Настоящая функция использует либо дугу окружности *api_circular_arc*, либо дугу эллипса *api_elliptical_arc*.



Basis_curve (= circle) — базовая кривая (окружность); sense_agreement (sense = FALSE) — отрицательное направление обхода контура (значение флажка равно «false»); given entity ENTNAM (api_circular_arc) after changing sense_agreement flag — заданная сущность ENTNAM (дуга окружности) после изменения значения флажка направления обхода контура; trim_2 — вторая точка вычленения; position — центр окружности; given entity ENTNAM (api_circular_arc) — заданная сущность ENTNAM (дуга окружности), trim_1 — первая точка вычленения; sense_agreement (sense = TRUE) — положительное направление обхода контура (значение флажка равно «true»)

Рисунок А.54 — Функция Chg_Sense_Ent

Внутренние ссылки: 6.1.12.2, 6.1.13.1, 6.2.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

А.7.5.3 Сущность гомотетии

Имя функции:	Уровень интерфейса:	1
Homotetia_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	curves, solid_model, grp
Ввод	PNTNAM	N	Имя декартовой точки *cartesian_point*	pnt
Ввод	K	D	Коэффициент гомотетии	(EPS ≤ |K| ≤ MAX)

Привязка языка FORTRAN:
CALL HOMOTETIA_ENT (ENTNAM, PNTNAM, K)

Результат использования функции

Функция изменяет сущность ENTNAM с помощью преобразования гомотетии. Для этого задается центр *C* с именем PNTNAM и коэффициент гомотетии *K*. В процессе преобразования каждая точка *P*, принадлежащая

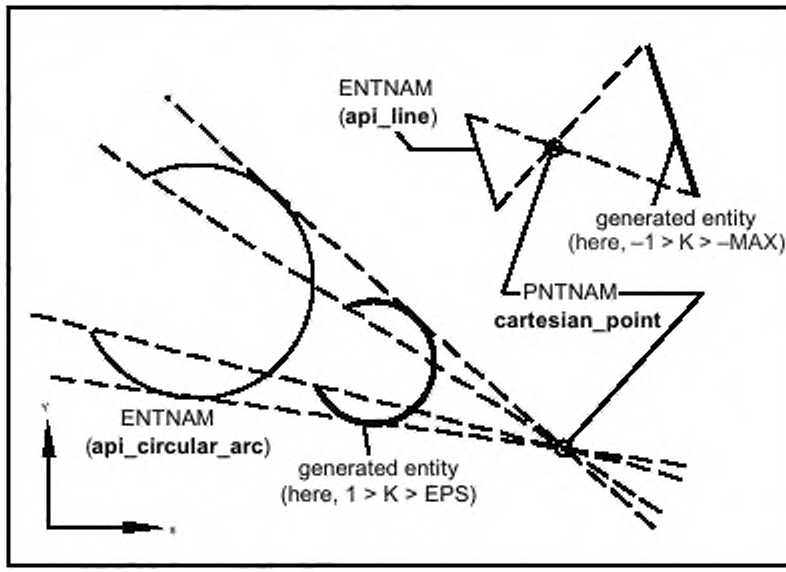
исходной сущности ENTNAM, переходит в точку *P'* по правилу *CP' = K × CP* (где жирным шрифтом выделены векторы *CP'* и *CP*).

Имя сущности ENTNAM, все характеристики визуального представления (например, стиль воспроизведения, уровень вида *view_level*), структура группы и множества сущностей не изменяются. При возникновении ошибки никакие изменения не производятся.

Примечания

1 Данная функция интерфейса гарантирует, что построение вырожденных сущностей не производится.

2 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все сущности, ссылающиеся на данную группу, подвергаются преобразованию гомотетии.



ENTNAM (api_line) — исходная сущность ENTNAM (отрезок); generated entity (here, -1 > K > -MAX) — полученная сущность (здесь -1 > K > -MAX); PNTNAM cartesian_point — центр гомотетии PNTNAM; ENTNAM — исходная сущность; api_circular_arc — исходная сущность (дуга окружности); generated entity (here, 1 > K > EPS) — полученная сущность (здесь -1 > K > -EPS)

Рисунок А.55 — Функция Homotetia_Ent

Внутренние ссылки: 6.1, 6.2.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
7	Действительное значение находится вне допустимого диапазона	101	Попытка создания вырожденной сущности
204	Функция несовместима с текущим уровнем мощности		

А.8 Утилиты

Утилиты позволяют прикладным программам получать характеристики сущностей путем вычислений, проводимых интерфейсом, с учетом ограничений.

А.8.1 Утилиты геометрических сущностей

Определение декартовых координат точки	Pnt_Retrieve_Coordinate
Определение компонент вектора направления	Dir_Retrieve_component

Определение начала локальной координатной системы *axis2_placement*	A2p_Retrieve_Location
Определение компонент вектора направления по отрезку	Lin_Retrieve_Dir
Определение размещения тела в полупространстве	Hss_Retrieve_A2p
Определение размещения дуги окружности	Arc_Retrieve_A2p
Определение радиуса дуги окружности	Arc_Retrieve_Rad
Определение направления обхода дуги окружности (эллипса)	Arc_Retrieve_Sense

A.8.1.1 Определение декартовых координат точки

Имя функции:	Уровень интерфейса:	1
Pnt_Retrieve_Coordinate	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PNTNAM	N	Имя декартовой точки *cartesian_point*	pnt
Вывод	X	D	Координата *X* заданной декартовой точки с именем PNTNAM	(0.0 или (EPS ≤ |Z| ≤ MAX))
Вывод	Y	D	Координата *Y* заданной декартовой точки с именем PNTNAM	(0.0 или (EPS ≤ |Z| ≤ MAX))
Вывод	Z	D	Координата *Z* заданной декартовой точки с именем PNTNAM	(0.0 или (EPS ≤ |Z| ≤ MAX))

Привязка языка FORTRAN:
CALL PNT_RETRIEVE_COORDINATE (PNTNAM, X, Y, Z)

Результат использования функции

Функция извлекает декартовы координаты заданного экземпляра сущности *cartesian_point* с именем PNTNAM. Полученные координаты измеряются в единицах длины *OVC_length_unit* и относятся к базовой координатной системе OVC. При необходимости интерфейс преобразуется к текущей OVC. При возникновении ошибки (например, если значение координаты больше максимально допустимого значения MAX), значения координат точки *X*, *Y* и *Z* не определяются.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то значение координаты *Z* точки равно 0.

Внутренние ссылки: 6.1.9.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

A.8.1.2 Определение компонент вектора направления

Имя функции:	Уровень интерфейса:	1
Dir_Retrieve_Component	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	DIRNAM	N	Имя сущности *direction*	dir
Вывод	X	D	Составляющая *X* вектора направления по оси (Ox)	
Вывод	Y	D	Составляющая *Y* вектора направления по оси (Oy)	
Вывод	Z	D	Составляющая *Z* вектора направления по оси (Oz)	

Привязка языка FORTRAN:
CALL PNT_RETRIEVE_COMPONENT (DIRNAM, X, Y, Z)

Результат использования функции

Функция извлекает компоненты вектора направления *direction_ratio* из заданного экземпляра сущности *direction* с именем DIRNAM. Полученные значения компонент *direction_ratio* относятся к базовой координатной системе OVC. При возникновении ошибки (например, если компонента превышает максимально допустимое значение MAX) значения компонент *X*, *Y* и *Z* не определены.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то значение составляющей *Z* вектора направления равно 0.

Внутренние ссылки: 6.1.9.3.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

A.8.1.3 Определение начала локальной координатной системы axis2_placement

Имя функции:	Уровень интерфейса:	1
A2p_Retrieve_Location	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	A2PNAM	N	Имя локальной координатной системы *axis2_placement*	a2p
Вывод	PNTNAM	N	Имя начала координат	pnt

Привязка языка FORTRAN:
CALL A2P_RETRIEVE_LOCATION (A2PNAM, PNTNAM)

Результат использования функции

Функция извлекает начало координат заданного экземпляра A2PNAM локальной координатной системы *axis2_placement*. Полученная декартова точка *cartesian_point* относится к базовой координатной системе OVC. При возникновении ошибки значение сущности PNTNAM равно 0.

Примечание — Нет.

Внутренние ссылки: 6.1.9, 6.1.9.2, 6.1.9.7, 6.1.9.8.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

A.8.1.4 Определение компонент вектора направления по отрезку

Имя функции:	Уровень интерфейса:	1
Lin_Retrieve_Dir	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	LINNAM	N	Имя сущности отрезка прямой *api_line*	lin
Вывод	DIRNAM	N	Имя сущности направления *direction*	dir

Привязка языка FORTRAN:
CALL LIN_RETRIEVE_DIR (LINNAM, DIRNAM)

Результат использования функции

Функция извлекает направление из заданного экземпляра LINNAM отрезка *api_line*. Утилита возвращает имя сущности *direction*, построенной на отрезке *api_line*. При возникновении ошибки значение сущности DIRNAM равно 0.

Примечание — Нет.

Внутренние ссылки: 6.1.9, 6.1.9.3, 6.1.12.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

A.8.1.5 Определение размещения тела в полупространстве

Имя функции:	Уровень интерфейса:	3
Hss_Retrieve_A2p	Уровень геометрической мощности:	3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	HSSNAM	N	Имя тела в полупространстве *half_space_solid*	hss
Вывод	A2PNAM	N	Имя локальной координатной системы *axis2_placement*	a2p

Привязка языка FORTRAN:
CALL HSS_RETRIEVE_A2P (HSSNAM, A2PNAM)

Результат использования функции

Функция определяет положение и направление заданного экземпляра HSSNAM сущности *half_space_solid*. Утилита возвращает имя сущности положения как A2PNAM. Полученная локальная координатная система

axis2_placement относится к базовой координатной системе OVC. При возникновении ошибки значение сущности A2PNAM равно 0.

Примечание — Нет.

Внутренние ссылки: 6.1.9, 6.1.9.7, 6.1.9.8, 6.1.18.8.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
203	Функция несовместима с текущим уровнем интерфейса	204	Функция несовместима с текущим уровнем мощности

A.8.1.6 Определение размещения дуги окружности

Имя функции:	Уровень интерфейса:	1
Arc_Retrieve_A2p	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ARCNAM	N	Имя дуги окружности *api_circular_arc*	arc
Вывод	A2PNAM	N	Имя локальной координатной системы *axis2_placement*	a2p

Привязка языка FORTRAN:
CALL ARC_RETRIEVE_A2P (ARCNAM, A2PNAM)

Результат использования функции

Функция определяет положение и направление по заданному экземпляру ARCNAM дуги окружности *api_circular_arc*. Утилита возвращает имя сущности положения базовой кривой *basis_curve* как A2PNAM. Полученная локальная координатная система *axis2_placement* относится к базовой координатной системе OVC. При возникновении ошибки значение сущности A2PNAM равно 0.

Примечание — Нет.

Внутренние ссылки: 6.1.9, 6.1.9.7, 6.1.9.8, 6.1.12.2.

Ошибки:

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

A.8.1.7 Определение радиуса дуги окружности

Имя функции:	Уровень интерфейса:	1
Arc_Retrieve_Rad	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ARCNAM	N	Имя дуги окружности *api_circular_arc*	arc
Вывод	RADIUS	D	radius	(EPS ≤ RAD ≤ MAX)

Привязка языка FORTRAN:
CALL ARC_RETRIEVE_RAD (ARCNAM, RADIUS)

Результат использования функции

Функция определяет радиус заданного экземпляра ARCNAM дуги окружности *api_circular_arc*. Утилита возвращает значение радиуса базовой кривой *basis_curve* как переменную RADIUS. При возникновении ошибки значение RADIUS равно 0.

Примечание — Нет.

Внутренние ссылки: 6.1.12.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

A.8.1.8 Определение направления обхода дуги окружности (эллипса)

Имя функции:	Уровень интерфейса:	1
Arc_Retrieve_Sense	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ARCNAM	N	Имя сущности *api_circular_arc* или сущности *api_elliptical_arc*	arc, elc
Вывод	SENSE	E	Значение атрибута *sense_agreement_flag*	[TRUE, FALSE, UNKNOWN]

Привязка языка FORTRAN:
CALL ARC_RETRIEVE_SENSE (ARCNAM, SENSE)

Результат использования функции

Функция доставляет информацию о значении флажка направления обхода заданного экземпляра ARCNAM дуги окружности *api_circular_arc* или дуги эллипса *api_elliptical_arc*. Утилита возвращает логическое значение (TRUE или FALSE) атрибута SENSE сущности *sense_agreement* базовой кривой *basis_curve*. При возникновении ошибки значение данного атрибута сущности неизвестно (UNKNOWN).

Примечание — Нет.

Внутренние ссылки: 6.1.12.2, 6.1.13.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

A.8.2 Утилиты запроса

Запрос типа сущности	Retrieve_Type_Ent
Запрос перечня членов группы	Retrieve_Member_Grp
Запрос перечня сущностей контура	Retrieve_Ent_Ctr

А.8.2.1 Запрос типа сущности

Имя функции:	Уровень интерфейса:	1
Retrieve_Type_Ent	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности	Все типы сущностей
Вывод	TYPE	S	Строка, содержащая 3 символа аббревиатуры типа сущности	См. таблицу А.5

Привязка языка FORTRAN:
CALL RETRIEVE_TYPE_ENT (ENTNAM, TYPE)

Результат использования функции

Функция определяет тип экземпляра существующей сущности ENTNAM. Утилита возвращает тип сущности в виде аббревиатуры из 3 символов (см. таблицу А.5).

Таблица А.5 — Аббревиатуры типов сущностей ENTNAM

Аббревиатура в полученной строке	Соответствующий тип экземпляра сущности
a1p	Локальная координатная система *axis1_placement*
a2p	Локальная координатная система *axis2_placement*
afa	Заполненная область *annotation_fill_area*
aps	Плоская поверхность *api_planar_surface* интерфейса прикладного программирования
arc	Дуга окружности *api_circular_arc* интерфейса прикладного программирования
blk	Блок *block*
brs	Результат выполнения булевой операции *boolean_result*
con	Прямой круговой конус *right_circular_cone*
ctr	Контур *api_contour* интерфейса прикладного программирования
cyl	Прямой круговой цилиндр *right_circular_cylinder*
dir	Направление *direction*
eas	Экструдированное геометрическое тело *extruded_area_solid*
elc	Дуга эллипса *api_elliptical_arc* интерфейса прикладного программирования
fsh	Стиль штриховки заполненной области *fill_area_style_hatching*
grp	Группа *api_group* интерфейса прикладного программирования
hss	Тело в полупространстве *half_space_solid*
hyp	Дуга гиперболы *api_hyperbolic_arc* интерфейса прикладного программирования
lin	Прямая линия *api_line* интерфейса прикладного программирования
par	Дуга параболы *api_parabolic_arc* интерфейса прикладного программирования
pln	Полилиния *polyline*
pnt	Декартова точка *cartesian_point*

Окончание таблицы А.5

ras	Геометрическое тело, полученное очерчиванием образующего контура вдоль направляющей *revolved_area_solid*
sph	Сфера *sphere*
tor	Тор *torus*
wdg	Прямой клин *right_angular_wedge*

Примечания

1 Если возникает ошибка, то полученная строка содержит три символа «звездочка» (***).

2 Количество символов в полученной строке всегда равно 3.

Внутренние ссылки: 5.1.3.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	204	Функция несовместима с текущим уровнем мощности
1003	Неправильное количество символов в строке		

А.8.2.2 Запрос перечня членов группы

Имя функции:	Уровень интерфейса:	1
Retrieve_Member_Grp	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	GRPNAM	N	Имя группы *api_group*	grp
Вывод	N	I	Длина перечня ENTLST, определяющего количество сущностей	≥ 0
Вывод	ENTLST	nxN	Перечень имен сущностей — членов группы *api_group*	Графические сущности, grp

Привязка языка FORTRAN

Ввод/вывод	Имя	Тип данных FORTRAN	Смысл	Допустимый тип/значение
Ввод	DIMLST	Integer (целое)	Длина выходного перечня ENTLST	1 ≤ DIMLST ≤ max

CALL RETRIEVE_MEMBER_GRP (GRPNAM, DIMLST, N, ENTLST)

Результат использования функции

Функция извлекает имена сущностей (геометрических или структурированных), принадлежащих существующей группе *api_group* с именем GRPNAM. При возникновении ошибки функция выдает значение *N*, равное 0.

Примечание — Если имя группы GRPNAM является именем текущей открытой группы *api_group*, то данная группа временно закрывается и открывается повторно после выполнения запроса.

Внутренние ссылки: 6.1.19.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
1002	Несоответствие номера сущности и длины перечня		

А.8.2.3 Запрос перечня сущностей контура

Имя функции:	Уровень интерфейса:	1
Retrieve_Ent_Ctr	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	CTRNAM	N	Имя контура *api_contour*	ctr
Вывод	N	I	Длина перечня ENTLST	≥ 0
Вывод	ENTLST	n × N	Перечень имен сущностей, составляющих *api_contour*	basic, conic_arc, pln

Привязка языка FORTRAN

Ввод/вывод	Имя	Тип данных FORTRAN	Смысл	Допустимый тип/значение
Ввод	DIMLST	Integer (целое)	Длина выходного перечня ENTLST	1 ≤ DIMLST ≤ max

CALL RETRIEVE_ENT_CTR (CTRNAM, DIMLST, N, ENTLST)

Результат использования функции

Функция извлекает перечень CTRNAM имен сущностей, составляющих контур *api_contour*. При возникновении ошибки функция возвращает значение *N* = 0.

Примечание — Если одна сущность или более из составляющих контур *api_contour*, является экземпляром типа группы *api_group*, то имена сущностей, содержащих данные группы, возвращаются в перечень сущностей ENTLST, а не в перечень имен членов группы.

Внутренние ссылки: 6.1.14.2, 6.1.19.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
1002	Несоответствие задаваемого номера и длины перечня		

А.8.3 Вычислительные утилиты

Вычисление расстояния между двумя точками	Distance_2_Pnt
Вычисление начального угла дуги окружности	Start_Angle_Arc
Вычисление конечного угла дуги окружности	End_Angle_Arc

А.8.3.1 Вычисление расстояния между двумя точками

Имя функции:	Уровень интерфейса:	1
Distance_2_Pnt	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PNTNM1	N	Имя первой декартовой точки *cartesian_point*	pnt
Ввод	PNTNM1	N	Имя второй декартовой точки *cartesian_point*	pnt
Вывод	DIST	D	Декартово расстояние между двумя точками	(EPS ≤ DIST ≤ MAX)

Привязка языка FORTRAN:
DIST = DISTANCE_2_PNT (PNTNM1, PNTNM2)

Результат использования функции

Функция вычисляет и возвращает значение декартова расстояния между двумя заданными декартовыми точками *cartesian_point* с именами PNTNM1 и PNTNM2. Расстояние измеряется в единицах длины *OVC_length_unit*. При возникновении ошибки значение расстояния DIST не определяется.

Примечание — Если данная функция используется в совокупности с функцией PNT_PROJECTION_ENT, то она может вычислить расстояние между точкой и любой другой базовой сущностью (например, декартова точка *cartesian_point*, прямая линия *api_line*, дуга окружности *api_circular_arc*). Если данная функция используется в совокупности с функцией PNT_PROJECTION_A2P, то она может вычислить расстояние между точкой и плоскостью (Oxy) локальной координатной системы *axis2_placement*.

Внутренние ссылки: 6.1.9.2, A.5.2.2.7, A.5.2.2.8.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

A.8.3.2 Вычисление начального угла дуги окружности

Имя функции:	Уровень интерфейса:	1
Start_Angle_Arc	Уровень геометрической мощности:	1, 2, 3

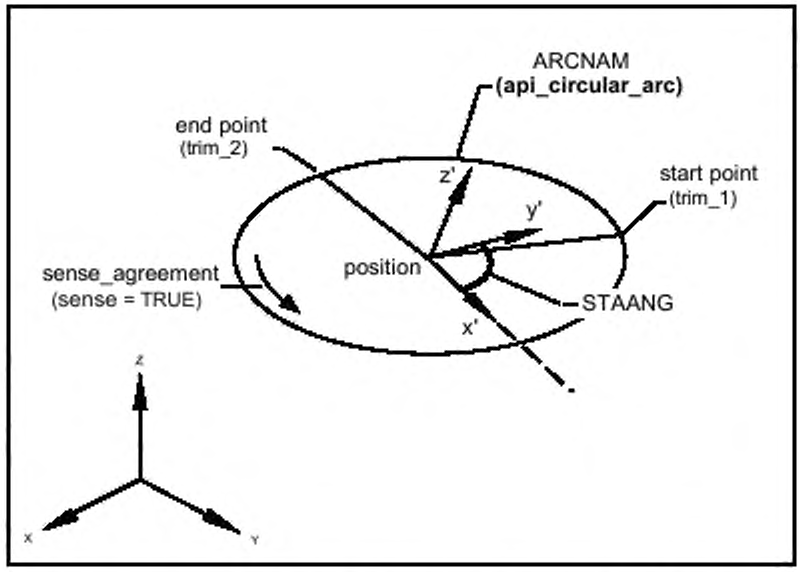
Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ARCNAM	N	Имя дуги окружности *api_circular_arc*	arc
Вывод	STAANG	D	Начальный угол дуги окружности	(0° ≤ STAANG ≤ 360°)

Привязка языка FORTRAN:
STAANG = START_ANGLE_ARC (ARCNAM)

Результат использования функции

Функция вычисляет и возвращает значение ориентированного начального угла заданной дуги окружности *api_circular_arc* с именем ARCNAM с учетом текущего положительного направления обхода дуги *sense_agreement_flag*. Вычисленный угол измеряется в единицах угла *OVC_angle_unit* в координатной плоскости (Oxy), в которой размещена заданная плоская дуга окружности *api_circular_arc* с именем ARCNAM. Угол отсчитывается от оси (Ox) локальной координатной системы и заканчивается в начальной точке указанной дуги *api_circular_arc*. При возникновении ошибки значение угла STAANG не определяется.



ARCNAM (api_circular_arc) — дуга окружности ARCNAM, end point (trim_2) — конечная точка дуги (точка вычленения 2), start point (trim_1) — начальная точка дуги (точка вычленения 1), sense_agreement (sense = TRUE) — положительное направление обхода кривой; position — центр окружности; STAANG — вычисляемый начальный угол дуги

Рисунок А.56 — Функция Start_Angle_Arc

Примечание — Нет.

Внутренние ссылки: 6.1.12.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

А.8.3.3 Вычисление конечного угла дуги окружности

Имя функции:	Уровень интерфейса:	1
End_Angle_Arc	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ARCNAM	N	Имя дуги окружности *api_circular_arc*	arc
Вывод	ENDANG	D	Конечный угол дуги окружности	(0° ≤ ENDANG ≤ 360°)

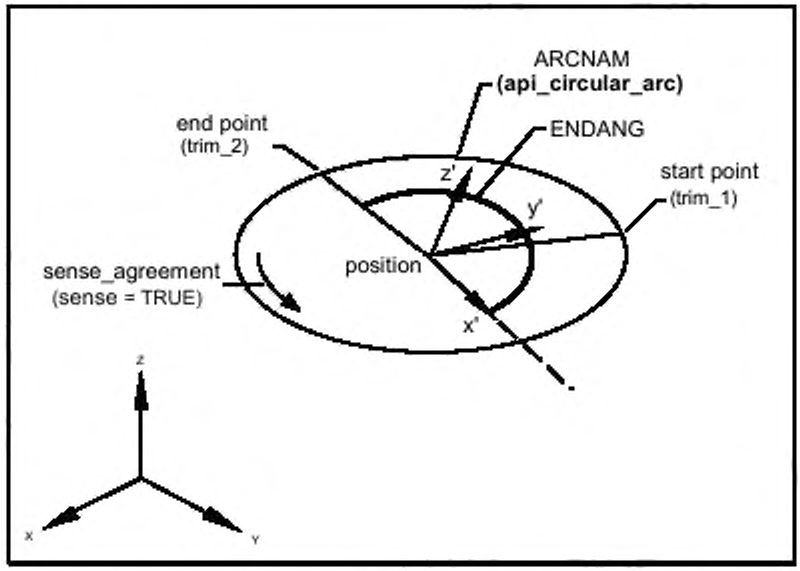
Привязка языка FORTRAN:
ENDANG = END_ANGLE_ARC (ARCNAM)

Результат использования функции

Функция вычисляет и возвращает значение ориентированного конечного угла заданной дуги окружности *api_circular_arc* с именем ARCNAM с учетом положительного направления дуги *sense_agreement_flag*. Вычисляемый угол измеряется в единицах угла *OVC_angle_unit* в плоскости (Oxy) заданной дуги окружности *api_circular_arc* с

именем ARCNAM. Угол отсчитывается от оси (Ox) до конечной точки заданной дуги окружности *api_circular_arc*. При возникновении ошибки полученное значение угла ENDANG не определяется.

Примечания — Нет.



ARCNAM (api_circular_arc) — дуга окружности ARCNAM; end point (trim_2) — конечная точка дуги (точка вычленения 2); ENDANG — конечный угол дуги окружности, start point (trim_1) — начальная точка дуги (точка вычленения 1), position — центр окружности; sense_agreement (sense = TRUE) - - положительное направление обхода кривой

Рисунок А.57 — Функция End_Angle_Arc

Внутренние ссылки: 6.1.12.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

А.9 Вспомогательные функции для выполнения преобразований

А.9.1 Построение и задание новых ссылочных координатных систем

Построение ссылочной координатной системы по трем точкам	Ref_Sys_3_Pnt
Построение ссылочной координатной системы по двум направлениям	Ref_Sys_2_Dir
Построение ссылочной координатной системы по двум направлениям (Ox) и (Oy)	Ref_Sys_2_Dir_Xy
Построение ссылочной координатной системы с помощью относительного позиционирования	Ref_Sys_Position_ Relative
Построение ссылочной координатной системы с помощью сущности *axis2_placement*	Ref_Sys_A2p

А.9.1.1 Построение ссылочной координатной системы по трем точкам

Имя функции:	Уровень интерфейса:	1
Ref_Sys_3_Pnt	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	CENPNT	N	Имя декартовой точки *cartesian_point*, определяющей начало координат	pnt
Ввод	AXSPNT	N	Имя декартовой точки *cartesian_point*, отложенной в направлении оси Z (игнорируется для 2D-вида)	pnt
Ввод	REFPNT	N	Имя декартовой точки *cartesian_point*, отложенной либо в направлени аппроксимаци оси *X*, либо в направлении точной оси *X* в случае 2D-вида	pnt
Ввод	KFIX	E	Хранение построенной сущности	[TDB, CAD]

Привязка языка FORTRAN:
CALL REF_SYS_3_PNT (CENPNT, AXSPNT, REFPNT)

Результат использования функции

Производится построение новой ссылочной OVC по заданным параметрам, относящимся к текущей ссылочной системе. Новая ссылочная система создается как ортогональная правосторонняя координатная система типа *axis2_placement*. Созданная сущность *axis2_placement* зависит от инициализации открытого вида: экземпляр сущности *axis2_placement_2d* создается в случае 2D-вида, экземпляр сущности *axis2_placement_3d* создается в случае 3D-вида. Для создания сущности *axis2_placement_3d* задаются три точки CENPNT, AXPNT и REFPNT, определяющие начало координат (O), и две оси (Oz и Ox) локальной координатной системы. Для создания сущности *axis2_placement_2d* необходимы только две точки вместо трех (CENPNT и REFPNT), которые определяют начало координат (O) и ось (Ox) локальной координатной системы.

Заданная декартова точка *cartesian_point* с именем CENPNT дублируется как точка *p1*. Данная точка *cartesian_point* определяет начало координат, она имеет нулевой стиль *null_style*.

В случае создания экземпляра локальной координатной системы *axis2_placement_3d*:

- пусть точки *P2* и *P3* являются синонимами двух заданных декартовых точек AXSPNT и REFPNT соответственно;

- создается экземпляр *d1* направления *direction* с компонентами *direction_ratio*, определенными путем вычитания координат точек *P2 — p1*. Указанное направление используется для определения точного направления локальной оси *Z* и имеет нулевой стиль. При этом расстояние между двумя декартовыми точками находится в диапазоне [EPS, MAX];

- создается экземпляр *d2* направления *direction* с компонентами *direction_ratio*, определенными путем вычитания координат точек *P3 — p1*. Указанное направление используется для определения аппроксимации направления оси *X* и имеет нулевой стиль. При этом расстояние между двумя декартовыми точками находится в диапазоне [EPS, MAX];

- создается экземпляр локальной координатной системы *axis2_placement_3d* с началом координат в точке *p1*, осью *d1* и ссылочным направлением *d2*. Полученная локальная координатная система *axis2_placement_3d* имеет нулевой стиль. Она принимается в качестве новой ссылочной OVC.

В случае создания экземпляра локальной координатной системы *axis2_placement_2d*:

- пусть точка *P3* является синонимом одной заданной декартовой точки *cartesian_point* с именем REFPNT;

- создается экземпляр *d2* направления *direction* с компонентами *direction_ratio*, определенного путем вычитания координат точек *P3 — p1*. Указанное направление используется для определения точного направления локальной оси *X* и имеет нулевой стиль. Расстояние между двумя декартовыми точками находится в диапазоне [EPS, MAX];

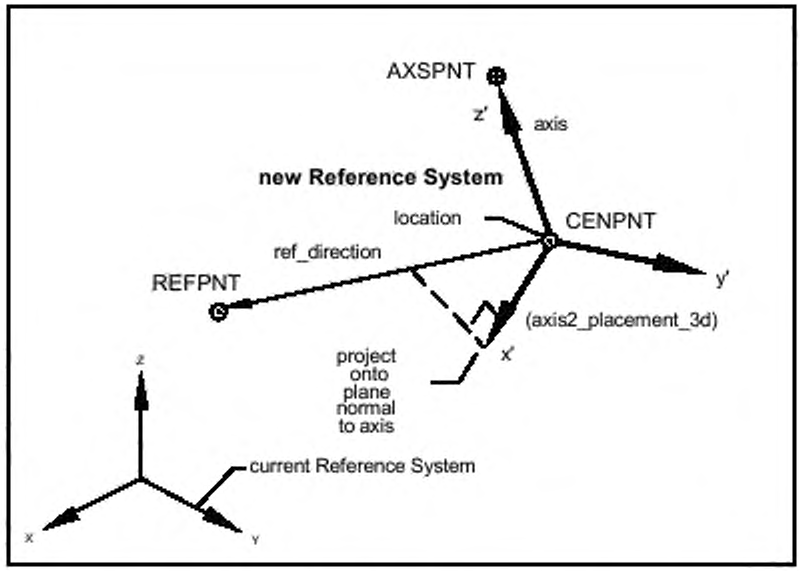
- создается экземпляр локальной координатной системы *axis2_placement_2d* с началом координат в точке *p1* и ссылочным направлением *d2*. Полученная локальная координатная система *axis2_placement_2d* имеет нулевой стиль и принимается в качестве новой ссылочной OVC.

При возникновении ошибки никакие изменения сущностей не производятся.

Примечания

1 При необходимости выполняется корректировка ссылочного направления *ref_direction* для обеспечения его ортогональности направлению оси путем проектирования указанного направления *ref_direction* на плоскость, перпендикулярную направлению оси *axis*.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то заданный параметр AXSPNT игнорируется интерфейсом.



AXSPNT — точка на новой оси Z'; axis — ось; new Reference System — новая ссылочная система; location — начало координат новой системы, CENPNT — точка CENPNT: ref_direction — ссылочное направление: REFPNT — ссылочная точка REFPNT; axis2_placement_3d — локальная координатная система, project onto plane normal to axis — проекция на плоскость, перпендикулярную оси: current Reference System — текущая ссылочная система

Рисунок А.58 — Функция Ref_Sys_3_Pnt (3D-вид)

Внутренние ссылки: 5.3.1, 6.1.9.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
103	Расстояние между двумя точками находится вне установленного диапазона [EPS, MAX]	105	Попытка создания вырожденного направления в процессе создания сущности
106	Попытка создания вырожденной локальной координатной системы *axis2_placement* в процессе создания сущности	116	Заданные точки являются линейно зависимыми
201	Переполнение временной базы данных	204	Функция несовместима с текущим уровнем мощности

А.9.1.2 Построение ссылочной координатной системы по двум направлениям

Имя функции:	Уровень интерфейса:	1
Ref_Sys_2_Dir	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	CENPNT	N	Имя декартовой точки *cartesian_point*, определяющей начало координат	pnt
Ввод	AXSDIR	N	Имя направления оси *Z* (игнорируется для 2D-вида)	dir
Вывод	REFDIR	N	Имя либо аппроксимации направления оси *X*, либо точного направления оси *X* в случае 2D-вида	dir

Привязка языка FORTRAN:
CALL REF_SYS_2_DIR (CENPNT, AXSDIR, REFDIR)

Результат использования функции

Функция строит и задает новую ссылочную OVC путем задания параметров, относящихся к текущей ссылочной системе. Новая ссылочная система является ортогональной правосторонней координатной системой типа *axis2_placement*. Созданная сущность *axis2_placement* зависит от инициализации открытого вида. В случае 2D-вида создается экземпляр сущности *axis2_placement_2d*, в случае 3D-вида — экземпляр сущности *axis2_placement_3d*. Для создания локальной координатной системы типа *axis2_placement_3d* задаются три параметра (CENPNT, AXDIR и REFDIR) для построения начала координат (O) и двух координатных осей (Oz и Ox) локальной координатной системы. Для создания локальной координатной системы *axis2_placement_2d* задаются только два параметра (CENPNT и REFDIR) для построения начала координат (O) и координатной оси (Ox).

Заданная декартова точка *cartesian_point* CENPNT дублируется как точка *p1*. Полученная декартова точка определяет начало координат и имеет нулевой стиль *null_style*.

В случае создания экземпляра сущности *axis2_placement_3d*:

- два направления AXSDIR и REFDIR дублируются как направления *d1* и *d2* соответственно. Настоящие направления определяют точное направление локальной оси *Z* и аппроксимацию направления локальной оси *X*. Указанные направления имеют нулевой стиль;
- создается экземпляр сущности *axis2_placement_3d* с началом координат в точке *p1*, осью *d1* и ссылочным направлением *ref_direction d2*. Функция возвращает имя полученной сущности *axis2_placement_3d*. Сущности назначен нулевой стиль, она является новой ссылочной OVC.

В случае создания экземпляра сущности *axis2_placement_2d*:

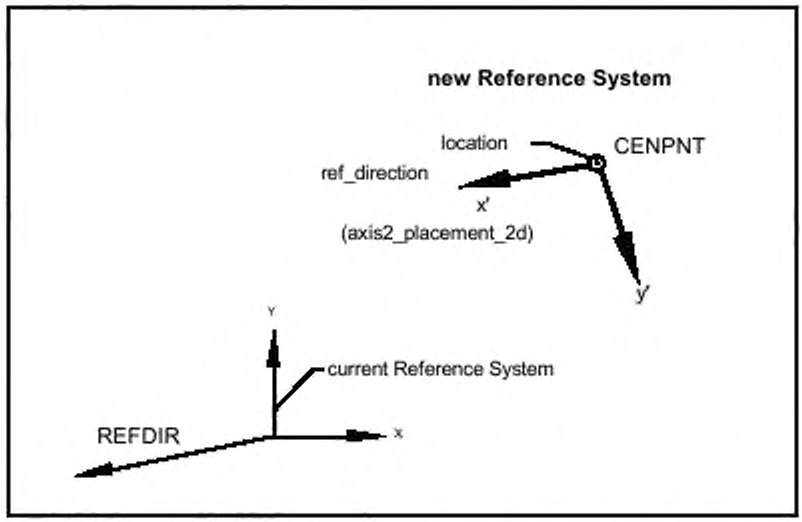
- ссылочное направление REFDIR дублируется как направление *d2*, ему назначен нулевой стиль. Данное направление используется для определения точного направления локальной оси *X*;
- создается экземпляр сущности *axis2_placement_2d* с началом координат *p1* и ссылочным направлением *ref_direction d2*. Функция возвращает имя построенной сущности *axis2_placement_2d*. Сущности назначен нулевой стиль, она является новой ссылочной OVC.

При возникновении ошибки никакие изменения не производятся.

Примечания

1 При необходимости производится корректировка ссылочного направления *ref_direction* для обеспечения ортогональности направлению оси. Корректировка производится путем проецирования ссылочного направления *ref_direction* на плоскость, перпендикулярную направлению оси *axis*.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то заданный параметр AXSDIR игнорируется интерфейсом.



New Reference System — новая ссылочная система; location — начало координат новой системы; CENPNT — точка CENPNT; ref_direction — ссылочное направление; axis2_placement_2d — локальная координатная система (сущность axis2_placement_2d); current Reference System — текущая ссылочная система; REFDIR — ссылочное направление

Рисунок А.59 — Функция Ref_Sys_2_Dir (2D-вид)

Внутренние ссылки: 5.3.1, 6.1.9.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
106	Попытка создания вырожденной локальной координатной системы *axis2_placement* в процессе создания сущности	117	Заданные направления параллельны
201	Переполнение временной базы данных	204	Функция несовместима с текущим уровнем мощности

A.9.1.3 Построение ссылочной координатной системы по двум направлениям (Ox) и (Oy)

Имя функции:	Уровень интерфейса:	1
Ref_Sys_2_Dir_Xy	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	CENPNT	N	Имя декартовой точки *cartesian_point*, определяющей начало координат	pnt
Ввод	REFDIR	N	Имя точного направления оси *X*	dir
Ввод	YAXDIR	N	Имя аппроксимации направления оси *Y* (игнорируется для 2D-вида)	dir

Привязка языка FORTRAN:
CALL REF_SYS_2_DIR_XY (CENPNT, REFDIR, YAXDIR)

Результат использования функции

Строится новая ссылочная OVC путем задания параметров, относящихся к текущей ссылочной системе. Новая ссылочная система является ортогональной правосторонней координатной системой типа *axis2_placement*. Создание сущности *axis2_placement* зависит от инициализации открытого вида. В случае 2D-вида создается экземпляр сущности *axis2_placement_2d*, в случае 3D-вида создается экземпляр сущности *axis2_placement_3d*. При создании локальной координатной системы *axis2_placement_3d* задаются три параметра (CENPNT, REFDIR и YAXDIR) для построения начала координат (O) и двух координатных осей (Ox и Oy). При создании локальной координатной системы *axis2_placement_2d* задаются только два из этих трех параметров (CENPNT и REFDIR) для построения начала координат (O) и координатной оси (Ox).

Декартова точка *cartesian_point* CENPNT дублируется как точка *p1* для задания начала координат. Направление REFDIR дублируется как направление *d1* для задания точного направления локальной оси *X*. Указанным двум сущностям назначен нулевой стиль *null_style*.

В случае создания экземпляра сущности *axis2_placement_3d*:

- направление *d2* создается путем вычисления проекции нормированного направления YAXDIR на плоскость, перпендикулярную направлению *d1*. Указанному направлению назначен нулевой стиль;
- создается экземпляр направления *d3* с атрибутами, вычисленными по векторному произведению направлений *d1* и *d2*. Данное направление задает точное направление локальной оси *Z*, имеющей нулевой стиль;
- создается экземпляр сущности *axis2_placement_3d* с началом координат в точке *p1*, осью *d3* и ссылочным направлением *d1*. Функция возвращает имя полученной сущности *axis2_placement_3d*. Сущности назначен нулевой стиль, он задается как новая ссылочная OVC.

В случае создания экземпляра сущности *axis2_placement_2d*:

- создается экземпляр сущности *axis2_placement_2d* с началом координат в точке *p1* и ссылочным направлением *d1*. Функция возвращает имя полученной сущности *axis2_placement_2d*. Сущности назначен нулевой стиль, она задается как новая ссылочная OVC.

При возникновении ошибки никакие изменения не производятся.

Примечание — Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то параметр YAXDIR игнорируется интерфейсом.

Внутренние ссылки: 6.1.9, 6.1.9.7, 6.1.9.8, 6.2.1.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
106	Попытка создания вырожденной локальной координатной системы *axis2_placement* в процессе создания сущности	117	Заданные направления параллельны
201	Переполнение временной базы данных	204	Функция несовместима с текущим уровнем мощности

A.9.1.4 Построение ссылочной координатной системы с помощью относительного позиционирования

Имя функции:	Уровень интерфейса:	1
Ref_Sys_Position_Relative	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	REFLST	6 × D	Перечень описаний шести последовательных вращений и относительных перемещений:	
			1) поворот в плоскости (Oxy) вокруг оси *Z* OVC	(–360° ≤ REFLST(1) ≤ 360°)
			2) поворот в плоскости (Ozy) вокруг оси *X* OVC	(–360° ≤ REFLST(2) ≤ 360°)
			3) поворот в плоскости (Ozx) вокруг оси *Y* OVC	(–360° ≤ REFLST(3) ≤ 360°)
			4) смещение в направлении оси (Ox) текущей ссылочной OVC	(0.0 или (EPS ≤ |REFLST(4)| ≤ MAX))
			5) смещение в направлении оси (Oy) текущей ссылочной OVC	(0.0 или (EPS ≤ |REFLST(5)| ≤ MAX))
			6) смещение в направлении оси (Oz) текущей ссылочной OVC	(0.0 или (EPS ≤ |REFLST(6)| ≤ MAX))

Привязка языка FORTRAN:
CALL REF_SYS_POSITION_RELATIVE (REFLST)

Результат использования функции

Строится новая ссылочная OVC путем задания параметра REFLST, значения которого относятся к текущей ссылочной системе. Новая ссылочная система является ортогональной правосторонней координатной системой типа *axis2_placement*. Тип созданной локальной координатной системы *axis2_placement* зависит от инициализации открытого вида. В случае 2D-вида создается экземпляр локальной координатной системы *axis2_placement_2d*. В случае 3D-вида создается экземпляр локальной координатной системы *axis2_placement_3d*.

В случае создания экземпляра сущности *axis2_placement_3d*:

- создается экземпляр локальной координатной системы *axis2_placement_3d* как копии базовой OVC. Для всех неявно созданных экземпляров сущности *axis2_placement_3d* и самой сущности назначен нулевой стиль *null_style*;

- матрицы преобразования, содержащиеся внутри заданного параметра REFLIST, применяются к новой системе *axis2_placement_3d* в следующей последовательности:

1) вращение вокруг оси *Z* текущей ссылочной OVC;

2) вращение вокруг оси *X* текущей ссылочной OVC;

3) вращение вокруг оси *Y* текущей ссылочной OVC;

4) смещение начала координат новой сущности *axis2_placement_3d* по осям *X*, *Y* и *Z* текущей ссылочной OVC;

- задается локальная координатная система *axis2_placement_3d* как новая ссылочная OVC.

В случае создания экземпляра сущностьа *axis2_placement_2d*:

- создается экземпляр сущности *axis2_placement_2d* как копии текущей ссылочной OVC;

- матрицы преобразования, содержащиеся внутри заданного параметра REFLIST, применяются к новой сущности *axis2_placement_2d* в следующей последовательности:

1) вращение в плоскости (Oxy) текущей ссылочной OVC;

2) смещение начала координат новой сущности *axis2_placement_2d* по осям *X* и *Y* текущей ссылочной OVC;

- задание сущности *axis2_placement_2d* как новой ссылочной OVC.

Углы поворота измеряются в единицах угла *OVC_angle_unit*. Смещения измеряются в единицах длины *OVC_length_unit*. Указанные величины либо равны 0, либо лежат в диапазоне [EPS, MAX]. При возникновении ошибки никакие изменения не производятся.

Примечания

1 Если вычисленная евклидова норма смещения находится в диапазоне [*ZERO_value*, EPS], то никакие смещения не производятся и никакие ошибки не возникают.

2 Если текущий открытый вид определен как 2D-вид (значение записи *geometrical_power_level* в таблице статуса интерфейса равно 1), то значения параметра REFLST для углов поворота и значения параметра REFLST для смещения по оси *Z* игнорируются интерфейсом.

Внутренние ссылки: 5.3.1, 6.1.9.

Ошибки

3	Значение меры длины находится вне допустимого диапазона	4	Значение меры плоского угла находится вне допустимого диапазона
106	Попытка создания вырожденной локальной координатной системы *axis2_placement* в процессе создания сущности	201	Переполнение временной базы данных
204	Функция несовместима с текущим уровнем мощности		

A.9.1.5 Построение ссылочной координатной системы с помощью сущности axis2_placement

Имя функции:	Уровень интерфейса:	1
Ref_Sys_A2p	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	A2PNAM	N	Имя декартовой точки *cartesian_point*, определяющей начало координат	a2p

Привязка языка FORTRAN:
CALL REF_SYS_A2P (A2PNAM)

Результат использования функции

Строится и задается новая ссылочная OVC с положением начала координат и направлением, идентичными положению и направлению заданной сущности *axis2_placement* с именем A2PNAM. Новая сущность является ортогональной правосторонней координатной системой типа *axis2_placement*.

Тип созданной локальной координатной системы *axis2_placement* зависит от инициализаци открытого вида. В случае 2D-вида создается экземпляр сущности *axis2_placement_2d*, в случае 3D-вида — экземпляр сущности *axis2_placement_3d*.

В случае создания экземпляра сущности *axis2_placement_2d*:

- создается экземпляр сущности *axis2_placement_2d* с началом координат, осью и ссылочным направлением *ref_direction* по заданной сущности *axis2_placement* с именем A2PNAM. Полученная сущность *axis2_placement_2d* задается как новая ссылочная OVC.

В случае создания экземпляра сущности axis2_placement_3d:

- создается экземпляр локальной координатной системы *axis2_placement_3d* с началом координат, осью и ссылочным направлением *ref_direction*, определяемым по заданной локальной координатной системе *axis2_placement* с именем A2PNAM. Сущность *axis2_placement_3d* задает новую ссылочную OVC.

При возникновении ошибки никакие изменения не производятся.

Примечание — Нет.

Внутренние ссылки: 5.3.1, 6.1.9.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
201	Переполнение временной базы данных	204	Функция несовместима с текущим уровнем мощности

А.10 Функции визуального представления

Данные функции позволяют прикладным программам задавать и запрашивать свойства глобальной визуализации элементов геометрического представления, а также изменять атрибуты визуализации для сущностей, созданных во временной базе данных (TDB).

А.10.1 Задание глобальных записей для атрибутов визуализации

Задание записи стиля точки	Set_Point_Style
Задание записи стиля кривой	Set_Curve_Style
Задание записи стиля заполненной области	Set_Fill_Area_Style
Задание записи стиля поверхности	Set_Surface_Style
Задание записи толщины линий штриховки	Set_Hatching_Width
Задание записи типа линий штриховки	Set_Hatching_Curve_Font
Задание записи цвета штриховки	Set_Hatching_Colour
Задание аспекта невидимых линий	Set_Hidden_Line_Aspect
Задание записи относительного уровня вида	Set_Relative_View_Level

А.10.1.1 Задание записи стиля точки

Имя функции:	Уровень интерфейса:	1
Set_Point_Style	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	EXTSOU	S	Имя источника, содержащего определение стиля точки	(ISO_13584 31, ISO_13584-1 +)
Ввод	PNTSTY	S	Идентификатор стиля точки	

Привязка языка FORTRAN:
CALL SET_POINT_STYLE (EXTSOU, PNTSTY)

Результат использования функции

Функция задает новое текущее значение записи *point_style* в таблице статуса интерфейса, определенное именем EXTSOU и идентификатором PNTSTY. Источником внешне определенного стиля должен быть настоящий стандарт или любой протокол обмена видами, соответствующий ИСО 13584. При возникновении ошибки никакие изменения не производятся.

Примечания

1 Стиль воспроизведения сущности точки определен экземпляром сущности *api_externally_defined_point_style* с источником *source*, равным EXTSOU и идентификатором *item_id*, равным PNTSTY.

2 Если требуемый внешне определенный стиль *externally_defined_style* для интерфейса не задан, то используется стиль точки *asterisk_point*. При этом ошибка не возникает.

Внутренние ссылки: 6.2.4.1, 8.2.

Ошибки

209	Превышение максимально допустимого количества символов в строке	401	Источник протокола обмена неизвестен
402	Идентификатор внешнего стиля неизвестен		

А.10.1.2 Задание записи стиля кривой

Имя функции:	Уровень интерфейса:	1
Set_Curve_Style	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	EXTSOU	S	Имя источника, содержащего определение стиля кривой	(ISO_13584 31, ISO_13584-1 +)
Ввод	CURSTY	S	Идентификатор стиля кривой	

Привязка языка FORTRAN:
CALL SET_CURVE_STYLE (EXTSOU, CURSTY)

Результат использования функции

Функция задает новое текущее значение стиля кривой *curve_style* в таблице статуса интерфейса с именем EXTSOU и идентификатором CURSTY. Источником внешне определенного стиля является настоящий стандарт или любой протокол обмена видами, соответствующий ИСО 13584. При возникновении ошибки никакие изменения не производятся.

Примечания

1 Стиль воспроизведения сущности кривой определен экземпляром сущности *api_externally_defined_curve_style* с источником *source*, равным EXTSOU, и идентификатором *item_id*, равным CURSTY.

2 Стиль кривой также используется сущностью назначения стиля презентации *presentation_style_assignment* для твердотельной модели *solid_model*.

3 Если требуемый внешне определенный стиль кривой *externally_defined_style* для интерфейса не задан, то используется стиль кривой *plain_solid_line*. При этом ошибки не возникают.

Внутренние ссылки: 6.2.4.2, 8.2.

Ошибки

209	Превышение максимально допустимого количества символов в строке	401	Неизвестный источник протокола обмена
402	Неизвестный идентификатор внешнего стиля		

А.10.1.3 Задание записи стиля заполненной области

Имя функции:	Уровень интерфейса:	1
Set_Fill_Area_Style	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	EXTSOU	S	Имя источника, содержащего определение стиля заполненной области	(ISO_13584_31, ISO_13584-1 +)
Ввод	AFASTY	S	Идентификатор стиля заполненной области	

Привязка языка FORTRAN:
CALL SET_FILL_AREA_STYLE (EXTSOU, AFASTY)

Результат использования функции

Функция задает новые текущие значения записи *fill_area_style* в таблице статуса интерфейса, определенные именем EXTSOU и идентификатором AFASTY. Источником внешне определенного стиля является настоящий стандарт или любой протокол обмена видами, соответствующий ИСО 13584. При возникновении ошибки никакие изменения не производятся.

Примечания

1 Стиль воспроизведения сущности заполненной области определен экземпляром сущности *api_externally_defined_fill_area_style* с источником *source*, равным EXTSOU и идентификатором *item_id*, равным AFASTY.

2 Если требуемый внешне определенный стиль *externally_defined_style* для настоящего интерфейса не задан, то используется стиль заполненной области *opaque_fill_area*. При этом ошибка не возникает.

Внутренние ссылки: 6.2.4.3, 8.2.

Ошибки

209	Превышение максимально допустимого количества символов в строке	401	Неизвестный источник протокола обмена
402	Неизвестный идентификатор внешнего стиля		

A.10.1.4 Задание записи стиля поверхности

Имя функции:	Уровень интерфейса:	1
Set_Surface_Style	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	EXTSOU	S	Имя источника, содержащего определение стиля поверхности	(ISO_13584_31, ISO_13584-1 +)
Ввод	SURSTY	S	Идентификатор стиля поверхности	

Привязка языка FORTRAN:
CALL SET_SURFACE_STYLE (EXTSOU, SURSTY)

Результат использования функции

Функция задает новые текущие значения записи *surface_style* в таблице статуса интерфейса, определенные именем EXTSOU и идентификатором SURSTY. Источником внешне определенного стиля является настоящий стандарт или любой протокол обмена видами, соответствующий ИСО 13584. При возникновении ошибки никакие изменения не производятся.

Примечания

1 Стиль воспроизведения сущности поверхности определен экземпляром сущности *api_externally_defined_surface_style* с источником *source*, равным EXTSOU, и идентификатором *item_id*, равным SURSTY.

2 Стиль поверхности также используется для назначения стиля презентации *presentation_style_assignment* твердотельных моделей *solid_model*.

3 Если требуемый внешне определенный стиль *externally_defined_style* в настоящем интерфейсе не задан, то используется стиль поверхности *solid_surface*. При этом ошибка не возникает.

Внутренние ссылки: 6.2.4.4, 8.2.

Ошибки

209	Превышение максимально допустимого количества символов в строке	401	Неизвестный источник протокола обмена
402	Неизвестный идентификатор внешнего стиля		

А.10.1.5 Задание записи толщины линий штриховки

Имя функции:	Уровень интерфейса:	1
Set_Hatching_Width	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	WIDTH	S	Метка предварительно определенной толщины линий штриховки, описанной в настоящем стандарте	(thin_hatching_line, middle_thick_ hatching_line, thick_hatching_line)

Привязка языка FORTRAN:
CALL SET_HATCHING_WIDTH (WIDTH, ERR)

Результат использования функции

Функция задает новые текущие значения записи *hatch_width* в таблице статуса интерфейса, определенные заданной меткой WIDTH. При возникновении ошибки никакие изменения не производятся.

Примечание — Стиль толщины линий штриховки заполненной области *fill_area_style_hatching* определен как экземпляр сущности *api_pre_defined_hatching_width* с именем WIDTH.

Внутренние ссылки: 6.2.5.1, 8.2.

Ошибки

6	Значение строки находится вне допустимого диапазона		

А. 10.1.6 Задание записи типа линий штриховки

Имя функции:	Уровень интерфейса:	1
Set_Hatching_Curve_Font	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	FONT	S	Метка предварительно определенного типа линий штриховки, описанная в настоящем стандарте	(continuous, dashed, chain, chain_double_dash, dotted)

Привязка языка FORTRAN:
CALL SET_HATCHING_CURVE_FONT (FONT)

Результат использования функции

Функция задает новое текущее значение записи *hatch_curve_font* в таблице статуса интерфейса, определенное заданной меткой FONT. При возникновении ошибки никакие изменения не производятся.

Примечание — Стиль типа линий штриховки заполненной области *fill_area_style_hatching* определен как экземпляр сущности *api_pre_defined_hatching_curve_font* с именем FONT.

Внутренние ссылки: 6.2.5.2, 8.2.

Ошибки

6	Значение строки находится вне допустимого диапазона		

А.10.1.7 Задание записи цвета штриховки

Имя функции:	Уровень интерфейса:	1
Set_Hatching_Colour	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	COLOUR	S	Метка предварительно определенного цвета штриховки, описанная в настоящем стандарте	hatch_line_colour

Привязка языка FORTRAN:
CALL SET_HATCHING_COLOUR (COLOUR)

Результат использования функции

Функция задает новое текущее значение записи *hatch_colour* в таблице статуса интерфейса, определенное заданной меткой COLOUR. При возникновении ошибки никакие изменения не производятся.

Примечание — Стиль толщины линий штриховки заполненной области *fill_area_style_hatching* определен как экземпляр сущности *api_pre_defined_hatching_colour* с именем COLOUR.

Внутренние ссылки: 6.2.5.3, 8.2.

Ошибки

6	Значение строки находится вне допустимого диапазона		

А.10.1.8 Задание аспекта невидимых линий

Имя функции:	Уровень интерфейса:	1
Set_Hidden_Line_Aspect	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	HIDSTY	S	Метка стиля невидимой линии, описанная в настоящем стандарте	(hidden_line_no_changed, hidden_line_dashed, hidden_line_invisible)

Привязка языка FORTRAN:
CALL SET_HIDDEN_LINE_ASPECT (HIDSTY)

Результат использования функции

Функция задает новое текущее значение записи *hidden_line_aspect* в таблице статуса интерфейса, определенное заданной меткой HIDSTY. При возникновении ошибки никакие изменения не производятся.

Примечание — Изображение невидимых линий определяется в таблице статуса интерфейса атрибутом сущности *api_pre_defined_occlusion_style* с именем HIDSTY и уровнем вида *view_level*, равным соответствующей текущей записи *view_level*.

Внутренние ссылки: 6.2.2.1, 6.2.5.4, 8.2.

Ошибки

6	Значение строки находится вне допустимого диапазона		

А.10.1.9 Задание записи относительного уровня вида

Имя функции:	Уровень интерфейса:	1
Set_Relative_View_Level	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	RVL	D	Виртуальная высота (уровень вида)	

Привязка языка FORTRAN:
CALL SET_RELATIVE_VIEW_LEVEL (RVL)

Результат использования функции

Функция задает новое текущее значение записи *view_level* в таблице статуса интерфейса, определенное заданным значением RVL. При возникновении ошибки никакие изменения не производятся.

Примечания

1 Значение |RVL| не может находиться в интервале между *ZERO_value* и EPS.

2 Относительный уровень вида определен атрибутом *view_level* сущности *api_pre_defined_occlusion_style*.

Внутренние ссылки: 6.2.2.1, 6.2.5.4, 8.2.

Ошибки

7	Действительное значение находится вне допустимого диапазона		

А.10.2 Запросы атрибутов визуализации из глобальных записей

Запрос записи стиля точки	Inq_Point_Style
Запрос записи стиля кривой	Inq_Curve_Style
Запрос записи стиля заполненной области	Inq_Fill_Area_Style
Запрос записи стиля поверхности	Inq_Surface_Style
Запрос записи толщины линий штриховки	Inq_Hatching_Width
Запрос записи типа линий штриховки	Inq_Hatching_Curve_Font
Запрос записи цвета штриховки	Inq_Hatching_Colour
Запрос записи аспекта невидимых линий	Inq_Hidden_Line_Aspect
Запрос записи относительного уровня вида	Inq_Relative_View_Level

А.10.2.1 Запрос записи стиля точки

Имя функции:	Уровень интерфейса:	1
Inq_Point_Style	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	EXTSOU	S	Имя источника, содержащего определение стиля точки	(ISO_13584_31, ISO_13584-1 +)
Вывод	PNTSTY	S	Идентификатор стиля точки	
Вывод	ERR	E	Индикатор ошибки *error_indicato*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_POINT_STYLE (EXTSOU, PNTSTY, ERR)

Результат использования функции

Функция доставляет текущее значение записи стиля точки *point_style* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* с именем ERR сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Стиль воспроизведения точки определен экземпляром сущности *api_externally_defined_point_style* с источником *source*, равным EXTSOU, и идентификатором *item_id*, равным PNTSTY.

Внутренние ссылки: 6.2.4.1, 8.2.

Ошибки

—	Ошибок нет		

А.10.2.2 Запрос записи стиля кривой

Имя функции:	Уровень интерфейса:	1
Inq_Curve_Style	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	EXTSOU	S	Имя источника, содержащего определение стиля кривой	(ISO_13584_31, ISO_13584-1 +)
Ввод	CURSTY	S	Идентификатор стиля кривой	
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_CURVE_STYLE (EXTSOU, CURSTY, ERR)

Результат использования функции

Функция доставляет текущее значение записи *curve_style* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* с именем ERR сообщает о проблемах, возникающих в процессе выполнения функции.

Примечания

1 Стиль воспроизведения кривой определен экземпляром сущности *api_externally_defined_curve_style* с источником, равным EXTSOU, и идентификатором *item_id*, равным CURSTY.

2 Стиль кривой используется также при назначении стиля представления *presentation_style_assignment* твердотельной модели *solid_model*.

Внутренние ссылки: 6.2.4.2, 8.2.

Ошибки

—	Ошибок нет		

А.10.2.3 Запрос записи стиля заполненной области

Имя функции:	Уровень интерфейса:	1
Inq_Fill_Area_Style	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	EXTSOU	S	Имя источника, содержащего определение стиля заполненной области	(ISO_13584_31, ISO_13584-1 +)
Вывод	AFASTY	S	Идентификатор стиля заполненной области	
Вывод	ERR	E	Идентификатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_FILL_AREA_Style (EXTSOU, AFASTY, ERR)

Результат использования функции

Функция доставляет текущее значение записи *fill_area_style* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* с именем ERR сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Стиль воспроизведения заполненной области определяется экземпляром сущности *api_externally_defined_fill_area_style* с источником *source*, равным EXTSOU, и идентификатором *item_id*, равным AFASTY.

Внутренние ссылки: 6.2.4.3, 8.2.

Ошибки

—	Ошибок нет		

А.10.2.4 Запрос записи стиля поверхности

Имя функции:	Уровень интерфейса:	1
Inq_Surface_Style	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	EXTSOU	S	Имя источника, содержащего определение стиля поверхности	(ISO_13584_31, ISO_13584-1 +)
Вывод	SURSTY	S	Идентификатор стиля поверхности	
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_SURFACE_STYLE (EXTSOU, SURSTY, ERR)

Результат использования функции

Функция доставляет текущее значение записи *surface_style* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* с именем ERR сообщает о проблемах, возникающих в процессе выполнения функции.

Примечания

1 Стиль воспроизведения поверхности определен экземпляром сущности *api_externally_defined_surface_style* с источником *source*, равным EXTSOU, и идентификатором *item_id*, равным SURSTY.

2 Стиль поверхности используется также при назначении стиля представления *presentation_style_assignment* твердотельной модели *solid_model*.

Внутренние ссылки: 6.2.4.4, 8.2.

Ошибки

—	Ошибок нет		

А.10.2.5 Запрос записи толщины линий штриховки

Имя функции:	Уровень интерфейса:	1
Inq_Hatching_Width	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	WIDTH	S	Метка предварительно определенной толщины линий штриховки, описанной в настоящем стандарте	(thin_hatching_line, middle_thick_ hatching_line, thick_hatching_line)
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_HATCHING_WIDTH (WIDTH, ERR)

Результат использования функции

Функция доставляет текущее значение записи *hatch_width* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* с именем ERR сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Стиль толщины линий штриховки заполненной области *fill_area_style_hatching* определен экземпляром сущности *api_pre_defined_hatching_width* с именем WIDTH.

Внутренние ссылки: 6.2.5.1, 8.2.

Ошибки

—	Ошибок нет		

А.10.2.6 Запрос записи типа линий штриховки

Имя функции:	Уровень интерфейса:	1
Inq_Hatching_Curve_Font	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	FONT	S	Метки предварительно определенных типов линий штриховки, описанных в настоящем стандарте	continuous, dashed, chain, chain double dash, dotted
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_HATCHING_CURVE_FONT (FONT, ERR)

Результат использования функции

Функция доставляет текущее значение записи *hatch_curve_font* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* с именем ERR сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Тип кривой штриховки заполненной области *fill_area_style_hatching* определен экземпляром сущности *api_pre_defined_hatching_curve_font* с именем FONT.

Внутренние ссылки: 6.2.5.2, 8.2.

Ошибки

—	Ошибок нет		

А.10.2.7 Запрос записи цвета штриховки

Имя функции:	Уровень интерфейса:	1
Inq_Hatching_Colour	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	COLOUR	S	Метка предварительно определенного цвета штриховки, описанного в настоящем стандарте	hatch_line_colour
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_HATCHING_COLOUR (COLOUR, ERR)

Результат использования функции

Функция доставляет текущее значение записи *hatch_colour* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* с именем ERR сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Толщина линий штриховки заполненной области *fill_area_style_hatching* определена экземпляром сущности *api_pre_defined_hatching_colour* с именем COLOUR.

Внутренние ссылки: 6.2.5.3, 8.2.

Ошибки

—	Ошибок нет		

А.10.2.8 Запрос записи аспекта невидимых линий

Имя функции:	Уровень интерфейса:	1
Inq_Hidden_Line_Aspect	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	HIDSTY	S	Метка стиля невидимых линий, описанного в настоящем стандарте	(hidden_line_no_changed, hidden_line_dashed, hidden_line_invisible)
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_HIDDEN_LINE_ASPECT (HIDSTY, ERR)

Результат использования функции

Функция доставляет текущее значение записи *hidden_line_aspect* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* с именем ERR сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Стиль изображения невидимой линии определен атрибутом сущности *api_pre_defined_occlusion_style* с именем HIDSTY и уровнем вида *view_level*, равным текущей записи *view_level* таблицы статуса интерфейса.

Внутренние ссылки: 6.2.2.1,6.2.5.4, 8.2.

Ошибки

—	Ошибок нет		

А.10.2.9 Запрос записи относительного уровня вида

Имя функции:	Уровень интерфейса:	1
Inq_Relative_View_Level	Уровень геометрической мощности:	0, 1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Вывод	RVL	D	Виртуальная высота (уровень вида)	
Вывод	ERR	E	Индикатор ошибки *error_indicator*	[NOERROR, ERROR]

Привязка языка FORTRAN:
CALL INQ_RELATIVE_VIEW_LEVEL (RVL, ERR)

Результат использования функции

Функция доставляет текущее значение записи *view_level* из таблицы статуса интерфейса. Индикатор ошибки *error_indicator* с именем ERR сообщает о проблемах, возникающих в процессе выполнения функции.

Примечание — Относительный уровень вида определен атрибутом *view_level* сущности *api_pre_defined_occlusion_style*.

Внутренние ссылки: 6.2.2.1, 6.2.5.4, 8.2.

Ошибки

—	Ошибок нет		

А.10.3 Изменение стиля воспроизведения сущностей

Изменение стиля воспроизведения точки	Chg_Point_Style
Изменение стиля воспроизведения кривой (тела)	Chg_Curve_Style
Изменение стиля воспроизведения заполненной области	Chg_Fill_Area_Style
Изменение стиля воспроизведения поверхности (тела)	Chg_Surface_Style
Изменение толщины линий штриховки сущности *fill_area_style_hatching*	Chg_Hatching_Width
Изменение типа линий штриховки сущности *fill_area_style_hatching*	Chg_Hatching_Curve_Font
Изменение цвета штриховки сущности *fill_area_style_hatching*	Chg_Hatching_Aolour
Изменение аспекта невидимых линий	Chg_Hidden_Line_Aspect
Изменение относительного уровня вида для сущности включения невидимых линий	Chg_Relative_View_Level

А.10.3.1 Изменение стиля воспроизведения точки

Имя функции:	Уровень интерфейса:	1
Chg_Point_Style	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PNTNAM	N	Имя декартовой точки *cartesian_point*	pnt, grp
Ввод	EXTSOU	S	Имя источника, содержащего определение стиля точки	(ISO_13584_31, ISO_13584-1 +)
Ввод	PNTSTY	S	Идентификатор стиля точки	

Привязка языка FORTRAN:
CALL CHG_POINT_STYLE (PNTNAM, EXTSOU, PNTSTY)

Результат использования функции

Функция изменяет стиль воспроизведения декартовой точки *cartesian_point* с именем PNTNAM путем его переназначения. Новый стиль воспроизведения сущности определен источником с именем EXTSOU и идентификатором PNTSTY. Источником внешне определенного стиля является настоящий стандарт или любой протокол обмена видами, соответствующий ИСО 13584. При возникновении ошибки стиль воспроизведения не изменяется.

Примечания

1 Стиль воспроизведения сущности точки определен экземпляром сущности *api_externally_defined_point_style* с источником *source*, равным EXTSOU, и идентификатором *item_id*, равным PNTSTY.

2 Если требуемый внешне определенный стиль *externally_defined_style* для интерфейса не задан, то используется стиль точки *asterisk_point*. При этом ошибка не возникает.

3 Если заданная сущность PNTNAM является экземпляром типа группы *api_group*, то все сущности *cartesian_point*, ссылающиеся на данную группу, модифицируются.

Внутренние ссылки: 6.1.9.2, 6.1.19.1, 6.2.4.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	209	Превышение максимально допустимого количества символов в строке
401	Неизвестный источник протокола обмена	402	Неизвестный идентификатор внешнего стиля

A.10.3.2 Изменение стиля воспроизведения кривой (тела)

Имя функции:	Уровень интерфейса:	1
Chg_Curve_Style	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности кривой или твердотельной модели *solid_model*	curves, solid_model, grp
Ввод	EXTSOU	S	Имя источника, содержащего определение стиля кривой	(ISO_13584_31, ISO_13584-1 +)
Ввод	CURSTY	S	Идентификатор стиля кривой	

Привязка языка FORTRAN:
CALL CHG_CURVE_STYLE (ENTNAM, EXTSOU, CURSTY)

Результат использования функции

Функция изменяет стиль воспроизведения заданной кривой или твердотельной модели *solid_model* с именем ENTNAM путем его переназначения. Новый стиль воспроизведения определен источником с именем EXTSOU и

идентификатором CURSTY стиля кривой. Источником внешне определенного стиля является настоящий стандарт или любой протокол обмена видами, соответствующий ИСО 13584. При возникновении ошибки стиль текущего воспроизведения не изменяется.

Примечания

1 Стиль воспроизведения сущности кривой определен экземпляром сущности *api_externally_defined_curve_style* с источником *source*, равным EXTSOU, и идентификатором *item_id*, равным CURSTY.

2 Стиль кривой также используется при назначении стиля представления *presentation_style_assignment* сущности твердотельной модели *solid_model*.

3 Если требуемый внешне определенный стиль *externally_defined_style* в настоящем интерфейсе не задан, то используется стиль кривой *plain_solid_line*. При этом ошибка не возникает.

4 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все кривые и твердотельные модели *solid_model*, ссылающиеся на данную группу, должны быть модифицированы.

Внутренние ссылки: 6.1.10, 6.1.19.1, 6.2.4.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	209	Превышение максимально допустимого количества символов в строке
401	Неизвестный источник протокола обмена	402	Неизвестный идентификатор внешнего стиля

A.10.3.3 Изменение стиля воспроизведения заполненной области

Имя функции:	Уровень интерфейса:	1
Chg_Fill_Area_Style	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	AFANAM	N	Имя сущности заполненной области	afa, grp
Ввод	EXTSOU	S	Имя источника, содержащего определение стиля заполненной области	(ISO_13584_31, ISO_13584-1 +)
Ввод	AFASTY	S	Идентификатор стиля заполненной области	

Привязка языка FORTRAN:
CALL CHG_FILL_AREA_STYLE (AFANAM, EXTSOU, AFASTY)

Результат использования функции

Функция изменяет стиль воспроизведения заданной заполненной области *annotation_fill_area* с именем AFANAM путем его переназначения. Новый стиль воспроизведения определен источником *source* с именем EXTSOU и идентификатором AFASTY стиля кривой. Источником внешне определенного стиля является настоящий стандарт или любой протокол обмена видами, соответствующий ИСО 13584. При возникновении ошибки стиль текущего воспроизведения не изменяется.

Примечания

1 Стиль воспроизведения заполненной области определяется экземпляром сущности *api_externally_defined_fill_area_style* с источником *source*, равным EXTSOU, и идентификатором *item_id*, равным AFASTY.

2 Если требуемый внешне определенный стиль *externally_defined_style* в настоящем интерфейсе не задан, то используется стиль заполненной области *opaque_fill_area*. При этом ошибка не возникает.

3 Если заданная сущность AFANAM является экземпляром типа группы *api_group*, то все сущности *annotation_fill_area*, ссылающиеся на данную группу, модифицируются.

Внутренние ссылки: 6.1.15, 6.1.19.1, 6.2.4.3.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	209	Превышение максимально допустимого количества символов в строке
401	Неизвестный источник протокола обмена	402	Неизвестный идентификатор внешнего стиля

А.10.3.4 Изменение стиля воспроизведения поверхности (тела)

Имя функции:	Уровень интерфейса:	1
Chg_Surface_Style	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя поверхности или сущности *solid_model*	aps, solid_model, grp
Ввод	EXTSOU	S	Имя источника, содержащего определение стиля поверхности	(ISO_13584_31, ISO_13584-1 +)
Ввод	SURSTY	S	Идентификатор стиля поверхности	

Привязка языка FORTRAN:
CALL CHG_SURFACE_STYLE (ENTNAM, EXTSOU, SURSTY)

Результат использования функции

Функция изменяет стиль воспроизведения заданной плоской поверхности *api_planar_surface* или твердотельной модели *solid_model* с именем ENTNAM путем его переназначения. Новый стиль воспроизведения определен источником с именем EXTSOU и идентификатором AFASTY стиля поверхности. Источником внешне определенного стиля является настоящий стандарт или любой протокол обмена видами, соответствующий ИСО 13584. При возникновении ошибки стиль текущего воспроизведения не изменяется.

Примечания

1 Стиль воспроизведения поверхности определяется экземпляром сущности *api_externally_defined_surface_style* с источником *source*, равным EXTSOU, и идентификатором *item_id*, равным SURSTY.

2 Стиль поверхности используется также для назначения стиля представления *presentation_style_assignment* твердотельной модели *solid_model*.

3 Если требуемый внешне определенный стиль *externally_defined_style* в настоящем интерфейсе не задан, то используется стиль поверхности *solid_surface*. При этом ошибка не возникает.

4 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все плоские поверхности *api_planar_surface* и твердотельные модели *solid_model*, ссылающиеся на данную группу, модифицируются.

Внутренние ссылки: 6.1.17, 6.1.18, 6.1.19.1, 6.2.3.2, 6.2.4.4.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	209	Превышение максимально допустимого количества символов в строке
401	Неизвестный источник протокола обмена	402	Неизвестный идентификатор внешнего стиля

А.10.3.5 Изменение толщины линий штриховки сущности fill_area_style_hatching

Имя функции:	Уровень интерфейса:	1
Chg_Hatching_Width	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	FSHNAM	N	Имя сущности *fill_area_style_hatching*	fsh, grp
Ввод	WIDTH	S	Метка предварительно определенной толщины линий штриховки, описанной в настоящем стандарте	(thin_hatching_line, middle_thick_hatching_lin, thick_hatching_line)

Привязка языка FORTRAN:
CALL CHG_HATCHING_WIDTH (FSHNAM, WIDTH)

Результат использования функции

Функция изменяет толщину линий штриховки заданной сущности *fill_area_style_hatching* с именем FSHNAM путем ее переназначения. Новая толщина линий штриховки определена именем WIDTH предварительно определенного стиля штриховки. При возникновении ошибки стиль текущего воспроизведения не изменяется.

Примечания

1 Толщина линий штриховки сущности *fill_area_style_hatching* определяется экземпляром сущности *api_pre_defined_hatching_width* с именем WIDTH.

2 Если заданная сущность FSHNAM является экземпляром типа группы *api_group*, то все сущности *fill_area_style_hatching*, ссылающиеся на данную группу, модифицируются.

Внутренние ссылки: 6.2.3.2, 6.2.5.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
6	Значение строки находится вне допустимого диапазона	204	Функция несовместима с текущим уровнем мощности

A.10.3.6 Изменение типа линий штриховки сущности fill_area_style_hatching

Имя функции:	Уровень интерфейса:	1
Chg_Hatching_Curve_Font	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	FSHNAM	N	Имя сущности *fill_area_style_hatching*	fsh, grp
in	FONT	S	Метка предварительно определенного типа линий штриховки, описанного в настоящем стандарте	continuous, dashed, chain, chain double dash, dotted

Привязка языка FORTRAN:
CALL CHG_HATCHING_CURVE_FONT (FSHNAM, FONT)

Результат использования функции

Функция изменяет тип линий штриховки заданной заполненной области *fill_area_style_hatching* с именем FSHNAM путем его переназначения. Новый тип линий штриховки определен именем FONT предварительно определенного типа линий штриховки. При возникновении ошибки стиль текущего воспроизведения не изменяется.

Примечания

1 Тип линий штриховки сущности *fill_area_style_hatching* определен экземпляром сущности *api_pre_defined_hatching_curve_font* с именем FONT.

2 Если заданная сущность FSHNAM является экземпляром типа группы *api_group*, то все сущности *fill_area_style_hatching*, ссылающиеся на данную группу, модифицируются.

Внутренние ссылки: 6.2.3.2, 6.2.5.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
6	Значение строки находится вне допустимого диапазона	204	Функция несовместима с текущим уровнем мощности

А.10.3.7 Изменение цвета штриховки сущности fill_area_style_hatching

Имя функции:	Уровень интерфейса:	1
Chg_Hatching_Colour	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	FSHNAM	N	Имя сущности *fill_area_style_hatching*	fsh, grp
Ввод	COLOUR	S	Метка предварительно определенного цвета штриховки, описанного в настоящем стандарте	hatch_line_colour

Привязка языка FORTRAN:
CALL CHG_HATCHING_COLOUR (FSHNAM, COLOUR)

Результат использования функции

Функция изменяет цвет штриховки *fill_area_style_hatching* сущности FSHNAM путем его переназначения. Новый цвет штриховки определен именем COLOUR предварительно определенного цвета штриховки. При возникновении ошибки стиль текущего воспроизведения не изменяется.

Примечания

1 Цвет штриховки сущности *fill_area_style_hatching* определяется экземпляром сущности *api_pre_defined_hatching_colour* с именем COLOUR.

2 Если заданная сущность FSHNAM является экземпляром типа группы *api_group*, то все сущности *fill_area_style_hatching*, ссылающиеся на данную группу, модифицируются.

Внутренние ссылки: 6.2.3.2, 6.2.5.3.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
6	Значение строки находится вне допустимого диапазона	204	Функция несовместима с текущим уровнем мощности

А.10.3.8 Изменение аспекта невидимых линий

Имя функции:	Уровень интерфейса:	1
Chg_Hidden_Line_Aspect	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности включения невидимых линий (HLI)	curves, afa, pnt, grp
Ввод	HIDSTY	S	Метка стиля невидимых линий, описанного в настоящем стандарте	hidden_line_no_changed, hidden_line_dashed, hidden_line_invisible

Привязка языка FORTRAN:
CALL CHG_HIDDEN_LINE_ASPECT (ENTNAM, HIDSTY)

Результат использования функции

Функция изменяет стиль представления невидимых линий, если сущность HLI с именем ENTNAM включена. При этом атрибут *name* с именем HIDSTY соответствующей сущности *api_pre_defined_occlusion_style* изменяется.

Если глобальные значения записей таблицы статуса интерфейса равны «on» (включено) для записи *hidden_line* и «true» для записи *hidden_line_involved* соответственно, то к сущности ENTNAM подключается сущность *api_pre_defined_occlusion_style* в процессе ее создания. Заданная сущность может быть экземпляром декартовой точки *cartesian_point*, экземпляром кривой или экземпляром заполненной области *annotation_fill_area*. Атрибут *name* корректируется, если два предшествующих утверждения справедливы. В противном случае возникает ошибка и изображение невидимых линий сущности ENTNAM не изменяется.

Примечания

1 Изображение невидимых линий задается сущностью *api_pre_defined_occlusion_style* с атрибутом *name*, равным HIDSTY, и уровнем вида *view_level*, равным текущей записи *view_level* таблицы статуса интерфейса.

2 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все включенные сущности невидимых линий, ссылающиеся на данную группу, модифицируются.

3 Алгоритм включения точек в глобальный процесс удаления невидимых линий в настоящем стандарте не рассматривается. Поэтому если заданная сущность ENTNAM является экземпляром декартовой точки, то интерфейс может изменять или не изменять изображение невидимой линии. Но в любом случае ошибка допустимого типа сущности не возникает.

Внутренние ссылки: 5.3.5, 6.1.12, 6.1.13, 6.1.14, 6.1.15, 6.1.19.1, 6.2.2.1, 6.2.5.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
6	Значение строки находится вне допустимого диапазона	204	Функция несовместима с текущим уровнем мощности

A.10.3.9 Изменение относительного уровня вида для сущности включения невидимых линий

Имя функции:	Уровень интерфейса:	1
Chg_Relative_View_Level	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности включения невидимых линий (HLI)	curves, afa, pnt, grp
Ввод	RVL	D	Виртуальная высота (уровень вида)	

Привязка языка FORTRAN:
CALL CHG_RELATIVE_VIEW_LEVEL (ENTNAM, RVL)

Результат использования функции

Функция изменяет изображение невидимых линий для сущности включения невидимых линий (HLI) с именем ENTNAM путем изменения значения атрибута *view_level* (заданного величиной RVL) сущности *api_pre_defined_occlusion_style*. Это означает, что если глобальные значения записей таблицы статуса интерфейса равны «on» (включено) для записи *hidden_line* и «true» для записи *hidden_line_involved* соответственно, то к сущности ENTNAM подключается сущность *api_pre_defined_occlusion_style* в процессе ее создания. Заданная сущность может быть экземпляром декартовой точки *cartesian_point*, экземпляром кривой или экземпляром заполненной области *annotation_fill_area*. Атрибут *name* корректируется, если два предшествующих утверждения справедливы. В противном случае возникает ошибка и изображение текущих невидимых линий сущности ENTNAM не изменяется.

Примечания

1 Значение |RVL| должно находиться в интервале между *ZERO_value* и EPS.

2 Относительный уровень вида для сущности включения невидимых линий (HLI) определяется атрибутом сущности *api_pre_defined_occlusion_style*. Поэтому атрибут экземпляра данной сущности, подключенного к сущности ENTNAM, должен быть скорректирован.

3 Если заданная сущность ENTNAM является экземпляром типа группы *api_group*, то все сущности HLI, ссылающиеся на данную группу, модифицируются.

4 Алгоритм включения точек в глобальный процесс удаления невидимых линий в настоящем стандарте не рассматривается. Поэтому если заданная сущность ENTNAM является экземпляром декартовой точки *cartesian_point*, то интерфейс может изменять или не изменять изображение невидимой линии. Но в любом случае ошибка диапазона допустимого типа сущности не возникает.

Внутренние ссылки: 5.3.5, 6.1.12, 6.1.13, 6.1.14, 6.1.15, 6.1.19.1, 6.2.2.1, 6.2.5.4.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
7	Действительное значение находится вне допустимого диапазона	204	Функция несовместима с текущим уровнем мощности
404	Стиль затенения невидимых линий не подключен		

А.10.4 Запрос стиля элемента из сущности

Запрос стиля точки из сущности точки	Retrieve_Point_Style
Запрос стиля кривой из сущности кривой (тела)	Retrieve_Curve_Style
Запрос стиля заполненной области из сущности заполненной области	Retrieve_Fill_Area_Style
Запрос стиля поверхности из сущности поверхности (тела)	Retrieve_Surface_Style
Запрос толщины линий штриховки из сущности *fill_area_style_hatching*	Retrieve_Hatching_Width
Запрос типа линий штриховки из сущности *fill_area_style_hatching*	Retrieve_Hatching_Curve_Font
Запрос цвета штриховки из сущности *fill_area_style_hatching*	Retrieve_Hatching_Colour
Запрос аспекта невидимых линий из HLI-сущности	Retrieve_Hidden_Line_Aspect
Запрос относительного уровня вида из HLI-сущности	Retrieve_Relative_View_Level

А.10.4.1 Запрос стиля точки из сущности точки

Имя функции:	Уровень интерфейса:	1
Retrieve_Point_Style	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	PNTNAM	N	Имя сущности *cartesian_point*	pnt
Вывод	EXTSOU	S	Имя источника, содержащего определение стиля точки	(ISO_13584_31, ISO_13584-1 +)
Вывод	PNTSTY	S	Идентификатор стиля точки	

Привязка языка FORTRAN:
CALL RETRIEVE_POINT_STYLE (PNTNAM, EXTSOU, PNTSTY)

Результат использования функции

Функция запрашивает стиль точки из существующей сущности *cartesian_point* с именем PNTNAM. Если возникает ошибка, то выходным значением для параметров EXTSOU и PNTSTY является пустая строка.

Примечание — Стиль воспроизведения точки определен экземпляром сущности *api_externally_defined_point_style* с источником, равным EXTSOU, и идентификатором *item_id*, равным PNTSTY.

Внутренние ссылки: 6.1.9.2, 6.1.19.1, 6.2.4.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	1003	Неправильная длина строки

А.10.4.2 Запрос стиля кривой из сущности кривой (тела)

Имя функции:	Уровень интерфейса:	1
Retrieve_Curve_Style	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности кривой *curve* или твердотельной модели *solid_model*	curves, solid_model
Вывод	EXTSOU	S	Имя источника, содержащего определение стиля кривой	(ISO_13584_31, ISO_13584-1 +)
Вывод	CURSTY	S	Идентификатор стиля кривой	

Привязка языка FORTRAN:
CALL RETRIEVE_CURVE_STYLE (ENTNAM, EXTSOU, CURSTY)

Результат использования функции

Функция запрашивает стиль кривой из существующей сущности кривой *curve* или сущности твердотельной модели *solid_model* с именем ENTNAM. Если возникает ошибка, то выходным значением параметров EXTSOU и CURSTY является пустая строка.

Примечания

1 Стиль воспроизведения кривой определен экземпляром сущности *api_externally_defined_curve_style* с источником, равным EXTSOU, и идентификатором *item_id*, равным CURSTY.

2 Стиль кривой используется также для назначения стиля представления *presentation_style_assignment* для сущности твердотельной модели *solid_model*.

Внутренние ссылки: 6.1.10, 6.1.19.1, 6.2.4.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	1003	Неправильная длина строки

А.10.4.3 Запрос стиля заполненной области из сущности заполненной области

Имя функции:	Уровень интерфейса:	1
Retrieve_Fill_Area_Style	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	AFANAM	N	Имя сущности заполненной области	afa
Вывод	EXTSOU	S	Имя источника, содержащего определение стиля заполненной области	(ISO_13584_31, ISO_13584-1 +)
Вывод	AFASTY	S	Идентификатор стиля заполненной области	

Привязка языка FORTRAN:
CALL RETRIEVE_FILL_AREA_STYLE (AFANAM, EXTSOU, AFASTY)

Результат использования функции

Функция запрашивает стиль заполненной области из существующей сущности *annotation_fill_area* с именем AFANAM. Если возникает ошибка, то выходным значением для параметров EXTSOU и AFASTY является пустая строка.

Примечание — Стиль воспроизведения заполненной области определен экземпляром сущности *api_externally_defined_fill_area_style* с источником, равным EXTSOU, и идентификатором *item_id*, равным AFASTY.

Внутренние ссылки: 6.1.15, 6.1.19.1, 6.2.4.3.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	1003	Неправильная длина строки

А.10.4.4 Запрос стиля поверхности из сущности поверхности (тела)

Имя функции:	Уровень интерфейса:	1
Retrieve_Surface_Style	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности поверхности *api_planar_surface* или твердотельной модели *solid_model*	aps, solid_model
Вывод	EXTSOU	S	Имя источника, содержащего определение стиля поверхности	(ISO_13584_31, ISO_13584-1 +)
Вывод	SURSTY	S	Идентификатор стиля поверхности	

Привязка языка FORTRAN:
CALL RETRIEVE_SURFACE_STYLE (ENTNAM, EXTSOU, SURSTY)

Результат использования функции

Функция запрашивает стиль поверхности из существующей сущности плоской поверхности *api_planar_surface* или сущности твердотельной модели *solid_model* с именем ENTNAM. При возникновении ошибки выходным значением параметров EXTSOU и SURSTY является пустая строка.

Примечания

1 Стиль воспроизведения поверхности определен экземпляром сущности *api_externally_defined_surface_style* с источником, равным EXTSOU, и идентификатором *item_id*, равным SURSTY.

2 Стиль поверхности используется также при назначении стиля представления *presentation_style_assignment* твердотельной модели *solid_model*.

Внутренние ссылки: 6.1.17, 6.1.18, 6.1.19.1, 6.2.3.2, 6.2.4.4.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	1003	Неправильная длина строки

А.10.4.5 Запрос толщины линий штриховки из сущности fill_area_style_hatching

Имя функции:	Уровень интерфейса:	1
Retrieve_Hatching_Width	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	FSHNAM	N	Имя сущности *fill_area_style_hatching*	fsh
Вывод	WIDTH	S	Метка предварительно определенной толщины линий штриховки, описанной в настоящем стандарте	thin_hatching_line, middle_thick_hatching_line, thick_hatching_line

Привязка языка FORTRAN:
CALL RETRIEVE_HATCHING_WIDTH (FSHNAM, WIDTH)

Результат использования функции

Функция запрашивает толщину линий штриховки из существующей сущности *fill_area_style_hatching* с именем FSHNAM. Если возникает ошибка, то выходное значение параметра WIDTH является пустой строкой.

Примечание — Толщина линий штриховки заполненной области *fill_area_style_hatching* определена экземпляром сущности *api_pre_defined_hatching_width* с именем WIDTH.

Внутренние ссылки: 6.2.3.2, 6.2.5.1.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	1003	Неправильная длина строки

А.10.4.6 Запрос типа линий штриховки из сущности fill_area_style_hatching

Имя функции:	Уровень интерфейса:	1
Retrieve_Hatching_Curve_Font	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	FSHNAM	N	Имя сущности *fill_area_style_hatching*	fsh
Вывод	FONT	S	Метка предварительно определенного типа линий штриховки, описанного в настоящем стандарте	continuous, dashed, chain, chain_double_dash, dotted

Привязка языка FORTRAN:
CALL RETRIEVE_HATCHING_CURVE_FONT (FSHNAM, TYPE)

Результат использования функции

Функция запрашивает тип линий штриховки из существующей сущности *fill_area_style_hatching* с именем FSHNAM. Если возникает ошибка, то выходным значением является пустая строка.

Примечание — Тип линий штриховки заполненной области *fill_area_style_hatching* определен экземпляром сущности *api_pre_defined_hatching_curve_font* с именем FONT.

Внутренние ссылки: 6.2.3.2, 6.2.5.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	1003	Неправильная длина строки

А.10.4.7 Запрос цвета штриховки из сущности fill_area_style_hatching

Имя функции:	Уровень интерфейса:	1
Retrieve_Hatching_Colour	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	FSHNAM	N	Имя сущности *fill_area_style_hatching*	fsh
Вывод	COLOUR	S	Метка предварительно определенного цвета штриховки, описанного в настоящем стандарте	hatch_line_colour

Привязка языка FORTRAN:
CALL RETRIEVE_HATCHING_COLOUR (FSHNAM, COLOUR)

Результат использования функции

Функция запрашивает цвет штриховки из существующей сущности *fill_area_style_hatching* с именем FSHNAM. Если возникает ошибка, то выходным значением параметра COLOUR является пустая строка.

Примечание — Цвет штриховки заполненной области *fill_area_style_hatching* определен экземпляром сущности *api_pre_defined_hatching_colour* с именем COLOUR.

Внутренние ссылки: 6.2.3.2, 6.2.5.3.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	1003	Неправильная длина строки

А.10.4.8 Запрос аспекта невидимых линий из HLI-сущности

Имя функции:	Уровень интерфейса:	1
Retrieve_Hidden_Line_Aspect	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности включения невидимых линий	curves, afa, pnt
Вывод	HIDSTY	S	Метка стиля невидимых линий, описанного в настоящем стандарте	hidden_line_no_changed, hidden_line_dashed, hidden_line_invisible

Привязка языка FORTRAN:
CALL RETRIEVE_HIDDEN_LINE_ASPECT (ENTNAM, HIDSTY)

Результат использования функции

Функция запрашивает стиль невидимых линий из существующей HLI-сущности с именем ENTNAM. Заданная сущность может быть экземпляром кривой, заполненной областью *annotation_fill_area* или декартовой точкой *cartesian_point*. При возникновении ошибки выходным значением параметра HIDSTY является пустая строка.

Примечания

1 Если ни один из стилей невидимых линий не подключен к рассматриваемой сущности ENTNAM, то ошибка не возникает, но выходным значением параметра HIDSTY является пустая строка.

2 Изображение невидимых линий определено атрибутом сущности *api_pre_defined_occlusion_style* с именем HIDSTY и уровнем вида *view_level*, равным текущей записи *view_level* из таблицы статуса интерфейса.

3 Включение точек в глобальный процесс удаления невидимых линий в настоящем стандарте не рассматривается.

Внутренние ссылки: 5.3.5, 6.1.12, 6.1.13, 6.1.14, 6.1.15, 6.1.19.1, 6.2.2.1, 6.2.5.4, 8.2.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности	1003	Неправильная длина строки

A.10.4.9 Запрос относительного уровня вида из HLI-сущности

Имя функции:	Уровень интерфейса:	1
Retrieve_Relative_View_Level	Уровень геометрической мощности:	1, 2, 3

Параметры

Ввод/вывод	Имя	Тип данных	Смысл	Допустимый тип/значение
Ввод	ENTNAM	N	Имя сущности включения невидимых линий	curves, afa, pnt
Вывод	RVL	D	Виртуальная высота (уровень вида)	

Привязка языка FORTRAN:
CALL RETRIEVE_RELATIVE_VIEW_LEVEL (ENTNAM, RVL)

Результат использования функции

Функция запрашивает относительный уровень вида RVL подключенного атрибута *api_pre_defined_occlusion_style* существующей сущности включения невидимых линий с именем ENTNAM. При возникновении ошибки выходным значением параметра RVL является пустое множество.

Примечания

1 Если ни один из стилей невидимых линий настоящей сущности ENTNAM не назначен, то ошибка не возникает и выходным значением параметра RVL является пустое множество.

2 Относительный уровень вида для HLI-сущности определен атрибутом *view_level* сущности *api_pre_defined_occlusion_style*, поэтому значение данного атрибута и является выходным.

3 Включение точек в глобальный процесс удаления невидимых линий в настоящем стандарте не рассматривается.

Внутренние ссылки: 5.3.5, 6.1.12, 6.1.13, 6.1.14, 6.1.15, 6.1.19.1, 6.2.2.1, 6.2.5.4.

Ошибки

1	Имя сущности не определено (равно 0 или неизвестно)	2	Недопустимый тип сущности
204	Функция несовместима с текущим уровнем мощности		

Приложение В
(справочное)

Регистрация информационного объекта

В.1 Идентификатор документа

Для обеспечения однозначной идентификации информационных объектов в открытых системах настоящего стандарта присвоен идентификатор:

{ISO standard 13584 part (31) version (1)}.

Значение данного идентификатора определено ИСО 8824-1.

В.2 Идентификатор схемы

Схеме API_ABSTRACT_SCHEMA интерфейса прикладного программирования (см. раздел 6) присвоен идентификатор:

{ISO standard 13584 part (31) version (1) object (1) API-abstract-schema (1)}.

В.3 Идентификатор интерфейса

Интерфейсу программирования, определенному в приложении А, присвоен идентификатор:

{ISO standard 13584 part (31) version (1) object (1) interface (2)}.

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ИСО 128:1982		*
ИСО 1539:1991		*
ИСО/МЭК 8824-11		*
ИСО 10303-11:1994	IDT	ГОСТ Р ИСО 10303-11—2009 «Системы промышленной автоматизации и их интеграция. Представление данных о продукции и обмен данными. Часть 11. Методы описания: справочное руководство по языку EXPRESS»
ИСО 10303-41:1994	IDT	ГОСТ Р ИСО 10303-41—99 «Системы промышленной автоматизации и их интеграция. Представление данных о продукции и обмен данными. Часть 41. Основные понятия описания и поддержки продукта»
ИСО 10303-42:1994		*
ИСО 10303-43:1994	IDT	ГОСТ Р ИСО 10303-43—2002 «Системы промышленной автоматизации и их интеграция. Представление данных о продукции и обмен данными. Часть 43. Интегрированный ресурс: структуры представлений»
ИСО 10303-46:1994	IDT	ГОСТ Р ИСО 10303-46—2002 «Системы промышленной автоматизации и их интеграция. Представление данных о продукции и обмен данными. Часть 46. Интегрированный групповой ресурс: визуальное представление»

* Соответствующий национальный стандарт отсутствует (в разработке). До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов.

Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов:

- IDT — идентичные стандарты.

Библиография

[1] ИСО 4014:1988 (ISO 4014:1988) Болты с шестигранной головкой. Классы изделия A и B (Hexagon head bolts — Product grades A and B)

[2] ИСО 10303–22:1998 (ISO 10303-22:1998) Системы промышленной автоматизации и их интеграция. Представление данных о продукции и обмен данными. Часть 22. Методы реализации. Стандартный интерфейс по доступу к данным (Industrial automation systems — Product data representation and exchange — Part 22: Implementation methods: Standard data access interface)

УДК 658.52.011.56:006.354 ОКС 25.040.40 Т58

Ключевые слова: автоматизированные промышленные системы, интеграция, жизненный цикл систем, управление производством

Редактор *М.Ю. Ярыгина*
Технический редактор *В.Н. Прусакова*
Корректор *Е.Р. Ароян*
Компьютерная верстка *И.В. Белюсенко*

Сдано в набор 09.11.2015. Подписано в печать 25.02.2016 Формат 60 × 84 1/8. Гарнитура Ариал.
Усл. печ. л. 34,88. Уч.-изд. л. 31,74.

Набрано в ИД «Юриспруденция», 115419, Москва, ул. Орджоникидзе, 11.
www.jurisizdat.ru gost.e-ecolog.ru

Издано во ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4
gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru