

ГОСУДАРСТВЕННЫЕ СТАНДАРТЫ
СОЮЗА ССР

ЯЗЫК ПРОГРАММИРОВАНИЯ АЛГОЛ 68 И АЛГОЛ 68 РАСШИРЕННЫЙ

ГОСТ 27974–88, ГОСТ 27975–88

Издание официальное

Б38—88/568, 569

gost.e-ecolog.ru

й КОМИТЕТ СССР ПО СТАНДАРТАМ
Москва

ГОСУДАРСТВЕННЫЕ СТАНДАРТЫ
СОЮЗА ССР

ЯЗЫК ПРОГРАММИРОВАНИЯ АЛГОЛ 68 И АЛГОЛ 68 РАСШИРЕННЫЙ

ГОСТ 27974–88, ГОСТ 27975–88

Издание официальное

МОСКВА – 1989

© Издательство стандартов, 1989

681.3.06:006.354 Группа П85

ГОСУДАРСТВЕННЫЙ СТАНДАРТ СОЮЗА ССР

ЯЗЫК ПРОГРАММИРОВАНИЯ АЛГОЛ 68

Programming language ALGOL 68

ГОСТ 27974–88

ОКСТУ 4002

Дата введения 01.07.90

Настоящий стандарт распространяется на язык программирования Алгол 68* и его варианты и устанавливает требования:

к программе на языке программирования Алгол 68, представленной на машинном носителе или в комплекте программной документации;

к реализациям языка программирования Алгол 68 и его вариантов, используемым при создании или эксплуатации программных средств, в части выполнения программ на языке Алгол 68.

Стандарт не распространяется на варианты языка Алгол 68 и программ на языке Алгол 68, составленные в учебных или исследовательских целях.

Вариантом языка Алгол 68 является определенный конкретной реализацией язык, сохраняющий основные конструкции языка Алгол 68, в описании которого имеется ссылка на настоящий стандарт и четко перечисляются отличия определяемого языка от языка, определенного настоящим стандартом.

Требования к машинному представлению программы приведены в приложении 2. Указатель применяемых в стандарте понятий приведен в приложении 3. Список метаправил приведен в приложении 4.

* Историческая справка о языке Алгол 68 приведена в приложении 1.

Издание официальное ★

Перепечатка воспрещена

1. ЯЗЫК И МЕТАЯЗЫК

1.1. Метод описания

1.1.1. *Введение*

a) Алгол 68 является языком, в котором могут формулироваться алгоритмы для каких-либо вычислителей, т. е. автоматов или людей. Он определяется настоящим стандартом в четыре стадии: „синтаксис" {b}, „семантика" {c}, „представления" {d} и „стандартная языковая обстановка" {e}.

b) Синтаксис есть механизм, посредством которого могут порождаться все конструкты данного языка. Этот механизм действует следующим образом:

(i) Заданы множества „гиперправил" и „метаправил" {1.1.3.4, 1.1.3.3}, из которых могут выводиться „порождающие правила". Входящие в эти правила „метапонятия" и „гиперпонятия" представлены в настоящем стандарте так, что каждое из них выглядит как грамматически правильная русская фраза, возможно с добавлением цифр и специальных знаков, в которой {изменяемые} слова стоят в требуемой грамматической форме. Однако определение синтаксиса строгого языка использует „приведенную форму" этих правил, для получения которой необходимо каждое слово, изменяемое в роде, числе или падеже, заменить на его форму именительного падежа единственного числа и, если возможно, среднего рода {, сохранив время и залог причастий. Полученное слово записывается малыми (большими) синтаксическими знаками, если исходное слово записано малыми (большими) синтаксическими знаками {, причем в последнем случае сохраняются цифры и специальные знаки, приписанные к этому слову}.

{Например, приведенной формой гиперпонятия „УПАКОВКА образованных основ сильно выдающих МАССИВ1 из ВИДА в СРЕДЕ" (3.3.1.d) будет „УПАКОВКА образ основа сильно выдающее МАССИВ1 из ВИД в СРЕДА".}

(ii) „Конструктом в строгом языке" является всякое „дерево порождения" {1.1.3.2.f}, порождаемое применением подмножества указанных порождающих правил; это дерево порождения содержит статическую {, т. е. известную во „время трансляции",} информацию, относящуюся к данному конструкту; дерево составлено из иерархии наследных деревьев порождения, оканчивающихся „символами" на самом нижнем уровне; с каждым деревом порождения связана „среда" из свойств, описанных на предыдущих уровнях и передаваемых к средам его наследников.

(iii) „Программа в строгом языке" есть дерево порождения для понятия 'программа' {2.2.1.a}. Кроме того, она должна соответствовать „языковой обстановке" {10.1.2}.

c) Семантика приписывает каждому конструкту {, т. е. каждому дере-

ву порождения,} „смысл" {2.1.4.1.a}, определяя эффект его „исполнения" {2.1.4.1} (которым, однако, может быть „не определено"). Это происходит следующим образом:

(i) Устанавливают динамическое {, т. е. во время работы программы,} дерево активных „действий" {2.1.4}: в большинстве случаев действием будет исполнение какого-нибудь дерева порождения Т в некотором „окружении", согласующемся со средой этого Т, причем оно может привести к исполнению некоторых наследников Т в подходящих вновь создаваемых наследных окружениях.

(ii) Смысл программы в строгом языке состоит в эффекте ее исполнения в пустом „первичном окружении".

d) Программа в строгом языке должна быть представлена в каком-нибудь „языке представления" {9.3.a}, выбираемом реализатором. В большинстве случаев им будет официальный „эталонный язык".

(i) Всякую программу в языке представления получают заменой всех символов какой-то программы в строгом языке определенными типографскими знаками {9.3}.

(ii) Даже эталонный язык допускает значительную свободу для реализатора {9.4.a,b,c}. Некоторую ограниченную форму эталонного языка, в которой эта свобода не использована, можно назвать „канонической формой" данного языка; предполагают, что она будет применяться для алгоритмов, предназначенных к публикации.

(iii) Смысл программы в языке представления — это смысл той программы {в строгом языке}, из которой она получена.

e) Любой алгоритм выражается посредством собственно-программы, которую вместе с описанной в настоящем стандарте стандартной языковой обстановкой следует рассматривать как вложенную в некоторый текст-программы {10.1.1.a}. Смысл собственно-программы {, в строгом языке или языке представления,} — это смысл программы, „подобной" этому тексту-программы {10.1.2.a}.

1.1.2. *Прагматика*

По разным местам настоящего стандарта рассеяны „прагматические" замечания, заключенные в фигурные скобки „ { " и „ }". Они не входят в определение языка, а служат для того, чтобы помочь понять назначение данных определений и вытекающих из них следствий, а также, чтобы помочь найти соответствующие разделы или правила.

{Некоторые из прагматических замечаний содержат примеры, написанные на эталонном языке. Использующие-индикаторы входят в эти примеры вне контекста своих определяющих-индикаторов. Если не оговорено противное, такие вхождения идентифицируют определяющие-индикаторы, входящие в стандартное- {, библиотечное-} или собственное-вступление и в собственное-заключение (10.2, 10.3, 10.5) (например, см. 10.2.3.12.a для пи, 10.5.1.b для псч и 10.5.2.a для стоп), или же в следующий текст:

```
цел i, j, k, m, n; вещ a, b, x, y; лог p, q,
    переполнение; лит c; формат f; слог r;
    строк s; бит t; компл w, z; имя вещ xx, yy;
    об (цел, вещ) uir; проц пуст задача 1, задача 2;
[1 : n] вещ x1, y1; подв [1 : n] вещ a1;
    [1 : m, 1 : n] вещ x2; [1 : n, 1 : n] вещ y2;
    [1 : n] цел i1; [1 : m, 1 : n] цел i2; [1 : n] компл z1;
проц x или y = имя вещ: если псч < .5 то x иначе y все;
проц ncos = (цел i) вещ: cos (2Х пиХ i/n);
проц nsin = (цел i) вещ: sin (2Х пи Х i/n);
проц финиш = пуст: на стоп;
вид книга = ст (строк текст, имя книга следующая);
    книга проект;
принстон: гренобль: сен пьер де шартрез: коотвейк:
    варшава: зандвоорт: амстердам: тиррения:
    норт бервик: мюнхен: финиш.}
```

1.1.3. *Синтаксис строгого языка*

1.1.3.1. Протопонятия.

а) В определении синтаксиса строгого языка используют формальную грамматику, в которой применяют специальные синтаксические знаки. Эти знаки можно классифицировать следующим образом:

(i) „малые синтаксические знаки", изображаемые в данном стандарте как

„а", „б", „в", „г", „д", „е", „ж", „з", „и", „й", „і", „к", „л", „м", „н", „о", „п", „р", „с", „т", „у", „ф", „х", „ц", „ч", „ш", „щ", „ъ", „ы", „ь", „э", „ю", „я", „(", „)";

(ii) „большие синтаксические знаки", изображаемые в данном стандарте как

„А", „Б", „В", „Г", „Д", „Е", „Ж", „З", „И", „Й", „К", „Л", „М", „Н", „О", „П", „Р", „С", „Т", „У", „Ф", „Х", „Ц", „Ч", „Ш", „Щ", „Ъ", „Ы", „Ь", „Э", „Ю", „Я", „0", „1", „2", „3", „4", „5", „6", „7", „8", „9", „?", „!";

(iii) „прочие синтаксические знаки", изображаемые в данном стандарте как „ . " („точка"), „ , " („запятая"), „ : "(„двоеточие"), „ ; " („точка с запятой"), „ ' " („апостроф"), „ - " („дефис") и „* " („звездочка")

b) „Протопонятие" есть возможно пустая последовательность малых синтаксических знаков.

c) „Понятие" есть {непустое} протопонятие, для которого можно вывести {1.1.3.2.a, 1.1.3.4.d} *порождающее правило*.

d) „Метапонятие" есть {непустая} последовательность больших синтаксических знаков, для которой задано или получено {1.1.3.3.a} какое-нибудь метаправило.

e) „Гиперпонятие" есть возможно пустая последовательность, каждый элемент которой является либо малым синтаксическим знаком, либо метапонятием.

{Таким образом, протопонятия (b) образуют подкласс класса гиперпонятий. Гиперпонятия используют в метаправилах (1.1.3.3), в гиперправилах (1.1.3.4), в качестве парапонятий (1.1.4.2), а также сами по себе, чтобы „обозначать" определенные классы протопонятий (1.1.4.1).}

{„Парапонятие" есть гиперпонятие, к которому применяют определенные специальные соглашения и интерпретации, как разъяснено в 1.1.4.2.}

f) „Символ" есть протопонятие, начинающееся с 'символ'. {Каждое парапонятие символ (9.1.1.h) обозначает конкретное вхождение такого протопонятия.}

g) Для того, чтобы выделить различные использования в тексте настоящего стандарта определенных выше терминов, приняты следующие соглашения:

(i) Внутри порождающих правил, метаправил и гиперправил никакие выделительные знаки {–кавычки, апострофы или дефисы–} не используют.

(ii) Метапонятия и гиперпонятия, рассматриваемые сами по себе, {т.е. не в качестве обозначений протопонятий,} заключают в кавычки.

(iii) Парапонятия не заключают ни во что {, но, проставляют дефисы там, где иначе были бы пробелы}.

(iv) Все остальные гиперпонятия, {включая протопонятия,} не рассмотренные выше, заключают в апострофы {, чтобы указать, что они обозначают некоторое протопонятие, как это определяется в 1.1.4.1.a}.

(v) Особенности типографского набора, такие, как пробел, перенос, переход на новую строчку или страницу, во внимание не принимают (см., однако, 9.4.d).

{Примеры:

(i) ЛОКАЛИЗУЮЩИЙ :: локальный; глобальный; первичный – является метаправилом;

(ii) „ЧИСЛОВОЕ" является метапонятием и не обозначает ничего, кроме самого себя;

(iii) идентификатор-выдающий-имя-ЧИСЛОВОГО, не заключаемый в апострофы, но снабженный дефисами, является парапонятием, обозначающим некоторый конструкт (1.1.4.2.a);

(iv) 'рациональное' является как гиперпонятием, так и протопонятием; рассматриваемое как гиперпонятие оно обозначает самого себя в качестве протопонятия;

(v) 'имя вещественного' значит то же, что и 'имявещественного'.}

1.1.3.2. Порождающие правила и деревья порождения.

a) „Порождающие правила" {b}, которые можно вывести из данных здесь „гиперправил" {1.1.3.4}, составляют {выводимые} порождающие правила настоящего языка; кроме того, некоторые правила неформально указаны в 8.1.4.1.d и 9.2.1.d.

b) Всякое „порождающее правило" состоит из следующих элементов, расположенных в указанном порядке:

возможной звездочки;
непустого протопонятия N;
двоеточия;
непустой последовательности „альтернатив", разделенных точками с запятой;
точки.

Такое правило называют порождающим правилом „для" {этого понятия (1.1.3.1.c)} N.

{Возможная звездочка, если она есть, показывает, что это понятие не используется в других порождающих правилах, а заведено только для того, чтобы облегчить изложение в семантике. Звездочка показывает также, что данное понятие может использоваться как „абстракция" (1.1.4.2.b) одной из своих альтернатив.}

c) Любая „альтернатива" есть непустая последовательность „звеньев", разделенных запятыми.

d) Всякое „звено" есть либо

(i) понятие {, и тогда его можно назвать продуктивным или нетерминальным}, либо

(ii) символ {, который терминален}, либо

(iii) пусто, либо

(iv) другое протопонятие {, для которого нельзя вывести никакого порождающего правила}, называемое в этом случае „тупиком".

{Например, звено 'изображение имени вещественного' (, выводимое из гиперправила 8.0.1.a,) является тупиком.}

{Примеры:

b) порядок: запись десятичного основания,
степень десяти. (8.1.2.1.g) ·
запись десятичного основания:
символ на десять в степени;
символ буква e либо символ буква e лат.
(8.1.2.1.h)

c) запись десятичного основания, степень десяти ·
символ на десять в степени ·
символ буква e либо символ буква e лат

d) запись десятичного основания ·
степень десяти ·
символ на десять в степени ·
символ буква e либо символ буква e лат}

e) „Конструктом в строгом языке" является любое „дерево порождения" {f}, которое можно „породить" из какого-нибудь порождающего правила данного языка.

f) „Дерево порождения" T для понятия N, называемого „прообразом" этого T, „порождается" следующим образом:

· пусть P есть {выводимое} порождающее правило для N;

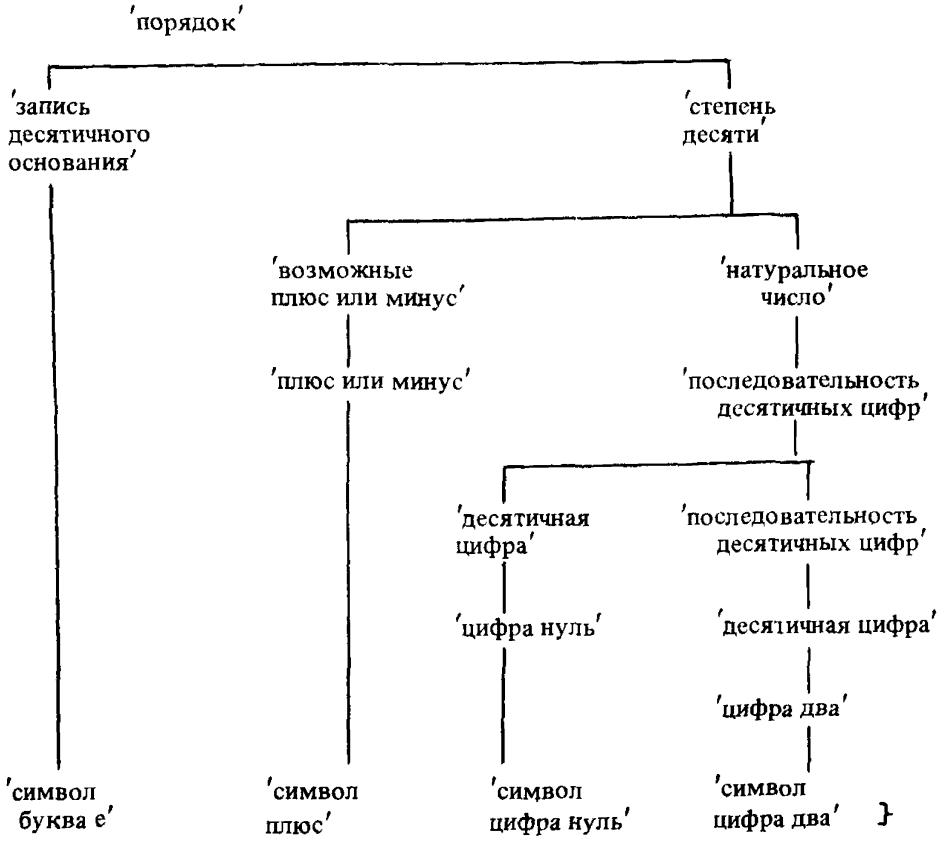
- берется копия N;
- к этой копии присоединяется последовательность деревьев порождения, называемых „прямыми наследниками" дерева Т, порожденных для каждого непустого звена какой-то {одной} альтернативы А правила Р; порядок {деревьев в} этой последовательности совпадает с порядком указанных звеньев в А;
- взятая копия прообраза вместе с присоединенными прямыми наследниками составляет дерево порождения Т.

„Дерево порождения" для символа состоит из копии этого символа {, т.е. оно состоит из какого-то символа}.

„Терминальным порождением" дерева порождения Т является последовательность, составленная из терминальных порождений прямых наследников этого Т, взятых в их порядке.

„Терминальным порождением" дерева порождения, состоящего только из одного символа, является этот символ.

{Пример:



{Терминальным порождением этого дерева будет последовательность символов на его концах. Ее представление в эталонном языке выглядело бы как e+02.}

„Терминальным порождением" понятия является терминальное порождение одного из деревьев порождения для этого понятия {, следовательно, существует много других терминальных порождений 'порядка', кроме показанного выше}.

{Синтаксис строгого языка был выбран так, чтобы данная последовательность символов, являющаяся терминальным порождением какого-то понятия, была таковой в силу существования либо единственного дерева порождения, либо некоторого множества деревьев порождения, отличающихся друг от друга лишь настолько, чтобы исходы их исполнения были одинаковыми (например, деревья порождения, выводимые из правил 3.2.1.e (уравнивание), 1.3.1.d,e (предикаты) и 6.7.1.a.b (выбор способа выписывания вида для приводимого, которое должно опустошаться); (см. также 2.2.2.a).

Поэтому на практике, в настоящем стандарте и в других случаях, вместо деревьев порождения берут терминальные порождения (или их представления). *На самом же деле исполнение программ определяется в семантике настоящего стандарта исходя из деревьев порождения;* семантика посвящена объяснению смысла конструктов, прообразом которых служит понятие 'программа'.}

g) Дерево порождения P является „наследником" дерева порождения Q, если оно прямой наследник {f} либо самого Q, либо некоторого его наследника. Говорят, что Q „содержит" своих наследников и что эти наследники „меньше" Q.

{Например, дерево порождения

'возможные плюс или минус'

|

'плюс или минус'

|

'символ плюс'

входит в качестве наследника в (, и меньше чем,) дерево порождения для 'порядка', (содержащее его и) показанное выше.}

h) Дерево порождения „видимо" („невидимо"), если его терминальное порождение непусто (пусто).

i) Наследник {g} U дерева порождения T расположен „прежде" („после") другого наследника V того же T, если терминальное порождение {f} этого U расположено прежде (после) терминального порождения V в терминальном порождении T. Это {частичное} упорядочение наследников T называется „текстуальным порядком". {В приведенном примере дерева порождения для понятия 'порядок' (f) дерево порождения, прообраз которого есть 'плюс или минус', расположено прежде дерева, прообраз которого есть 'цифра два'.

j) Наследник A дерева порождения „следует" („предшествует") другому наследнику B в некотором текстуальном порядке, если A расположен

после (прежде) B в этом текстуальном порядке и не существует видимого {h} наследника C, расположенного между A и B. {Тем самым подразумевается „непосредственное" следование (предшествование).}

k) Дерево порождения A „подобно" дереву порождения B, если терминальное порождение {f} A совпадает с терминальным порождением B.

1.1.3.3. Метаправила и простая подстановка.

{Метаправила образуют в настоящем языке множество контекстосвободных грамматик, определяющих „метаязык".}

a) „Метаправилами" {b} данного языка служат метаправила {в приведенной форме}, заданные в разделах настоящего стандарта, заголовки которых начинаются со слов „Синтаксис", „Метасинтаксис" или „Метаправила", а также метаправила, получаемые следующим образом:

- для каждого заданного метаправила, относящегося к какому-нибудь метапонятию M, создаются дополнительные правила, каждое из которых состоит из некоторой копии этого M и непосредственно следующего за ней одного из больших синтаксических знаков „0", „1", „2", „3", „4", „5", „6", „7", „8" или „9", за которыми следуют два двоеточия, другая копия метапонятия M и точка. {Таким образом, следует добавить метаправило „ВИД1::ВИД.".}

b) Всякое „метаправило" состоит из следующих элементов, расположенных в указанном порядке:

возможной звездочки;

непустой последовательности M больших синтаксических знаков;

двух двоеточий;

непустой последовательности гиперпонятий {1.1.3.1.e}, разделенных точками с запятой;

точки.

Такое метаправило называют метаправилом „для" {этого метапонятия (1.1.3.1d)} M.

{Звездочка, если она есть, показывает, что это метапонятие не используется в других мета- или гиперправилах, а заведено только для того, чтобы облегчить изложение в семантике.}

{Примеры:

ЧИСЛОВОЕ :: ?РАЗМЕРНОЕ целое;

?РАЗМЕРНОЕ вещественное. (1.2.1.C)

?РАЗМЕРНОЕ :: длинное ?ДЛИННОЕ;

короткое ?КОРОТКОЕ; ПУСТО. (1.2.1.D)}

c) „Терминальное метапорождение" метапонятия M есть любое протопонятие, получаемое „простой подстановкой" {d} из одного из гиперпонятий {, стоящих в правой части} метаправила для M.

d) Протопонятие P получается „простой подстановкой" из гиперпонятия H, если копию {приведенной формы} этого H можно преобразовать в некоторую копию {приведенной формы} P заменой каждого метапонятия M в указанной копии {приведенной формы} H каким-нибудь терминальным метапорождением M.

{Например, двумя возможными терминальными метапорождениями (c) „ЧИСЛОВОГО" будут 'целое' и 'длинное длинное вещественное'. Это объясняется тем, что из гиперпонятий '?РАЗМЕРНОЕ целое' и '?РАЗМЕРНОЕ вещественное' (гиперпонятий метаправила для {приведенной формы} „ЧИСЛОВОГО") можно при помощи простой подстановки (d) вывести 'целое' и 'длинное длинное вещественное'; это в свою очередь возможно потому, что ' ' (пустое протопонятие) и 'длинное длинное' являются терминальными метапорождениями „?РАЗМЕРНОГО".}

{Используемые в настоящем стандарте метапонятия выбраны так, чтобы конкатенация одного или нескольких из них не приводила к той же последовательности больших синтаксических знаков, что и при другой такой конкатенации. Этим устраняется источник возможной неоднозначности.

Хотя рекурсивная природа некоторых метаправил позволяет порождать терминальные метапорождения произвольной длины, длина терминальных метапорождений, вовлекаемых с необходимостью в порождение любой данной программы, конечна.}

1.1.3.4. Гиперправила и согласованная подстановка.

a) Гиперправилами {b} настоящего языка являются гиперправила {в приведенной форме}, заданные в разделах стандарта, заголовки которых начинаются со слова „Синтаксис".

b) Всякое „гиперправило" состоит из следующих элементов, расположенных в указанном порядке:

возможной звездочки;

непустого гиперпонятия H;

двоеточия;

непустой последовательности „гиперальтернатив", разделенных точками с запятой;

точки.

Такое правило называют гиперправилом „для" {этого гиперпонятия (1.1.3.1.e)} H.

c) Всякая „гиперальтернатива" есть непустая последовательность гиперпонятий, разделенных запятыми.

{Примеры:

b) последовательность ПОНЯТИЙ :

ПОНЯТИЕ; ПОНЯТИЕ, последовательность ПОНЯТИЙ. (1.1.3.b)

c) ПОНЯТИЕ, последовательность ПОНЯТИЙ}

d) Порождающее правило PR {1.1.3.2.b} выводится из некоторого гиперправила HR, если копию HR можно преобразовать в копию {приведенной формы} этого PR , применяя „согласованную подстановку" {e} к множеству всех {приведенных форм} гиперпонятий указанной копии HR.

e) Множество {одного или большего числа} протопонятий PP получают, применяя „согласованную подстановку" к соответствующему множеству гиперпонятий HH, если копию HH можно преобразовать в копию {приведенной формы} PP при помощи следующего шага:

Шаг: Если копия {приведенной формы НН} содержит одно или более метапонятий, то для некоторого терминального метапорождения Т одного из этих метапонятий М каждое вхождение М в данную копию заменяется копией этого Т и данный шаг повторяется.

{См. 1.1.4.1.а по поводу другого применения согласованной подстановки.}

{Применяя указанный процесс выведения к данным выше (с) гиперправилам, можно создать правило

последовательность десятичных цифр:

десятичная цифра;

десятичная цифра,

последовательность десятичных цифр,

которое поэтому есть порождающее правило настоящего языка. Отметим, что

последовательность десятичных цифр: десятичная цифра;

десятичная цифра, последовательность букв б.

не является порождающим правилом данного языка, поскольку замена метапонятия „ПОНЯТИЕ" одним из его терминальных метапорождений должна проводиться согласованно повсюду.}

{Так как некоторые метапонятия имеют бесконечное число терминальных метапорождений, число порождающих правил, которые можно вывести, бесконечно. Однако настоящий язык построен так, что для порождения любой программы конечной длины потребуется только конечное число таких порождающих правил.}

{f) Правила в Синтаксисе снабжены „перекрестными ссылками", понимаемыми следующим образом·

Каждое гиперпонятие Н некоторой гиперальтернативы гиперправила А сопровождают ссылками на те гиперправила В, откуда выводятся порождающие правила для понятий, которые можно подставить в это Н. Точно так же гиперпонятия каждого гиперправила В сопровождают обратными ссылками на А. Однако, если Н следует заменить каким-нибудь символом, его сопровождают ссылкой на его представление в п. 9.4.1. Кроме того, вместо ссылок на многие гиперправила в некоторых случаях удобнее сделать ссылку на одно метаправило, и тогда опущенные ссылки можно найти в этом метаправиле.

Каждая такая ссылка служит в принципе номером пункта, за которым идет буква, указывающая строчку с нужным правилом или представлением. При этом используют следующие соглашения:

(i) ссылки, номер пункта в которых совпадает с номером пункта, где они встречаются, дают первыми, и этот номер пункта опускают; например, „8.2.1.а" появляется в п. 8.2.1 как „а";

(ii) опускаются все точки и последняя 1, а 10 заменяют на А; например, „8.2.1.а" входит во все остальные пункты как „82а", а „10.3.4.1.1.i" входит в виде А341 i",

(iii) опускают номер пункта, если он тот же, что и у предыдущей ссылки; например, „82a, 82b, 82c" входят как „82a, b, c";

(iv) посредством „-" отмечают наличие тупика, выводимого из данного гиперпонятия; например, в 8.0.1.a после „изображение ЗНАЧЕНИЯ", поскольку „ЗНАЧЕНИЕ" можно заменить, например, на 'имя вещественного', а 'изображение имени вещественного' не является понятием}

1.1.4. *Семантика*

Семантика определяет „смысл" программ {2.2.1.a} в строгом языке с помощью предложений {некоторого формализованного естественного языка}, устанавливающих, какие „действия" должны проводить во время „исполнения" {2.1.4.1} этих программ. „Смысл" программы в языке представления – это смысл программы в строгом языке, которую она представляет {9.3}.

1.1.4.1. Гиперпонятия, обозначение и заложение.

{Гиперпонятия, заключенные в апострофы, используют, чтобы "обозначать" протопонятия, принадлежащие к определенным классам; например, 'ЛОКАЛИЗУЮЩИЙ' обозначает любое из протопонятий 'локальный', 'первичный' и 'глобальный'.}

a) Находящиеся в тексте данного стандарта гиперпонятия, кроме случаев, когда они входят в гиперправила {1.1.3.4.b} или метаправила {1.1.3.3.b}, „обозначают" любые протопонятия, которые можно получить, применяя к ним согласованную подстановку {1.1.3.4.e}; согласованную подстановку применяют ко всем гиперпонятиям, содержащимся в каждом законченном отрывке текста (это или отрывок, выделяемый буквой со скобкой, если такой есть, или же нумерованный раздел целиком).

{Так, например, 'ОБОЗНАЧЕНИЕ для ПРИЗНАКА' есть гиперпонятие, обозначающее такие протопонятия, как 'буква и лат для целого', 'буква х для вещественного' и т.п. Если в каком-нибудь контексте оно фактически обозначает 'букву и лат для целого', то все вхождения метапонятия „ПРИЗНАК" в текущий отрывок должны обозначать в этом контексте 'целое', а все вхождения „ОБОЗНАЧЕНИЯ" должны обозначать 'букву и лат'. Тогда, например, из отрывка 4.8.2.a можно вывести, что когда „сцена приписывается некоторому определяющему-букву-и-лат-индикатору-выдающему-целое", именно 'буква и лат для целого' „получает доступ к V внутри соответствующего участка".}

Иногда, когда контекст требует этого явно, согласованная подстановка распространяется менее чем на законченный отрывок текста. {Например, во введении к п.2.1.1.2 есть несколько вхождений „'ЗНАЧЕНИЕ'", причем два из них служат для того, чтобы обозначать конкретные (и разные) протопонятия, выписанные полностью, а другие, очевидно, используют, чтобы обозначать различные элементы из класса терминальных метапорождений некоторого „ЗНАЧЕНИЯ".}

b) Если протопонятие (гиперпонятие) P составлено конкатенацией протопонятий (гиперпонятий) A, B и C с возможно пустыми A и C, то P „содержит" B на месте, определяемом в P длиной A. Так, например, 'абвгдевгжз' содержит 'вг' на третьем и седьмом местах.}

c) Протопонятие P2, будучи протопонятием, обозначаемым гиперпонятием H2, „заложено" в протопонятие P1, если P2 или какой-нибудь его эквивалент {2.1.1.2.a} содержится {b} на некотором месте в P1, но не содержится ни на каком месте в любом другом {промежуточном} протопонятии P3, также содержащемся в P1 и таком, что H2 может обозначать и это P3.

{Так, например, 'ВИД', заложенный в 'замкнутое предложение выдающее имя вещественного' есть 'имя вещественного', а не 'вещественное'; кроме того, в вид (2.1.1.2.b), специфицируемый описателем ст (вещ a, ст (лог b, лит c) d), заложены только два 'ПОЛЯ'.}

1.1.4.2. Парапонятия.

{„Парапонятия" введены в данном стандарте, чтобы облегчить рассмотрение конструктов с определенными прообразами. Парапонятие – это правильная фраза (русского языка), обозначающая конструкты (1.1.3.2.e); смысл парапонятия не обязательно тот, который можно найти в словаре, его можно вывести из приведенных ниже правил.}

a) „Парапонятие" P есть {не заключенное в апострофы} гиперпонятие, используемое в тексте данного стандарта, чтобы „обозначать" любой конструкт, прообраз O которого удовлетворяет следующему условию;

- рассматриваемое как гиперпонятие {, т.е. как если бы оно было заключено в апострофы,} P обозначает {1.1.4.1.a} некоторую „абстракцию" {b} прообраза O.

{Например, парапонятие „натуральное-число" могло бы обозначать конструкт, имеющий представление 02, поскольку если бы оно было в апострофах, то обозначало бы абстракцию понятия 'натуральное число', являющегося прообразом данного конструкта. Однако то же представление можно было бы описать и как последовательность-десятичных-цифр и тогда оно было бы прямым наследником этого натурального-числа.}

{Чтобы было легче выделять парапонятия среди других гиперпонятий, они не заключены в апострофы и снабжены дефисами там, где иначе были бы пробелы.}

Изменение по роду, числу и/или падежу слов, составляющих парапонятия {, рассматриваемые в качестве конструкций русского языка}, не влечет за собою изменения смысла этих парапонятий. {Таким образом, вхождение парапонятия в какое-нибудь предложение в тексте настоящего стандарта подчинено правилам согласования слов в предложениях русского языка.}

Малый синтаксический знак в начале парапонятия часто {, например, в начале предложения,} заменен {, чтобы улучшить вид текста,} соответствующим большим синтаксическим знаком без изменения смысла этого парапонятия {; например, „Идентификатор" имеет тот же смысл, что и „идентификатор"}.

b) Протопонятие P2 есть „абстракция" протопонятия P1, если

(i) P2 есть абстракция понятия, порождающее правило для которого начинается со звездочки, а P1 – одна из альтернатив этого правила.

{например, 'ограничение' (5.3.2.1.h) есть абстракция любого из понятий, обозначаемых гиперпонятиями, 'отрезок в СРЕДЕ', 'индекс в СРЕДЕ', 'возможная сдвинутая нижняя граница в СРЕДЕ' , или

(ii) в P1 заложено протопонятие P3, обозначаемое одним из „опускаемых гиперпонятий", перечисленных ниже в п. c), а P2 – абстракция протопонятия, состоящего из P1 без этого заложенного P3

{например, 'старт выбирающего по логическому' есть абстракция понятий 'краткий старт выбирающего по логическому' и 'выделенный старт выбирающего по логическому' (с опущенным 'ОФОРМЛЕННОЕ' из 9.1.1.a) }, или

(iii) P2 эквивалентно {2.1.1.2.a} P1

{например, 'символ выделенное начало' есть абстракция протопонятия 'символ выделенное начало'}.

{Чтобы привести пример, включающий все три правила, заметим, что 'определяющий индикатор выдающий объединение целого вещественного воедино' есть абстракция некоторого 'определяющего букву а идентификатора в СРЕДЕ выдающего объединение вещественного целого воедино' (4.8.1.a). 'Краткий старт выбирающего по объединению целого вещественного воедино' не является абстракцией понятия 'краткий старт выбирающего по объединению целого вещественного логического воедино', потому что 'логического', которое, очевидно, опущего, не есть 'ЗНАЧЕНИЕ', заложенное в это понятие.}

c) Упомянутые выше в разд. b) „опускаемые гиперпонятия" следующие:

„ОФОРМЛЕННОЕ" • „НОМЕР" • „ЛОКАЛИЗУЮЩИЙ" • „ПРИМЕНЯЮЩИЙ" • „ЛЮБОЙ" • „ПРИВОДИМО" • „ЗНАЧЕНИЕ" • „для ЗНАЧЕНИЯ" • „выдающее ИМЯ ПРОВИДА" • „для метки" • „для процедуры" • „вида ПРОВИД" • „в СРЕДЕ" • „!ПАРЫ" • „с ?ОПИСАНИЯМИ ?МЕТКАМИ" • „через ?ОПИСАНИЯ ?МЕТКИ" • „определяющее СЛОЙ" • „ОБОЗНАЧЕНИЕ" • „как ИМЯ ПРОВИДА".

{Какое из нескольких возможных понятий или символов служит прообразом конструкта, обозначаемого данным парапонятием, выясняют из контекста, в котором это парапонятие встречают. Например, когда говорится о формальном-описателе какого-то описания-тождества, его прообразом будет некоторое понятие, обозначаемое гиперпонятием 'формальный описатель имени вещественного в СРЕДЕ', если терминальное порождение (1.1.3.2.f) этого описания-тождества есть имя вещ x= лок вещ.}

{Так как всякое парапонятие обозначает некоторый конструкт, то все определенные для конструктов технические термины можно без дополнительной формализации применять и к парапонятиям.}

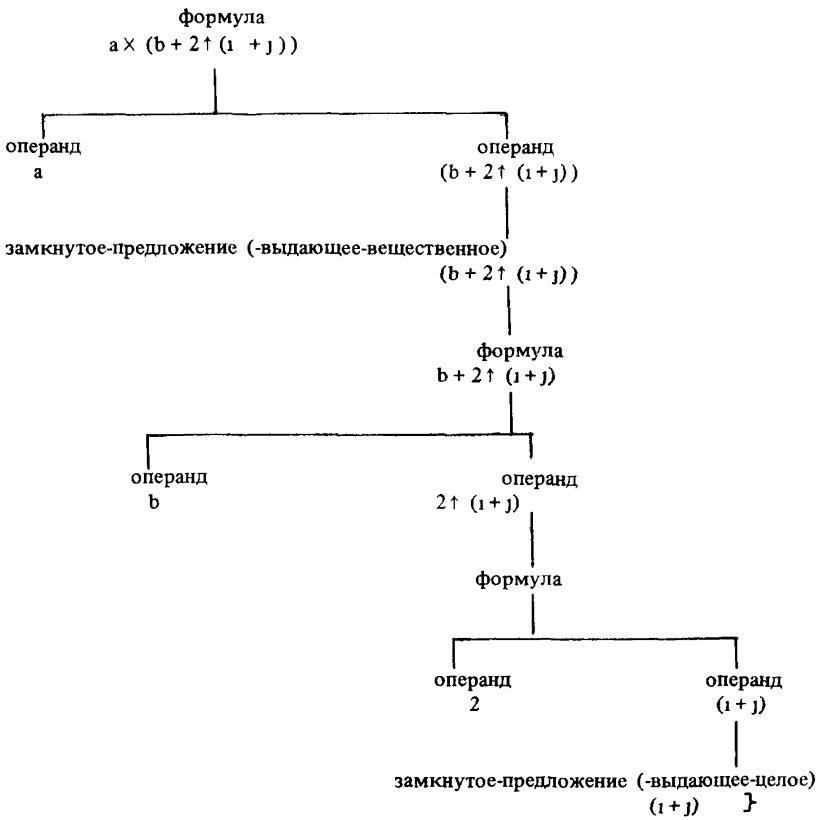
d) Если два парапонятия P и Q обозначают два конструкта S и T соответственно, то P называют „составляющим" Q, если S – наследник T и нет такого {промежуточного конструкта} U, что

(i) S – наследник U,

(ii) U – наследник T и

(iii) P или Q может {в равной мере} обозначать U

{Так например, a (S1) – это составляющий операнд формулы a×(b+2↑(1+j)) (T), a b (S2) – нет, поскольку это наследник промежуточной формулы b + 2↑(1 + j) (U), которая сама есть наследник T Аналогично (b + 2↑(1 + j)) – составляющее замкнутое-предложение формулы T, а замкнутое-предложение (1 + j) – нет, поскольку это наследник промужеточного замкнутого-предложения Однако (1 + j) – составляющее замкнутое-предложение-выдающее-целое формулы T, так как указанное промежуточное замкнутое-предложение фактически является замкнутым-предложением выдающим-вещественное



1.1.4.3. Неопределенности.

a) Если что-то оставлено „не определенным" или о чем-то сказано, что оно „не определено", то это означает, что оно не определено лишь настоящим стандартом и для его определения надо принять в расчет какую-нибудь информацию, находящуюся вне этого стандарта.

{Необходимо отличать выдачу неопределенного значения (, после чего исполнение продолжается с возможно непредсказуемыми результатами,) от полной неопределенности дальнейшего исполнения. Действия, которые следует предпринять в последнем случае, оставляются на усмотрение реализатора. Это может быть одна из форм продолжения (, не обязательно одинаковая в разных реализациях,) или некоторая форма прерывания (2.1.4.3.h), осуществляемая какой-нибудь проверкой во время работы программы.}

b) Если „требуется", чтобы какое-то условие удовлетворялось при некотором исполнении, то дальнейшее исполнение не определено, если это условие не удовлетворяется.

c) „Осмысленная" программа – это программа {2.2.1.a}, исполнение которой определено настоящим стандартом.

1.2. Общие метаправила

1.2.1. *Метаправила для видов*

A) ВИД :: ПРОСТОЕ; СОСТАВНОЕ; ИМЯ ВИДА; ПРОЦЕДУРА; ПРЕДСТАВИТЕЛЬ; ЦИ определение ВИДА; использование ЦИ.
B) ПРОСТОЕ :: ЧИСЛОВОЕ; логическое; литерное.
C) ЧИСЛОВОЕ :: ?РАЗМЕРНОЕ целое; ?РАЗМЕРНОЕ вещественное.
D) ?РАЗМЕРНОЕ :: длинное ?ДЛИННОЕ; короткое ?КОРОТКОЕ; ПУСТО.
E) ?ДЛИННОЕ :: длинное ?ДЛИННОЕ; ПУСТО.
F) ?КОРОТКОЕ :: короткое ?КОРОТКОЕ; ПУСТО.
G) ПУСТО :: .
H) СОСТАВНОЕ :: структура содержащая !ПОЛЯ в себе; ?ПОДВИЖНЫЙ МАССИВ из ВИДА.
I) !ПОЛЯ :: ПОЛЕ; !ПОЛЯ ПОЛЕ.
J) ПОЛЕ :: СЛОВО {942A} для выборки ВИДА.
K) ?ПОДВИЖНОЕ :: подвижное; ПУСТО.
L) МАССИВ :: вектор; МАССИВ векторов.
M) ИМЯ :: имя; временное имя.
N) ПРОЦЕДУРА :: процедура ?ПАРАМЕТРИЗОВАННАЯ вырабатывающая ЗНАЧЕНИЕ.
O) ?ПАРАМЕТРИЗОВАННАЯ :: с! ПАРАМЕТРАМИ; ПУСТО.
P) !ПАРАМЕТРЫ :: ПАРАМЕТР; !ПАРАМЕТРЫ ПАРАМЕТР.
Q) ПАРАМЕТР :: параметр вида ВИД.
R) ЗНАЧЕНИЕ :: ВИД; пустое значение.
S) ПРЕДСТАВИТЕЛЬ :: объединение !ОБЫЧНЫХ воедино.
T) !ОБЫЧНЫЕ :: ОБЫЧНОЕ; !ОБЫЧНЫЕ ОБЫЧНОЕ.
U) ОБЫЧНОЕ :: ПРОСТОЕ; СОСТАВНОЕ; имя ВИДА; ПРОЦЕДУРА; пустое значение.

V) ЦИ :: ци НОМЕР.
W) НОМЕР :: I; НОМЕР.

1.2.2. *Метаправила, связанные с фразами и приведением*

A) ЗАКРЫТОЕ :: замкнутое; совместное; параллельное; ВЫБИРАЮЩЕЕ {34A}; циклическое.
B) ДЕЙСТВУЮЩЕЕ :: в СРЕДЕ ПРИВОДИМО выдающее ЗНАЧЕНИЕ.
C) ПРИВОДИМО : сильно; крепко; раскрыто; слабо; мягко.

1.2.3. *Метаправила, связанные со средами*

A) СРЕДА :: СЛОЙ; СРЕДА с СЛОЕМ.
B) СЛОЙ :: новые ?ОПИСАНИЯ ?МЕТКИ.
C) ?ОПИСАНИЯ :: ! ОПИСАНИЯ; ПУСТО.
D) !ОПИСАНИЯ :: ОПИСАНИЕ; !ОПИСАНИЯ ОПИСАНИЕ.
E) ОПИСАНИЕ :: СЛОВО {942A} для ВИДА; ИНФИКС {942F} для приоритета ПРИОРИТЕТ; ИНДИКАНТ {942D} для ЗНАЧЕНИЯ НОМЕР; ИНФИКС {942F} для ДВУМЕСТНОЙ; ПРЕФИКС {942K} для ОДНОМЕСТНОЙ.
F) ПРИОРИТЕТ :: I; II; III; III I; III II; III III; III III I; III III II; III III III.
G) ОДНОМЕСТНАЯ :: процедура с ПАРАМЕТРОМ вырабатывающая ЗНАЧЕНИЕ.
H) ДВУМЕСТНАЯ :: процедура с ПАРАМЕТРОМ ПАРАМЕТРОМ2 вырабатывающая ЗНАЧЕНИЕ.
I) ?МЕТКИ :: !МЕТКИ; ПУСТО.
J) !МЕТКИ :: МЕТКА; !МЕТКИ МЕТКА.
K) МЕТКА :: СЛОВО {942A} для метки.

1.3. Общие гиперправила

{Предикаты используют в синтаксисе, чтобы обеспечить в деревьях порождения соблюдение определенных ограничений типа того, что каждый использующий-индикатор должен идентифицировать некоторый однозначно находимый определяющий-индикатор. Их более скромная цель состоит в уменьшении числа гиперправил при помощи группировки нескольких сходных случаев в качестве альтернатив одного правила. В этих случаях с помощью предикатов можно выяснить, какую из альтернатив применить.}

1.3.1. *Синтаксис общих предикатов*

A) ПОНЯТИЕ :: ЛИТЕРА; ПОНЯТИЕ ЛИТЕРА.
B) ЛИТЕРА :: а; б: в; г; д; е; ж; з; и; й; к; л; м; н; о; п; р; с; т; у; ф; х; ц; ч; ш; щ; ъ; ы; ь; э; ю; я; I.
C) ?ПОНЯТИЕ :: ПОНЯТИЕ; ПУСТО.
D) УТВЕРЖДЕНИЕ :: ПОНЯТИЕ; (?ПОНЯТИЕ) ?ПОНЯТИЕ2;
УТВЕРЖДЕНИЕ (?ПОНЯТИЕ1) ?ПОНЯТИЕ2.
E) ЕСЛИ :: если; если неверно что.
a) если истина : ПУСТО.
b) если неверно что ложь : ПУСТО.
c) если УТВЕРЖДЕНИЕ1 и УТВЕРЖДЕНИЕ2:
если УТВЕРЖДЕНИЕ1, если УТВЕРЖДЕНИЕ2.

d) если УТВЕРЖДЕНИЕ1 или УТВЕРЖДЕНИЕ2: если УТВЕРЖДЕНИЕ1; если УТВЕРЖДЕНИЕ2.

e) если неверно что УТВЕРЖДЕНИЕ1 и УТВЕРЖДЕНИЕ2: если неверно что УТВЕРЖДЕНИЕ1; если неверно что УТВЕРЖДЕНИЕ2.

f) если неверно что УТВЕРЖДЕНИЕ1 или УТВЕРЖДЕНИЕ2: если неверно что УТВЕРЖДЕНИЕ1, если неверно что УТВЕРЖДЕНИЕ2.

g) ЕСЛИ (?ПОНЯТИЕ1) есть (?ПОНЯТИЕ2) :
ЕСЛИ (?ПОНЯТИЕ1) начинается с (?ПОНЯТИЯ2) {h, i, j}
и (?ПОНЯТИЕ2) начинается с (?ПОНЯТИЯ1) {h, i, j}.

h) ЕСЛИ (ПУСТО) начинается с (ПОНЯТИЯ) {g, j} :
ЕСЛИ ложь {b, -}.

i) ЕСЛИ (?ПОНЯТИЕ) начинается с (ПУСТО) {g, j}:
ЕСЛИ истина {a, -}.

j) ЕСЛИ (ЛИТЕРА1 ?ПОНЯТИЕ1) начинается с (ЛИТЕРЫ2 ?ПОНЯТИЯ2) {g, j, m}:
ЕСЛИ (ЛИТЕРА1) совпадает с (ЛИТЕРОЙ2) в
(абвгдежзийклмнопрстуфкцчшщьыъэюяЇ) {k, l, -}
и (ПОНЯТИЕ1) начинается с (ПОНЯТИЯ2) {h, i, j}.

k) если (ЛИТЕРА) совпадает с (ЛИТЕРОЙ) в (ПОНЯТИИ) {j}:
если истина {a}

l) если неверно что (ЛИТЕРА1) совпадает с (ЛИТЕРОЙ2) в (ПОНЯТИИ) {j}:
если (ПОНЯТИЕ) содержит (ЛИТЕРУ1 ?ПОНЯТИЕ ЛИТЕРУ2) {m}
или (ПОНЯТИЕ) содержит (ЛИТЕРУ2 ?ПОНЯТИЕ ЛИТЕРУ1) {m}.

m) ЕСЛИ (ЛИТЕРА ?ПОНЯТИЕ) содержит (ПОНЯТИЕ) {l, m}
ЕСЛИ (ЛИТЕРА ?ПОНЯТИЕ) начинается с (ПОНЯТИЯ) {j}
или (?ПОНЯТИЕ) содержит (ПОНЯТИЕ) {m, n}.

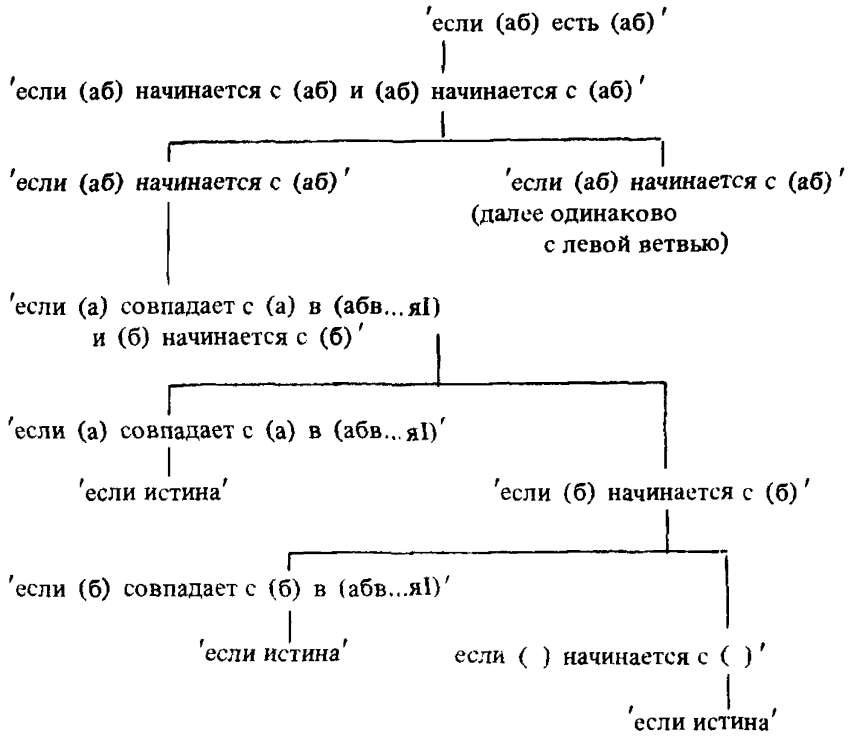
n) ЕСЛИ (ПУСТО) содержит (ПОНЯТИЕ) {m}:
ЕСЛИ ложь {b, -}.

{Малые синтаксические знаки „("и„)" используют, чтобы простым способом обеспечить однозначное применение этих предикатов.}

1.3.2. *Выполнимость предикатов*

Всякий „предикат" есть протопонятие, начинающееся с 'если' или 'если неверно что' {объединяемых в 'ЕСЛИ'}. Для каждого предиката Р либо можно породить одно или несколько деревьев порождения {1.1.3.2.f} {, каждое из которых невидимо,} и тогда Р „выполняется", либо нельзя породить никакого дерева порождения {, поскольку каждая попытка породить какое-нибудь из них заводит в тупик,} и тогда Р „не выполняется".

{Например, предикат 'если (аб) есть (аб)' выполняется. Его дерево порождения можно изобразить так:



Если предикат выполняется, то его дерево порождения всегда оканчивается на 'если истина' или 'если неверно что ложь'. Если он не выполняется, то тупиками обычно оказываются 'если ложь' и 'если неверно что истина'. Хотя почти все соответствующие гиперправила написаны для гиперпонятий, начинающихся с „ЕСЛИ" и потому каждый раз обеспечивают порождающие правила для пары предикатов типа 'если УТВЕРЖДЕНИЕ1' и 'если неверно что УТВЕРЖДЕНИЕ1', это не значит, что в каждом таком случае должен выполняться какой-то один из этой пары предикатов. Например, предикат 'если цифра четыре считает III' {4.3.1.c} не выполняется, но не приняты никакие меры для того, чтобы выполнялся предикат 'если неверно что цифра четыре считает III', поскольку в настоящем стандарте он не применяется.

В семантике не приписывают никакого смысла конструктам, прообразами которых служат предикаты. Они нужны для чисто синтаксических целей.}

1.3.3. *Синтаксис общих конструкций*

A) ОФОРМЛЕННОЕ :: краткое; выделенное; стиля НОМЕР.

a) возможное ПОНЯТИЕ : ПОНЯТИЕ; ПУСТО.

b) последовательность ПОНЯТИЙ {b} : ПОНЯТИЕ; ПОНЯТИЕ, последовательность ПОНЯТИЙ {b}.

c) список ПОНЯТИЙ {c} : ПОНЯТИЕ; ПОНЯТИЕ, знак а также {94f}, список ПОНЯТИЙ {c}.

d) упакованное ОФОРМЛЕННОЕ ?ПОНЯТИЕ : знак начало ОФОРМЛЕННЫЙ {94f, -}, ?ПОНЯТИЕ, знак конец ОФОРМЛЕННЫЙ {94f, -}.

e) индексирующее ОФОРМЛЕННОЕ ПОНЯТИЕ : знак открыть индексы ОФОРМЛЕННЫЙ {94f, -}, ПОНЯТИЕ, знак закрыть индексы ОФОРМЛЕННЫЙ {94f, -}.

f) УТВЕРЖДЕНИЕ1 либо УТВЕРЖДЕНИЕ2: УТВЕРЖДЕНИЕ1; УТВЕРЖДЕНИЕ2.

{Из этого синтаксиса прямо следует, что имеются такие порождающие правила, как

последовательность десятичных цифр: десятичная цифра;

десятичная цифра, последовательность десятичных цифр.

(которое используют в порождении примера в 1.1.3.2.f, но для которого больше нет никакого явного порождающего правила.) Таким образом уменьшено число гиперправил, фактически выписанных в настоящем стандарте, а оставшиеся, стали нагляднее, так как эти общие конструкции выписаны словами, позволяющими предположить, какими должны быть их порождения.

По той же причине ссылки (1.1.3.4.f) на эти правила заменены более содержательными ссылками; например, вместо „последовательность десятичных цифр {133b}" в 8.1.1.1.b дано более содержательное „последовательность десятичных цифр {c}". Кроме того, ссылки внутри самих общих конструкций ограничены необходимым минимумом.}

2. ВЫЧИСЛИТЕЛЬ И ПРОГРАММА

Смысл программы в строгом языке объясняют в терминах гипотетического вычислителя, который осуществляет множество действий {2.1.4}, составляющих исполнение {2.1.4.1} этой программы. Вычислитель работает с некоторым множеством „объектов" {2.1.1}.

2.1. Терминология

2.1.1. *Объекты*

„Объект" является или конструктом {1.1.3.2.e}, или „значением" {2.1.1.1.a}, или „участком" {2.1.1.1.b}, или „окружением" {2.1.1.1.c}, или „сценой" {2.1.1.1.d}.

{Конструкты можно отнести к „внешним объектам", так как они соответствуют тексту программы, которая для более реальных вычислителей могла бы транслироваться в какую-нибудь внутреннюю форму, в которой она была бы способна действовать с „внутренними объектами", а именно со значениями, участками, окружениями и сценами. Однако рассматриваемый здесь гипотетический вычислитель не нуждается в фазе трансляции. Предполагают, что он может анализировать программу и все ее наследные конструкты тогда же, когда он обрабатывает внутренние объекты.}

2.1.1.1. Значения, участки, окружения и сцены

а) Всякое „значение" является или „простым значением" {2.1.3.1}, или

„именем" {2.1.3.2}, или „составным значением" (т.е. „структурой {2.1.3.3} или „массивом" {2.1.3.4}), или „процедурой" {2.1.3.5}.

b) Всякий „участок" {есть внутренний объект, который} соответствует каким-то '?ОПИСАНИЯМ ?МЕТКАМ' {1.2.3.C,I}. „Незанятый участок" – это участок, для которого '?ОПИСАНИЯ ?МЕТКИ' есть 'ПУСТО'.

{Каждое 'ОБОЗНАЧЕНИЕ для ПРИЗНАКА' (4.8.1.F,G), заложенное в данные '?ОПИСАНИЯ ?МЕТКИ', соответствует определяющему-ОБОЗНАЧЕНИЕ-индикатору-выдающему-ПРИЗНАК (т.е. какому-нибудь идентификатору, обозначению-операции или индикатору-вида), описанному в конструкте, исполнение которого вызвало создание данного участка. Указанное 'ОБОЗНАЧЕНИЕ для ПРИЗНАКА' может „получить доступ" к какому-то значению или сцене „внутри" этого участка (2.1.2.c).

Образом участка может служить ряд ячеек памяти, в которые помещены эти доступные объекты}

{Все терминальные метапорождения метапонятий „ОПИСАНИЕ", „МЕТКА" и „ПОЛЕ" (, или включающее их и чаще употребляемое метапонятие „ПАРА",) имеют форму 'ОБОЗНАЧЕНИЕ для ПРИЗНАКА'. Обозначаемые 'ПАРАМИ' „свойства" применяют в синтаксисе и семантике, чтобы в конкретной ситуации связывать с таким 'ОБОЗНАЧЕНИЕМ' определенный признак.}

c) Всякое „окружение" или пусто, или составлено из {какого-либо другого} окружения и некоторого участка.

{Поэтому каждое окружение выводится из ряда других окружений, вытекающих в конечном счете из пустого „первичного окружения", в котором исполняется программа (2.2.2.a).}

d) Всякая „сцена" S есть объект, составленный из конструкта C {1.1.3.2.e} и окружения E. C называется конструктом, а E окружением „этого" S {или окружением „и" S}.

{Доступ к сценам внутри участков (2.1.2.c) осуществляется через 'МЕТКУ' или 'ОПИСАНИЕ', возникающие из идентификаторов-выдающих-метку или индикаторов-вида; сцены могут также быть значениями (2.1.3.5).}

2.1.1.2. Виды.

{Каждое значение имеет атрибут, называемый его „видом". Он определяет, как это значение связано с другими значениями и какие с ним можно производить действия. Этот атрибут описывают или, точнее, „выписывают" посредством какого-нибудь 'ЗНАЧЕНИЯ' (1.2.1.R) (; например, существует вид, выписываемый как 'вещественное', и вид, выписываемый как 'структура содержащая букву эр лат букву е лат для выборки вещественного букву и лат букву эм лат для выборки вещественного в себе'). Поскольку преследуется цель, чтобы виды, специфицируемые индикаторами-вида a и b в

вид a =ст (имя a b),

вид b =ст (имя ст (имя b b) b

фактически были одинаковыми, то необходимо, чтобы как 'ЗНАЧЕНИЕ'

'ци I определение структуры
содержащей букву бе лат для
выборки имени использования ци I в себе',
так и 'ЗНАЧЕНИЕ'
'ци II определение структуры
содержащей букву бе лат для
выборки имени структуры
содержащей букву бе лат для
выборки имени использования ци II в себе в себе'
(а на самом деле и многие другие) были возможными выписываниями одного и того же вида. Аналогично специфицируемый описателем об (цел, вещ) вид можно выписать как 'объединение целого вещественного воедино', так и 'объединение вещественного целого воедино'. Все протопонятия 'ЗНАЧЕНИЕ', выписывающие один и тот же вид, называются „эквивалентными" (а).

Некоторые протопонятия 'ЗНАЧЕНИЕ', такие, как 'имя использования ци III', 'имя ци III I определения имени использования ци III I', 'объединение вещественного имени вещественного воедино' и 'структура содержащая букву а для выборки целого букву а для выборки вещественного в себе', не являются правильно построенными (7.4, 4.7. 1.f, 4.8.1.c) и не выписывают никакого вида.

Хотя для большинства практических применений „вид" можно рассматривать просто как 'ЗНАЧЕНИЕ', его строгое определение включает целый класс протопонятий 'ЗНАЧЕНИЕ', эквивалентных друг другу, и любое из них может описывать этот вид.}

а) 'ЗНАЧЕНИЕ' {1.2.1.R} „эквивалентно" 'ЗНАЧЕНИЮ2', если выполняется {1.3.2} предикат 'если ЗНАЧЕНИЕ1 эквивалентно ЗНАЧЕНИЮ' {7.3.1.a}.

{Правильно построенное 'ЗНАЧЕНИЕ' всегда эквивалентно самому себе; 'объединение целого вещественного воедино' эквивалентно 'объединению вещественного целого воедино'.}

Протопонятие P „эквивалентно протопонятию Q, если можно преобразовать копию Pc {приведенной формы} протопонятия P в копию Qc {приведенной формы} протопонятия Q при помощи следующего шага:

Шаг: Если Pc не совпадает с Qc, некоторое 'ЗНАЧЕНИЕ1', содержащееся в Pc, но не содержащееся ни в каком {большем} 'ЗНАЧЕНИИ2', содержащемся в Pc, заменяют некоторым эквивалентным ему 'ЗНАЧЕНИЕМ' и этот Шаг предпринимают снова.

{Таким образом, 'идентификатор выдающий объединение целого вещественного воедино' эквивалентен 'идентификатору выдающему объединение вещественного целого воедино'.}

b) Всякий „вид" есть такой класс C протопонятий 'ЗНАЧЕНИЕ', что каждый его элемент эквивалентен {a} каждому другому его элементу, а также {, чтобы обеспечить правильность построения}, самому себе, но не эквивалентен никакому 'ЗНАЧЕНИЮ1', не принадлежащему C.

{Однако если эквивалентность видов специально не обсуждается, можно говорить о виде просто как о терминальном метапорождении „ЗНАЧЕНИЯ" в силу сокращения, которое будет дано в 2.1.5.f.}

c) Каждое значение имеет один конкретный вид.

{Например, видом значения 3.14 является 'вещественное'. Не существует, однако, значений, вид которых начинается с 'объединение', 'временное имя' или 'подвижный МАССИВ из' (см. 2.1.3.6).}

2.1.1.3. Области действия.

{Значение V может „именовать" (2.1.2.e) другой внутренний объект O или быть составлено из (2.1.1.1.d) него (, например, имя может именовать значение, а такая сцена, как процедура, частично составлена из окружения). Далее, время жизни ячеек памяти, содержащих (2.1.3.2.a) этот объект O или используемых в связи с ним (2.1.1.1.b), может оказаться ограниченным (, поскольку спустя некоторое время они могут обновиться), и потому нельзя, чтобы V сохранялось дольше этого времени жизни, так как иначе можно было бы попытаться достичь через V какую-то уже не существующую ячейку памяти. Чтобы выразить это ограничение, говорят, что если V должно быть „присвоено" (5.1.2.1.b) какому-нибудь имени W, то „область действия" W не должна быть „старше" области действия V. Таким образом, область действия значения V есть мера возраста указанных ячеек памяти и, следовательно, их времени жизни.}

a) Каждое значение имеет одну определенную „область действия" {, зависящую от его вида и от способа, которым оно создано; область действия значения определяется так, чтобы совпадать с областью действия некоторого окружения}.

b) Каждое окружение имеет одну определенную „область действия". {Область действия каждого окружения „младше" (2.1.2.f) области действия того окружения, из которого оно составлено (2.1.1.1.c).}

{Не следует смешивать область действия окружения с областями действия значений, доступных внутри его участка. Область действия окружения предпочтительно используют при определении области действия сцен, для которых оно необходимо (7.2.2.c), или области действия выдачи генераторов, для которых оно является „локализующим" (5.2.3.2.b). Область действия окружения определяют относительно (2.1.2.f) области действия некоторого другого окружения, так что создаются иерархии областей действия, в конечном счете зависящие от области действия первичного окружения (2.2.2.a).}

2.1.2. *Соотношения*

a) Соотношения либо „постоянны", т.е. не зависят от данной программы и ее исполнения, либо под влиянием некоторых действий могут становиться „справедливыми" или „несправедливыми". Кроме того, соотношения могут быть „транзитивными" , т.е. если „*" – такое соотношение и A*B и B*C оба справедливы, то справедливо и A*C.

b) „Быть выдачей" есть соотношение между значением и действием, а именно исполнением сцены. Данное соотношение становится справедливым по завершении этого исполнения {2.1.4.1.b}.

c) „Иметь доступ" есть соотношение между 'ПАРОЙ' {4.8.1.E} и значением или сценой V; оно может быть справедливым „внутри" определенного участка L {, в '?ОПИСАНИЯ ?МЕТКИ' которого заложена эта ПАРА'}. Данное соотношение становится справедливым, когда эта 'ПАРА' „получает доступ" к V внутри L {3.5.2. Шаг 4, 4.8.2.a}, и тогда оно будет справедливым внутри L между любой 'ПАРОЙ1', эквивалентной {2.1.1.2.a} этой 'ПАРЕ', и V.

d) Постоянные соотношения между значениями: „быть того же вида, что и" {2.1.1.2.c}, „быть меньше", „быть обобщаемым до", „быть удлиняемым до" {2.1.3.1.e} и „быть эквивалентным" {2.1.3.1.g}. Если одно из них вообще определено для данной пары значений, то оно справедливо или несправедливо постоянно. Все эти соотношения транзитивны.

e) „Именовать" есть соотношение между „именем" {2.1.3.2.a} N и каким-нибудь другим значением. Данное соотношение становится справедливым, когда N „начинает именовать" это значение, и перестает быть справедливым, когда N начинает именовать другое значение.

f) Существуют три транзитивных соотношения между областями действия, а именно область действия A {2.1.1.3} может быть либо „младше", либо „такая же, как и", либо „старше" области действия B. Если A младше B, то B старше A, и обратно. Если A такая же, как и B, то A не старше и не младше B {; обратное не обязательно справедливо, так как для некоторых пар областей действия соответствующее соотношение может быть не определено вообще}.

g) „Быть подыменем" есть соотношение между именем и „составным именем" {2.1.3.2.b}. Данное соотношение становится справедливым, когда это составное имя „снабжается подыменем" {2.1.3.3.e, 2.1.3.4.g} или „генерируется" {2.1.3.4.j, 1}; оно остается справедливым до тех пор, пока составное имя не будет снабжено другим множеством подымен.

2.1.3. *Значения*

2.1.3.1. Простые значения.

a) Всякое простое значение является либо „арифметическим значением", т.е. „целым числом" или „вещественным числом", либо „истинностным значением" f , либо „литерой" g , либо „пустым значением" h .

b) Всякое арифметическое значение имеет „размер", т.е. целое число, характеризующее степень различия, с которой это значение сохраняется в вычислителе.

c) *Видом целого или вещественного числа размера n является некоторое* '?РАЗМЕРНОЕ целое' или '?РАЗМЕРНОЕ вещественное' соответственно; при этом если n положительно (равно нулю, отрицательно), то это '?РАЗМЕРНОЕ' есть повторенное n раз 'длинное' (пусто, повторенное – n раз 'короткое').

d) Число различаемых целых или вещественных чисел данного размера увеличивается (уменьшается) вместе с ним вплоть до достижения определенного размера, а именно до „числа добавочных удлинений" (взятого с обратным знаком „числа добавочных укорочений") целых или веществен-

ных чисел соответственно {10.2.1.a, b, d, e}, после чего оно остается постоянным.

e) Для объяснения смысла приведения, называемого обобщением, и описанных в стандартном-вступлении обозначений-операций предполагают, что арифметические значения обладают следующими свойствами:

- для каждой пары целых чисел или вещественных чисел одинакового размера определено соотношение „быть меньше" в его обычном математическом смысле {10.2.3.3.a, 10.2.3.4.a};
- для каждой пары целых чисел одинакового размера может существовать отличимое от них третье число того же размера: первое целое число „минус" второе {10.2.3.3.g};
- для каждой пары вещественных чисел одинакового размера могут существовать три различимых вещественных числа того же размера: первое вещественное число „минус" („умноженное на", „деленное на") второе число {10.2.3.4.g, l, m};
- данные термины „минус", „умножить на" и „разделить на" имеют их обычный математический смысл с той лишь разницей, что в случае вещественных чисел результаты таких действий получаются „в смысле *численного анализа*", т.е. *действия осуществляются над числами*, слегка отклоняющимися от заданных {; это отклонение оставлено в данном стандарте не определенным};
- каждое целое число данного размера „обобщаемо" до некоторого близкого к нему вещественного числа того же размера {6.5};
- каждое целое (вещественное) число данного размера может быть „удлинено до" некоторого близкого к нему целого (вещественного) числа, имеющего на единицу больший размер {10.2.3.3.q, 10.2.3.4.n}.

f) Всякое „истинностное значение" есть или „ИСТИНА", или „ЛОЖЬ". Его видом является 'логическое'.

g) Каждая „литера" „эквивалентна" некоторому неотрицательному целому числу размера нуль – ее „целочисленному эквиваленту" {10.2.1.n}; это соотношение определено только в той степени, что разные литеры имеют разные целочисленные эквиваленты и существует „наибольший целочисленный эквивалент" {10.2.1.p} . Видом литеры является 'литерное'.

h) Единственное „пустое значение" есть „пустое". Его видом является 'пустое значение'.

{Исполнение конструкта выдает пустое значение, когда нет надобности в более полезном результате. Поскольку синтаксисом не предусмотрены переменные-вида-пустое-значение, описания-тождеств-для-пустого-значения, формальные-параметры-вида-пустое-значение, программист не может воспользоваться пустыми значениями, за исключением тех, которые появляются при объединении (6.4).}

i) Областью действия всякого простого значения является область действия первичного окружения {2.2.2.a}.

2.1.3.2. Имена.

a) Всякое „имя" есть значение, которое может или „начать именовать"

{d,5.2.3.2.a, 5.2.1.2.b} какое-нибудь другое значение, или быть „псевдоименем" {и не именовать тогда никакого значения}; более того, для каждого вида, начинающегося с 'имя', существует в точности одно имя этого вида, которое является псевдоименем.

Имя может быть „вновь созданным" {при исполнении генератора (5.2.3.2) или ФОРМЫ-после-векторизации (6.6.2), когда некоторое составное имя снабжается подымснами (2.1.3.3.e, 2.1.3.4.g) и, возможно, когда имя „генерируется" (2.1.3.4.j, l)}. Созданное таким образом имя отличается от всех остальных уже существующих имен.

{Имя можно рассматривать как адрес ячейки или ячеек памяти в вычислителе, используемых, чтобы содержать именуемое значение. Создание имени предполагает отведение в памяти места для этого значения.}

b) Видом всякого имени N является некоторое 'имя ВИДА' и любое именуемое N значение должно быть „приемлемо для" {2.1.3.6.d} этого 'ВИДА'. Если 'ВИД' есть какое-то 'СОСТАВНОЕ', то N называется „составным именем".

c) Область действия всякого имени – это область действия некоторого конкретного окружения {, которым обычно будет „локализующее окружение" (5.2.3.2.b) некоторого генератора}. Область действия имени, являющегося псевдоименем, есть область действия первичного окружения {2.2.2.a}.

d) Если N – составное имя, именующее структуру (массив) V {2.1.3.3, 2.1.3.4}, и какое-то его подымя {2.1.2.g}, выбираемое {2.1.3.3.e, 2.1.3.4.g} по 'СЛОВУ' (индексу) l, начинает именовать {новое} значение X, то N начинает именовать структуру (массив), отличающуюся от V только своим полем (элементом), выбираемым по l, которое {стало теперь} этим X.

{Относительно вида подымени см. 2.1.3.3.d и 2.1.3.4.f}

2.1.3.3. Структуры.

a) Всякая „структура" составлена из последовательности других значений – ее „полей", каждое из которых „выбирается" {b} по определенному 'СЛОВУ' {9.4.2.1.A} {О выборке поля по указателю-поля см. 2.1.5.g.}

{Упорядоченность полей структуры используется в семантике записей-структуры (3.3.2.b), текстов-формата (10.3.4) и в выстраивании (10.3.2.3.c).}

b) Видом структуры V является некоторая 'структура содержащая !ПОЛЯ в себе'. Если заложенное в эти '!ПОЛЯ' n-е 'ПОЛЕ' есть некоторое 'СЛОВО для выборки ВИДА', то это n-е поле значения V „выбирается" по данному 'СЛОВУ' и приемлемо для {2.1.3.6.d} данного 'ВИДА'.

c) Область действия всякой структуры – это самая младшая из областей действия ее полей.

d) Если вид имени N {, именующего какую-то структуру,} есть некоторое 'имя структуры содержащей !ПОЛЯ в себе' и выполняется предикат 'если СЛОВО для выборки ВИДА находится в !ПОЛЯХ' {7.2.1.b, c}, то ви-

дом выбираемого {e} по этому 'СЛОВУ' подымени данного N является 'имя ВИДА'.

e) Когда имя N, именующее структуру V, „снабжается подыменами" {e, 2.1.3.4.g, 4.4.2.b, 5.2.3.2.a}, тогда для каждого 'СЛОВА', выбирающего поле F в V:

- создается новое подымя M с такой же, как и у N, областью действия;
- M начинает именовать поле F;
- M называется именем, „выбираемым" по этому 'СЛОВУ' в N;
- если M – составное имя {2.1.3.2.b}, то оно само снабжается подыменами { e, 2.1.3.4.g}.

2.1.3.4. Массивы.

a) Всякий {n-мерный} „массив" составлен из „паспорта" и последовательности других значений, его „элементов", каждый из которых можно „выбрать" по определенному набору из n целых чисел, называемому „индексом" этого элемента.

b) „Паспорт" имеет форму

$$((l_1, u_1), (l_2, u_2), ..., (l_n, u_n)),$$

где каждая пара (l_i, u_i), $i = 1, ..., n$, является целочисленной „граничной парой", в которой l_i есть i-я „нижняя граница", а u_i есть i-я „верхняя граница".

c) Если $u_i < l_i$ для некоторого i, $i = 1, ..., n$, то паспорт называется „вырожденным" и существует единственный элемент массива, называемый „скрытым элементом" {и не выбираемый ни по какому индексу; см. также 5.2.1.2.b}; в противном случае число элементов равно

$(u_1 - l_1 + 1) \times (u_2 - l_2 + 1) \times ... \times (u_n - l_n + 1)$ и каждый из них выбирается по определенному индексу $(r_1, ..., r_n)$, где $l_i \leqslant r_i \leqslant u_i$, $i = 1, ..., n$.

d) Вид всякого массива V – это 'МАССИВ из ВИДА', где 'МАССИВ' содержит 'вектор' столько раз, сколько граничных пар в паспорте этого V, и каждый элемент значения V приемлем для {2.1.3.6.d} 'ВИДА'.

{ Например, если [ ] об (цел, вещ) ruir = (1; 2,0), то вид выдачи ruir есть 'вектор из объединения целого вещественного воедино', вид ее первого элемента есть 'целое', а вид второго – 'вещественное'.}

e) Областью действия массива, если его паспорт не вырожден, является самая младшая из областей действия его элементов, а иначе – область действия первичного окружения {2.2.2.a}.

f) Массив вида 'МАССИВ из ВИДА' может именоваться либо „именем подвижного" с видом 'имя подвижного МАССИВА из ВИДА1', либо „именем фиксированного" с видом 'имя МАССИВА из ВИДА1', где {в обоих случаях} ‚ВИД' „фиксирует" {2.1.3.6.b} 'ВИД1'.

{Это различие предполагает возможную разницу в способе, которым данное значение хранится в вычислителе. В случае подвижного должно быть позволено присваивать {5.2.1.2.b} массивы с различными границами одному и тому же имени, в то время как в случае фиксированного можно быть уверенным, что эти границы будут оставаться фиксированными с течение

всего времени существования данного имени. Отметим, что „подвижность" есть свойство имени: указываемое значение является в обоих случаях одним и тем же массивом.}

Если видом имени N {, именующего массив,} является некоторое 'имя ?ПОДВИЖНОГО МАССИВА из ВИДА', то видом каждого подымени этого N является 'имя ВИДА'.

g) Когда имя N, именующее массив V, „снабжается подыменами" {g, 2.1.3.3.e, 4.4.2.b, 5.2.1.2.b, 5.2.3.2.a}, тогда для каждого индекса, выбирающего элемент E в V;

- создается новое подымя M с такой же, как и у N, областью действия;
- M начинает именовать элемент E;
- M называется именем, „выбираемым" в N по этому индексу;
- если M – составное имя {2.1.3.2.b}, то оно само снабжается подыменами {g, 2.1.3.3.e}

{В дополнение к выбору по индексу элемента (a) или имени (g) можно выбирать значение или генерировать новое имя, именующее такое значение, с помощью отрезка (h, i, j) или 'СЛОВА' (k, l). Как эти отрезки, так и индексы используются при исполнении вырезок (5.3.2.2.).}

h) Всякий „отрезок" – это набор из n элементов, причем каждый элемент есть либо целое число {, соответствующее индексу}, либо тройка (l, u, d) {, соответствующая отрезку или возможной-сдвинутой-нижней-границе}, так что по крайней мере один из элементов отрезка является тройкой {; если все эти элементы – целые числа, то данный n-набор есть индекс (a)}. Каждый из элементов упомянутых троек или является целым числом, или „отсутствует".

{Отрезок (или индекс) выдается при исполнении индексатора (5.3.2.2.b).}

i) Массив W {размерности m}, „выбираемый" по некоторому отрезку T в {n-мерном, $1 \leqslant m \leqslant n$} массиве V, определяется так:

- Пусть T состоит из целых чисел и троек T_i, $i = 1, ..., n$, m из которых фактически являются тройками; пусть j-я тройка есть (l_j, u_j, d_j), $j = 1, ..., m$;
- W составляется из:

  (i) паспорта $((l_1 - d_1, u_1 - d_1), (l_2 - d_2, u_2 - d_2), ..., (l_m - d_m, u_m - d_m))$;

  (ii) элементов массива V, причем выбираемым в W по индексу $(w_1, ..., w_m)$ {$l_j - d_j \leqslant w_j \leqslant u_j - d_j$} элементом, если он существует, будет элемент, выбираемый в V по индексу $(v_1, ..., v_n)$, определяемому следующим образом:

Для $i = 1, ..., n$,

Случай A : T_i – целое число:

$v_i = T_i$;

Случай B : T_i есть j-я тройка (l_j, u_j, d_j) отрезка T:

$v_i = w_j + d_j$.

j) Имя M, „генерируемое" по отрезку T из имени N, именующего массив V, есть имя {фиксированного, – не обязательно вновь созданное}, имеющее ту же область действия, что и N. Имя M именует массив W, выбираемый {i} по T в V. Каждое подымя этого M, выбираемое по индексу l_W, является одним из выбираемых по некоторому индексу l_V {уже существующих} подымен имени N, где каждый индекс l_V определяется по этому T и соответствующему l_W определенным в предыдущем подразделе методом.

k) Массив W, „выбираемый" по 'СЛОВУ' в массиве V {, каждый элемент которого является структурой,} составляется из:

(i) паспорта этого V и

(ii) выбираемых по данному 'СЛОВУ' в элементах этого V полей; причем выбираемым в W по некоторому индексу l элементом, если он существует, будет выбираемое по данному 'СЛОВУ' поле в том элементе массива V, который выбирается по индексу l.

l) Имя M, „генерируемое" по 'СЛОВУ' из имени N, именующего массив V {, каждый элемент которого является структурой}, есть имя {фиксированного – не обязательно вновь созданное –}, имеющее ту же область действия, что и N. Имя M именует массив, выбираемый {k} в V по этому 'СЛОВУ'. Каждое выбираемое по индексу l подымя этого M является {уже существующим} именем, выбираемым {2.1.3.3.e} по данному 'СЛОВУ' в том подымени имени N, которое выбирается {g} по индексу l.

2.1.3.5. Процедуры.

a) Всякая „процедура" есть сцена {2.1.1.1.d}, составленная из текста процедуры {5.4.1.1.a, b} и некоторого окружения {2.1.1.1.c}

{Процедура может быть „вызвана (5.4.3.2.b) и тогда будет исполняться основа ее текста-процедуры.}

b) Видом процедуры, составленной из текста-процедуры-выдающего-ПРОЦЕДУРУ, является 'ПРОЦЕДУРА'.

c) Область действия всякой процедуры – это область действия ее окружения.

2.1.3.6. Приемлемость значений.

a) {Значений, вид которых начинается с 'объединение', не существует, но существуют имена, вид которых начинается с 'имя объединения', например u в об (цел, вещ) u; . В данном случае u, видом которого является 'имя объединения целого вещественного воедино', именует либо значение вида 'целое', либо значение вида 'вещественное'. При помощи сопоставляющего-предложения (3.4.1.q) в любой момент можно проверить, какая из этих ситуаций имеет место.}

Вид 'ЗНАЧЕНИЕ' „объединен из" вида 'ОБЫЧНОЕ', если это 'ЗНАЧЕНИЕ' – некоторое 'объединение ?ОБЫЧНЫХ1 ОБЫЧНОГО ?ОБЫЧНЫХ2 воедино'.

b) Не существует значений, вид которых начинается с 'подвижное', но существуют имена подвижного, вид которых начинается с 'имя подвижного', например a1 в подв [1:n] вещ a1; . В данном случае a1, видом которого

является 'имя подвижного вектора из вещественных', именует массив с видом 'вектор из вещественных' (см. также 2.1.3.4.f). Вообще существуют значения только тех видов, которые получаются „фиксацией". }

Вид 'ЗНАЧЕНИЕ2' „фиксирует" вид 'ЗНАЧЕНИЕ1', если выполняется предикат 'если ЗНАЧЕНИЕ2 фиксирует ЗНАЧЕНИЕ1' {4.7.1.a, b, c}

{В ходе процесса фиксации 'ЗНАЧЕНИЕ2', получается за счет устранения всех 'подвижное', содержащихся в 'ЗНАЧЕНИИ1' на тех местах, на которых они не содержатся ни в каком 'ИМЕНИ ЗНАЧЕНИЯ3'. Так, например,

'структура содержащая букву а для выборки подвижного вектора из литерных в себе';

не являющаяся видом никакого значения, фиксируется

'структурой содержащей букву а для выборки вектора из литерных в себе';

являющейся поэтому видом значения, которое может именоваться подвижным именем, имеющим вид

'имя структуры содержащей букву а для выборки подвижного вектора из литерных в себе'.

Этот последний вид уже есть вид имени и, следовательно, его нельзя подвергнуть фиксации.}

c) {Не существует имен, вид которых начинается с 'временное имя',

„Временное имя" вида 'имя ВИДА' – это выдача некоторой ФОРМЫ-выдающей-временное-имя-ВИДА, но так как в настоящем языке нет описателей-временного-имени-ВИДА (4.6.1), то синтаксисом гарантируется, что никакое временное имя никогда не будет присвоено, приписано или выработано в результате вызова процедуры.

Например, xx : =a1 [i] не есть присваивание потому, что xx не есть идентификатор-выдающий-имя-временного-имени-вещественного. Временные имена возникают в вырезках, выборках из массива или при векторизации имен подвижного.}

d) Значение вида M1 „приемлемо для" вида M2, если

(i) M1 такой же, как и M2, или

(ii) M2 объединен из {a} M1 {например, для специфицированного описателем об (вещ, цел) вида приемлемы значения, вид которых специфицируется либо вещ, либо цел}, или

(iii) M1 фиксирует {b} M2 {, например, для вида 'подвижный вектор из вещественных' (, для которого не существует значений,) приемлемы такие значения, как выдача фактического-описателя подв [1 : n] вещ, являющаяся значением вида 'вектор из вещественных'},

или

(iv) M1 – 'имя ВИДА', а M2 – 'временное имя ВИДА' {например, для вида 'временное имя вещественного' приемлемы значения (как выдача вырезки a1 [i]), вид которых есть 'имя вещественного'}.

{См. 2.1.4.1.b в связи с приемлемостью выдачи сцены.}

2.1.4. *Действия*

2.1.4.1. Исполнение.

a) „Исполнение" определенных сцен {, а именно тех, чьи конструкты обозначаются определенными парапонятиями,} описывается в разделах настоящего стандарта, озаглавленных „Семантика". В них задается последовательность „действий", которые надо провести в ходе исполнения каждой такой сцены.

{Примеры действий, которые могут описываться:

- сделать справедливыми какие-то соотношения,
- создать новые имена и
- исполнить другие сцены.}

„Смысл" сцены – это эффект действий, проводимых в ходе ее исполнения. Любое из этих действий или любую их комбинацию можно заменить любым действием или комбинацией, вызывающими тот же эффект.

b) Исполнение сцены S может „выдавать" значение. Если конструктом из S является ПОНЯТИЕ-выдающее-ЗНАЧЕНИЕ, то выдаваемое значение, если не оговорено противное, {имеет такой вид, что оно} приемлемо для {2.1.3.6.d} этого 'ЗНАЧЕНИЯ'.

{Это правило позволяет обсуждать выдачи в семантике без явного упоминания их вида.}

c) Если исполнение некоторого конструкта A в некотором окружении E не описано явным способом, отличным от сказанного в этом пункте, и B – единственный прямой наследник A, требующий исполнения {см. ниже}, то исполнение A в E состоит в исполнении B в E, а выдача A, если таковая полагается, есть выдача этого B, если она существует {; такое автоматическое исполнение называется „предыполнением" A в E}.

Конструкт не требует исполнения, если он невидим {1.1.3.2.h}, является символом {9.1.1.h} или его исполнение не описано никаким способом в настоящем стандарте и ни один из его прямых наследников не требует исполнения.

{Например, исполнением замкнутого-предложения-выдающего-имя-вещественного (3.1.1.a) (x:=3.14) является (, и выдает то же значение, что и) исполнение его составляющего последовательного-предложения-выдающего-имя-вещественного (3.2.1.a) x:=3.14}

2.1.4.2. Последовательные и совместные действия.

a) Всякое действие может быть „неделимым", „последовательным' или „совместным". Всякое последовательное или совместное действие состоит из одного или нескольких других действия, называемых его „поддействиями". Неделимое действие не состоит из других действия {; какие действия неделимы, оставлено настоящим стандартом не определенным}.

b) „Наследным действием" другого действия B является поддействие либо B, либо наследного действия B.

c) Действие A есть „наддействие" действия B, если B – поддействие {a} для A.

d) Поддействия всякого последовательного действия S выполняются

одно за другим; т.е. за завершением {2.1.4.3.c, d}поддействия для S идет запуск {2.1.4.3.b, c} следующего поддействия этого S, если оно есть. {Исполнение сцены, состоящее в общем случае из последовательности действий, будет последовательным действием.}

e) Поддействия совместного действия совмещаются во времени; точнее, выбирается и доводится до конца одно из тех его наследных неделимых действий, которое в данный момент „активно" {2.1.4.3.a}; и по его завершении {2.1.4.3.c} выбирается другое такое действие и т.д. {вплоть до завершения их всех}.

Способ такого выбора оставлен настоящим стандартом не определенным, за тем лишь исключением, что если два действия {, совместные между собой,} названы „несовместимыми" {10.2.4} друг с другом, то {они не должны совмещаться, т.е} ни одно из наследных неделимых действий одного из них (одно из них {, если оно само неделимо,}) не должно выбираться, если другое {действие} активно в данный момент и одно или несколько, но не все из его наследных неделимых действий уже завершены.

f) Если одна или несколько сцен должны быть „исполнены совместно", то это исполнение есть совместное действие, состоящее из {совместного} исполнения данных сцен.

2.1.4.3. Запуск, завершение и прекращение.

a) Всякое действие либо „активно", либо „неактивно". Действие становится активным, когда оно „запускается" {b, c} или „возобновляется" {g}; действие становится неактивным, когда оно „завершается" {c, b}, „прекращается" {e}, „приостанавливается" {f} или „прерывается" {h}.

b) Когда последовательное действие „запускается", запускается его первое поддействие. Когда „запускается" совместное действие, запускаются все его поддействия.

c) Когда „запускается" неделимое действия, то оно может быть доведено до конца {см. 2.1.4.2.e}, после чего оно становится „завершенным".

d) Всякое последовательное действие „завершается", когда завершается его последнее поддействие. Всякое совместное действие „завершается", когда завершены все его поддействия.

e) Когда {последовательное или совместное} действие „прекращается", прекращаются все его поддействия {и, следовательно, все его наследные действия;} {после чего вместо него может запуститься другое действие . Прекращение действия вызывается исполнением перехода (5.4.4.2).}

f) Когда действие „приостанавливается" приостанавливаются все его поддействия {и, следовательно, все его активные наследные действия} {Действие может приостановиться в течение „вызова" процедуры, выдаваемой обозначением-операции вниз (10.2.4.d), после чего оно может возобновиться впоследствии, во время вызова процедуры, выдаваемой обозначением-операции вверх (10.2.4.e).}

Если в какой-то момент приостанавливается некоторое действие, не являющееся наследным действием „процесса" „параллельного действия'

{10.2.4}, у другого процесса (других процессов) которого продолжают существовать активные наследные неделимые действия, то дальнейшее исполнение не определено.

g) Когда действие А „возобновляется", возобновляются его поддействия, приостановленные вследствие приостановки самого А.

h) Всякое действие может „прерваться" событием, {например, „переполнением",} не определяемым семантикой настоящего стандарта, но вызванным вычислителем, если его возможности {2.2.2.b} не позволяют обеспечить удовлетворительное исполнение. Когда действие прерывается, прерываются все его поддействия и, возможно, его наддействия. {Возобновятся ли эти действия после прерывания, будут ли запущены другие действия или же окончится исполнение данной программы, настоящим стандартом оставлено не определенным.}

{Результат данных выше определений следующий:

В ходе исполнения программы (2.2.2.a) исполнение ее замкнутого-предложения в пустом первичном окружении активно. В любой данный момент исполнение одной сцены может вызвать исполнение другой сцены или совместное исполнение нескольких других сцен. В том случае, когда исполнение этой другой сцены или сцен завершится, предпринимается следующий шаг по исполнению первоначальной сцены и т.д., до тех пор пока он в свою очередь не завершится.

Можно видеть, что все это аналогично обращению одной подпрограммы к другой – выполнение вызвавшей подпрограммы продолжается по завершении выполнения вызванной подпрограммы. Данные в настоящем стандарте семантические правила для исполнения различных парапонятий соответствуют текстам таких подпрограмм; эти семантические правила могут даже в подходящих обстоятельствах рекурсивно вызывать сами себя (, но в каждом таком случае с различными конструктами или в различных окружениях.)

Таким образом, в каждый момент существует дерево активных действий, наследных (2.1.4.2.b) для исполнения данной программы.}

2.1.5. *Сокращения*

{На всем протяжении текста настоящего стандарта свободно используются некоторые сокращения, позволяющие избежать некоторых длинных и запутанных фраз, необходимых в противном случае в семантике.}

a) „А (этого) В" или „А из В", где А и В – парапонятия, заменяет фразу „А, которое есть прямой наследник {1.1.3.2.f} В".

{Это позволяет сокращать „прямой наследник (конструкта)" до „его", „из", „этого" или же просто употреблять форму родительного падежа; например, i в присваивании (5.2.1.1.a) i:=1 есть „его" получатель (, или i-получатель „этого" присваивания, или даже i-получатель присваивания i: = 1); в то время как i не есть получатель последовательного-предложения i := 1; j: = 2(, хотя он – составляющий получатель (1.1.4.2.d) этого предложения).}

b) „С в Е", где С – конструкт, а Е – окружение, заменяет фразу „сце-

на, составленная {2.1.1.1.d} из С и Е". Иногда она сокращается просто до „С", если ясно, о каком окружении идет речь.

{Так как процесс исполнения (2.1.4.1.a) может применяться только к сценам, это сокращение чаще всего встречается в таких формах, как „Циклическое-предложение С в окружении Е1 исполняется..." (3.5.2) или „Присваивание А исполняется..." (5.2.1.2.a, где говорится об исполнении А в любом подходящем окружении).}

c) „Выдача (этого) S", где S – сцена, исполнение которой не предписано явно, заменяет фразу „выдача, получаемая в результате запуска исполнения сцены S, если дожидаться завершения этого исполнения".

{Таким образом, высказывание (3.2.2.c) „W есть выдача этой основы;", (использующее также сокращение, определенное в b выше,) надо интерпретировать как означающее

„Исполняется сцена, составленная из данной основы и того окружения, о котором идет речь; W является выдачей, получаемой по завершении исполнения этой сцены;".}

d) „Выдачи S_1, ..., S_n", где S_1, ..., S_n – сцены, исполнение которых не предписано явно, заменяет фразу „выдачи, получаемые в результате запуска совместного исполнения {2.1.4.2.f} сцен S_1, ..., S_n, если дожидаться завершения этого исполнения {, что предполагает завершение исполнения всех сцен}".

Если некоторые или все сцены S_1, ..., S_n описаны как определенные составляющие какого-то конструкта, взятые в некотором окружении, то их выдачи должны рассматриваться расположенными в текстуальном порядке {1.1.3.2.i} этих составляющих в данном конструкте.

{Таким образом, высказывание (3.3.2.b)

„пусть V_1, ..., V_m будут {совместными} выдачами составляющих основ этого С;"

должно интерпретироваться как означающее

„пусть V_1, ..., V_m будут соответствующими выдачами, получаемыми в результате запуска и последующего завершения совместного исполнения сцен, состоящих из составляющих основ этого С, взятых в их текстуальном порядке, и того окружения, в котором исполняется С;".}

e) „Если А есть (является) В", где А и В – гиперпонятия, заменяет фразу „если А эквивалентно {2.1.1.2.a} В".

{Таким образом, в высказывании „Случай С: 'ВЫБИРАЮЩЕЕ' есть некоторое 'выбирающее по ПРЕДСТАВИТЕЛЮ'" (3.4.2.b) несущественно, будет ли это 'ВЫБИРАЮЩЕЕ' начинаться с 'выбирающее по объединению' или же с 'выбирающее по ЦИ определению объединения'.}

f) „Вид есть (является) А", где А – гиперпонятие, заменяет фразу „вид {, являющийся классом протопонятий 'ЗНАЧЕНИЕ'}, который включает это А".

{Это позволяет употреблять такие сокращенные формы, как „вид есть некоторая 'структура содержащая !ПОЛЯ в себе'", „данный вид начинается

с 'объединение'" или „вид, в который заложено 'ПОЛЕ',, в общем случае вид можно указать, приведя только одно входящее в него 'ЗНАЧЕНИЕ'.}

g) „Значение, выбираемое (генерируемое) по данному указателю-поля F," заменяет фразу „если F – {ПРИМЕНЯЮЩИЙ-} СЛОВО-указатель-поля {4.8.1.f}, то данное значение выбирается {2.1.3.3.a, e 2.1.3.4.k} (генерируется {2.1.3.4.l}) по этому 'СЛОВУ'".

2.2. Программа

2.2.1. *Синтаксис*

a) программа: замкнутое предложение в новом {пустом окружении} сильно выдающее пустое значение {31 a}

{См. также 10.1}

2.2.2. *Семантика*

a) Исполнение программы есть исполнение ее замкнутого-предложения-в-новом-сильно-выдающего-пустое-значение в пустом окружении {2.1.1.1.c}, называемом „первичным окружением".

{Несмотря на то, что цель настоящего стандарта – определять смысл собственно-программ (10.1.1.g), это смысл устанавливается только через предваряющее определение смысла программы, в которую эта собственно-программа вложена (10.1.2).}

{В настоящем стандарте синтаксис определяет, какие последовательности символов являются терминальными порождениями понятия 'программа', а семантика – какие действия осуществляются вычислителем, когда программа исполняется. Как синтаксис, так и семантика рекурсивны. Хотя некоторые последовательности символов могут быть терминальными порождениями 'программы' (см. также 1.1.3.2.f), порождаемыми более чем одним способом, эта синтаксическая неоднозначность не приводит к семантической двусмысленности.}

b) В Алголе 68 предусмотрен специальный синтаксис для конструктов. Вместе с их рекурсивным определением он дает возможность описывать и различать произвольно большие деревья порождения, различать произвольно много значений данного вида (исключая такие виды, как 'логическое и стое значение') и различать как угодно много видов. Этот синтаксис позволяет существовать в вычислителе произвольно большому числу объектов и позволяет исполнению программы включать в себя произвольно большое, не обязательно конечное, число действий. Из этого не следует, что способ записи этих объектов в вычислителе тот же, что и в настоящем стандарте и что он имеет те же возможности. Не предполагается, что эти два способа записи одинаковы и даже что между ними существует взаимно однозначное соответствие, фактически множество разных способов обозначения объектов данной категории может быть конечным. Не предполагается, что вычислитель может обрабатывать произвольные объемы предлагаемой информации, что скорость вычислителя достаточна для исполнения заданной программы за предписанный промежуток времени и что количество объектов и соотношений, которые можно определить в вычислителе, достаточно для ее исполнения вообще.

c) Модель гипотетического вычислителя, использующая реальную вычислительную машину, называется „реализацией" Алгола 68, если она не ограничивает применение настоящего языка в других аспектах, отличных от упомянутых выше. Кроме того, если определяется язык А, собственно-программы которого являются также собственно-программами языка В; и смысл каждой такой собственно-программы, определяемый языком А, совпадает с ее смыслом, определяемым языком В, то А называется „подъязыком" для В, а В называется „надъязыком" для А.

{Так, например, подъязык Алгола 68 можно определить, опуская отдельные правила синтаксиса, обедняя данное стандартное-вступление и/или оставляя не определенным кое-что, определяемое в настоящем стандарте. Тем самым можно будет обеспечить более эффективное решение определенных классов задач или осуществить реализацию для малых машин.

Аналогично надъязык Алгол 68 можно определить за счет некоторых добавлений к синтаксису, семантике или стандартному-вступлению и тогда можно улучшить эффективность (, позволяя пользователю поставлять добавочную информацию,) или дать возможность решать задачи, с трудом поддающиеся Алголу 68.}

Модель называют реализацией подъязыка, если она не ограничивает использование этого подъязыка в иных аспектах, нежели упомянутые выше.

{См. 9.3.c по поводу термина „реализация эталонного языка".}

3. ПРЕДЛОЖЕНИЯ

{Предложения обеспечивают

- иерархическую структуру программ,
- введение новых блоков определений,
- последовательную или совместную композицию, параллелизм, разветвления и циклы.

3.0.1. *Синтаксис*

a) *фраза: основа ДЕЙСТВУЮЩАЯ {32d};
описание !ОПИСАНИЙ в СРЕДЕ {41a}.

b) *выражение ПРИВОДИМО выдающее ВИД:
ОСНОВА {5A} в СРЕДЕ ПРИВОДИМО выдающая ВИД.

c) *оператор:
ОСНОВА {5A} в СРЕДЕ сильно выдающая пустое значение.

d) *константа вида ЗНАЧЕНИЕ : ПРИМЕНЯЮЩИЙ СЛОВО
идентификатор в СРЕДЕ выдающий ЗНАЧЕНИЕ {48a, b};
изображаемое в СРЕДЕ выдающее ЗНАЧЕНИЕ {80a}.

e) *переменная вида ВИД : ПРИМЕНЯЮЩИЙ СЛОВО
идентификатор в СРЕДЕ выдающий имя ВИДА {48a, b}.

f) * блок в СРЕДЕ : определяющее СЛОЙ последовательное
предложение ПРИВОДИМОЕ В СРЕДЕ {32a};
состав ВЫБИРАЮЩЕГО предложения
ОФОРМЛЕННЫЙ ПРИВОДИМЫЙ в СРЕДЕ {34b};
вариант выбирающий по ПРЕДСТАВИТЕЛЮ ПРИВОДИМЫЙ в
СРЕДЕ {34i};
ОФОРМЛЕННЫЙ цикл с ОПИСАНИЕМ в СРЕДЕ {35e};
ОФОРМЛЕННЫЙ подчиненный условию цикл в СРЕДЕ {35f};
текст процедуры в СРЕДЕ выдающий ПРОЦЕДУРУ {541a, b}

{Блоки-в-СРЕДЕ появляются в определении „идентификации" (7.2.2.b).}

3.0.2. *Семантика*

Всякая „среда" есть некоторая 'СРЕДА'. „Средой конструкта" является 'СРЕДА', заложенная в прообраз этого конструкта, но не заложенная ни в какой содержащийся в этом прообразе 'определяющий СЛОЙ'.

{Среда конструкта содержит запись обо всех описаниях, образующих окружение, в котором этот конструкт должен интерпретироваться.

Конструкты, содержащиеся в некотором блоке R, но не содержащиеся ни в каком меньшем блоке, содержащемся в R, можно назвать образующими „зону". Все конструкты в данной зоне имеют одну и ту же среду, а именно среду непосредственно окружающей ее зоны с добавлением одного дополнительного 'СЛОЯ'. Синтаксис гарантирует (3.2.1.b, 3.4.1.i, j, k, 3.5.1.e, 5.4.1.1.b), что каждой 'ПАРЕ' (4.8.1.E), отражающей некоторое „свойство" в этом дополнительном 'СЛОЕ', соответствует определяющий-индикатор (4.8.1.a), содержащийся в каком-то определении в данной зоне.}

3.1. Замкнутые предложения

{Замкнутые-предложения обычно используют, чтобы создавать основы из последовательных-предложений, как, например,

(вещ x; чит (x); x) в
(вещ x; чит (x); x) + 3.14}

3.1.1. *Синтаксис*

A) ПРИВОДИМОЕ :: ПРИВОДИМО выдающее ЗНАЧЕНИЕ.
) УПАКОВКА :: упакованное ОФОРМЛЕННОЕ.
замкнутое предложение в СРЕДЕ
ПРИВОДИМОЕ {22a, 5D, 551a, A341h, A349a}:
УПАКОВКА определяющего СЛОЙ последовательного предложения
ПРИВОДИМОГО в СРЕДЕ {32a}
{СЛОЙ :: новые ?ОПИСАНИЯ ?МЕТКИ.}
{Пример:
а) начало x:=1; y:=2 конец}

{Выдачей замкнутого-предложения является вследствие предысполнения (2.1.4.1.c) выдача его составляющего последовательного предложения.}

3.2. Последовательные предложения

{Назначение последовательных-предложений состоит в:

- построении новых блоков определений и
- последовательной композиции действий.

Всякое последовательное-предложение состоит из возможно пустой последовательности непомеченных фраз, последняя из них, если она есть, является описанием, за которым идет последовательность возможно помеченных основ. Эти фразы и основы разделяются знаками-продолжать, а именно точками с запятой. Некоторые из этих основ могут, однако, разделяться завершителями, а именно выходами; основа, следующая за завершителем, должна быть помечена, чтобы быть достижимой. Значение последней основы или основы, предшествующей выходу, определяет значение всего последовательного-предложения.

Например, следующее последовательное-предложение выдает значение истина тогда и только тогда, когда вектор а содержит целое число 8:

```
цел n; чит (n);
[1 : n] цел a; чит (a);
для i до n цк если a [i] = 8 то на найдено все кц;
ложь выход
найдено : истина}
```

3.2.1. *Синтаксис*

```
a) определяющее новые ?ПАРЫ последовательное предложение
       ПРИВОДИМОЕ В СРЕДЕ {31a, 34f, l, 35h}:
   кортеж с ?ПАРАМИ ПРИВОДИМЫЙ
       в СРЕДЕ с новыми ?ПАРАМИ {b}
   {Здесь ?ПАРЫ :: ?ОПИСАНИЯ ?МЕТКИ.}
b) кортеж с ?ПАРАМИ ПРИВОДИМЫЙ в СРЕДЕ {a, b, 34c}:
   основа в СРЕДЕ сильно выдающая пустое значение {d},
     знак продолжать {94f},
   кортеж с ?ПАРАМИ ПРИВОДИМЫЙ в СРЕДЕ {b};
   если (?ПАРЫ) есть (!ОПИСАНИЯ ?ОПИСАНИЯ ?МЕТКИ),
   описание !ОПИСАНИЙ в СРЕДЕ {41a}, знак продолжать {94f},
   кортеж с !ОПИСАНИЯМИ ?МЕТКАМИ ПРИВОДИМЫЙ в
     СРЕДЕ {b};
   если (?ПАРЫ) есть (МЕТКА ?МЕТКИ),
     определение метки через МЕТКУ в СРЕДЕ {c},
     кортеж с ?МЕТКАМИ ПРИВОДИМЫЙ в СРЕДЕ {b};
   если (?ПАРЫ) есть (МЕТКА ?МЕТКИ) и ПРИВОДИМОЕ
     уравнивает ПРИВОДИМОЕ1 и ПРИВОДИМОЕ2 {e},
   основа в СРЕДЕ ПРИВОДИМАЯ1 {d}, знак завершить {94f},
     определение метки через МЕТКУ в СРЕДЕ {c},
   кортеж с ?МЕТКАМИ ПРИВОДИМЫЙ2 в СРЕДЕ {b};
     если (?ПАРЫ) есть (ПУСТО),
     основа в СРЕДЕ ПРИВОДИМАЯ {d}.
c) определение метки через СЛОВО для метки в СРЕДЕ {b}:
```

определяющий СЛОВО идентификатор в СРЕДЕ
выдающий метку {48a}, знак метка {94f}.

d) основа ДЕЙСТВУЮЩАЯ {b, 33b, g, 34i, 35d, 46m, n, 521c, 532e, 541a, b, 543c, A34Ab, c, d}:
ОСНОВА {5A, –} ДЕЙСТВУЮЩАЯ.

e) ЕСЛИ ПРИВОДИМО выдающее ЗНАЧЕНИЕ уравнивает
ПРИВОДИМО1 выдающее ЗНАЧЕНИЕ1 и
ПРИВОДИМО2 выдающее ЗНАЧЕНИЕ2 {b, 33b, 34d, h} :
ЕСЛИ ПРИВОДИМО уравнивает ПРИВОДИМО1 и
ПРИВОДИМО2 {f} и ЗНАЧЕНИЕ уравнивает
ЗНАЧЕНИЕ1 и ЗНАЧЕНИЕ2 {g}.

f) ЕСЛИ ПРИВОДИМО уравнивает
ПРИВОДИМО1 и ПРИВОДИМО2 {e, 522a}:
если (ПРИВОДИМО1 есть (сильно),
ЕСЛИ (ПРИВОДИМО2) есть (ПРИВОДИМО);
если (ПРИВОДИМО2) есть (сильно),
ЕСЛИ (ПРИВОДИМО1) есть (ПРИВОДИМО).

g) ЕСЛИ ЗНАЧЕНИЕ уравнивает ЗНАЧЕНИЕ1 и ЗНАЧЕНИЕ2 {e}.
если (ЗНАЧЕНИЕ1) есть (ЗНАЧЕНИЕ2),
ЕСЛИ (ЗНАЧЕНИЕ) есть (ЗНАЧЕНИЕ1);
если (ЗНАЧЕНИЕ1) есть (временное ЗНАЧЕНИЕ2).
ЕСЛИ (ЗНАЧЕНИЕ) есть (ЗНАЧЕНИЕ1);
если (ЗНАЧЕНИЕ2) есть (временное ЗНАЧЕНИЕ1).
ЕСЛИ (ЗНАЧЕНИЕ) есть (ЗНАЧЕНИЕ2).

h)* ПРИВОДИМОЕ основное предложение:
основа в СРЕДЕ ПРИВОДИМАЯ {d}.

i)* определяющее предложение:
определяющее СЛОЙ последовательное
предложение ПРИВОДИМОЕ в СРЕДЕ {32a};
определяющее СЛОЙ выясняющее
предложение выдающее ВИД в СРЕДЕ {34c}
{Примеры:
b) чит (x1); вещ s := 0;
суммирование : для i до n цк (x1 [i] > 0 | s +: =x1 [i]
| не положительное) кц выход
не положительное : печ (s) ·
вещ s := 0;
суммирование : для i до n цк (x1 [i] > 0 | s +:= x1 [i]
| не положительное) кц выход
не положительное : печ (s) ·
суммирование : для i до n цк (x1 [i] > 0 | s +:=x1 [i]
| не положительное) кц выход
не положительное : печ (s) ·
для i до n цк (x1 [i] > 0 | s +:=x1 [i] | не положительное) кц выход
не положительное : печ (s) ·

печ (s)

c) суммирование: d) печ (s)}

{Во многих случаях кортежи должны „уравниваться" (3.2.1.e). Замечания относительно уравнивания см. в 3.4.1.}

3.2.2. *Семантика*

a) Выдачей последовательного-предложения в окружении E является выдача исполнения его кортежа или любого кортежа, исполняемого „вместо него" {5.4.4.2}, в окружении, „устанавливаемом" {b} вокруг E согласно этому последовательному-предложению; требуется, чтобы по области действия эта выдача не была младше данного E.

b) Окружение E, „устанавливаемое"

• по окружению E1, возможно не обусловленному, {которое определяет его область действия,}

• вокруг окружения E2, {определяющего его состав,}

• согласно определяющему-новые-?ПАРЫ-ПОНЯТИЮ C, возможно отсутствующему, {которое задает его участок,}

• со значениями V_1, ..., V_n, возможно отсутствующими, {которые возможно будут приписаны,}

определяется следующим образом:

• если E1 не обусловлено, то пусть E1 будет E2;

• E младше E1 по области действия и составлено из E2 и нового участка, соответствующего '?ПАРАМ', если C присутствует, а иначе соответствующего 'ПУСТО';

Случай A: C есть определяющее-предложение:

Для каждого составляющего определения-вида M этого C, если они вообще есть.

• сцена, составленная из

(i) фактического-описателя этого M и

(ii) окружения, необходимого для {7.2.2.c} этого фактического-описателя в E, приписывается индикатору-вида этого M в E;

Для каждого составляющего определения-метки L этого C, если они вообще есть,

• сцена, составленная из

(i) кортежа, для которого L – прямой наследник, и

(ii) окружения E,

приписывается идентификатору-метки этого L в E;

Если каждая 'ПАРА', заложенная в '?ПАРЫ', есть 'ИНФИКС для БИНАРНОГО' или 'СЛОВО для метки', то E называется „нелокализующим" {см. 5.2.3.2.b};

Случай B: C есть задание-аргументов, заглавие-цикла или спецификация:

Для i = 1, ..., n, где n – число 'ОПИСАНИЙ', заложенных в '?ПАРЫ',

• V_i приписывается {4.8.2.a} i-му составляющему определяющему-идентификатору этого C в E, если они вообще есть, а иначе {в случае невидимого заглавия-цикла} некоторому определяющему-буквуалеф-идентификатору-выдающему-целое;

Если С служит заглавием-цикла или спецификацией, то Е является нелокализующим.

{В остальных случаях, т.е. когда С отсутствует:

- Е является локализующим (см. 5.2.3.2.b), но дальнейшее не определено.}

c) Выдача W всякого кортежа С определяется следующим образом:

Если С содержит прямую наследную основу, за которой нет знака-продолжать,

то

- W – выдача этой основы;

а иначе

- исполняется описание или основа этого С, если они вообще есть;
- W – выдача кортежа этого С.

{См. также 5.4.4.2. Случай А.}

3.3. Совместные и параллельные предложения

{Совместные-предложения допускают произвольное совмещение потоков действий. Параллельные-предложения обеспечивают, кроме того, уровни координации для синхронизации (10.2.4) этого совмещения.

Всякое совместное- или параллельное-предложение состоит из последовательности основ, разделенных символами-а-также (, а именно „ , ”), и заключено в скобки или пару начало-конец; кроме того, всякое параллельное-предложение начинается с пар.

Совместные-предложения, но не параллельные-предложения, могут выдавать составные значения, составленные из выдач их составляющих основ.

Примеры совместных-предложений, выдающих составные значения:

```
[ ] цел q = (1, 4, 9, 16, 25);
    ст (цел цена, строк класс) велосипед : = (150, „спортивный”).
```

Пример параллельного-предложения, синхронизирующего еду и разговоры за едой:

```
проц пуст есть, говорить; сема рот = уст 1;
пар начало
  цк вниз рот; есть; вверх рот
  кц,
  цк вниз рот; говорить; вверх рот
  кц
конец.}
```

3.3.1. *Синтаксис*

a) совместное предложение в СРЕДЕ
сильно выдающее пустое значение {5D, 551a};
УПАКОВКА образа основ
сильно выдающих пустое значение в СРЕДЕ {b}.

b) образ основ ПРИВОДИМЫХ в СРЕДЕ {a, b, c, d, 34g}:
если ПРИВОДИМОЕ уравнивает
ПРИВОДИМОЕ1 и ПРИВОДИМОЕ2 {32e},
основа в СРЕДЕ ПРИВОДИМАЯ1 {32d},

знак а также {94f},
основа в СРЕДЕ ПРИВОДИМАЯ2 {32d} либо
образ основ ПРИВОДИМЫХ2 в СРЕДЕ {b}.

c) параллельное предложение в СРЕДЕ
сильно выдающее пустое значение {5D, 551a}:
знак параллельно {94f},
УПАКОВКА образа основ
сильно выдающих пустое значение в СРЕДЕ {b}.

d) совместное предложение в СРЕДЕ
сильно выдающее МАССИВ из ВИДА {5D, 551a};
если (МАССИВ) есть (вектор),
УПАКОВКА образа основ
сильно выдающих ВИД в СРЕДЕ {b};
если (МАССИВ) есть (вектор МАССИВОВ1),
УПАКОВКА образа основ
сильно выдающих МАССИВ1 из ВИДА в СРЕДЕ {b};
УПАКОВКА ПУСТО.

e) совместное предложение в СРЕДЕ
сильно выдающее структуру
содержащую !ПОЛЯ ПОЛЕ в себе {5D, 551a}:
УПАКОВКА образа !ПОЛЕЙ ПОЛЯ в СРЕДЕ {f}.

f) образ !ПОЛЕЙ ПОЛЯ в СРЕДЕ {e, f}:
образ !ПОЛЕЙ в СРЕДЕ {f, g}, знак а также {94f},
образ ПОЛЯ в СРЕДЕ {g}.
{ПОЛЕ :: СЛОВО для выборки ВИДА.}

g) образ СЛОВА для выборки ВИДА в СРЕДЕ {f}:
основа в СРЕДЕ сильно выдающая ВИД {32d}.

h)* запись структуры : совместное предложение
в СРЕДЕ сильно выдающее структуру содержащую
!ПОЛЯ ПОЛЕ в себе {e}.

i)* запись массива : совместное предложение
в СРЕДЕ сильно выдающее МАССИВ из ВИДА {d}.

j)* запись составного : совместное предложение
в СРЕДЕ сильно выдающее СОСТАВНОЕ {d, e}.

к)* вакуум : УПАКОВКА ПУСТО.

{Примеры:

a) (x := 1, y := 2) b) x := 1, y := 2
c) пар (задача1, задача2) d) (1, 2) (в [ ] вещ (1, 2))
e) (1, 2) (в компл (1, 2)) f) 1, 2 g) 1 }

{Записи-структур должны содержать по крайней мере два образа-ПОЛЯ.

Записи-массивов содержат нуль, две или более составляющих основ. Одно значение тоже можно сделать массивом, например [1 : 1] цел v := 123, но при этом используется приведение, называемое векторизацией (6.6.).}

3.3.2. *Семантика*

a) Исполнение совместного-предложения-выдающего-пустое-значение или параллельного-предложения-выдающего-пустое-значение состоит в совместном исполнении его составляющих основ и выдает пустое.

b) Выдача W совместного-предложения-выдающего-СОСТАВНОЕ С определяется следующим образом:

Если прямой наследник из С есть вакуум,

то

{'СОСТАВНОЕ' есть 'МАССИВ ИЗ ВИДА',} каждая граничная пара в паспорте выдачи W равна (1,0) {, и имеется один скрытый элемент, значение которого не существенно};

иначе

• пусть V_1, ..., V_m будут {совместными} выдачами составляющих основ из С;

Случай А: 'СОСТАВНОЕ' есть 'структура содержащая !ПОЛЯ в себе'

• V_1, ..., V_m, взятые в их порядке, служат полями W;

Случай В: 'СОСТАВНОЕ' есть 'вектор из ВИДА1':

• W состоит из

(i) паспорта ((1, m)),

(ii) V_1, ..., V_m;

Для i = 1, ..., m

• V_i – элемент, выбираемый по индексу (i) в W;

Случай С: 'СОСТАВНОЕ есть вектор МАССИВОВ из ВИДА2'

• требуется, чтобы паспорта значений V_1,...,V_m были идентичны;

• пусть паспортом {, например,} V_1 будет $((l_1, u_1),..., (l_n, u_n))$.

• W состоит из

(i) паспорта $((1, m), (l_1, u_1), ..., (l_n, u_n))$;

(ii) элементов этих V_1, ..., V_m;

Для i = 1, ..., m,

• элементом, выбираемым по индексу $(i, i_1, ..., i_n)$ в W, будет элемент, выбираемый по $(i_1, ..., i_n)$ в V_1.

{Отметим, что в [„] лит группа = („абв", „где") паспортом трехмерной выдачи W будет ((1, 2), (1, 1), (1, 3)), поскольку основы „абв" и „где" сначала векторизуются (6.6.), так что V_1 и V_2 имеют паспорта ((1, 1), (1, 3)).}

3.4 Выбирающие предложения

{Выбирающие-предложения позволяют осуществлять динамический выбор среди различных путей вычисления. Выбор среди альтернатив (главной-части-ВЫБИРАЮЩЕГО-предложения и продолжения-ВЫБИРАЮЩЕГО-предложения) определяется результатом проверки некоторого истинностного значения, целого числа или вида. Подвергаемое проверке значение вычисляется выясняющим-предложением прежде, чем делается выбор.

Всякое выбирающее-по-логическому-предложение (или условное-предложение) имеет форму

(x > 0 | x| 0) в „кратком" оформлении, или

если x > 0 то x иначе 0 все в оформлении „стиля1":

x > 0 – его выясняющее-предложение, то x – главная-часть-ВЫБИРАЮЩЕГО-предложения, иначе 0 – продолжение-ВЫБИРАЮЩЕГО-предложения: так как выбирающие-предложения полностью замкнуты, то каждая из этих трех составляющих может иметь синтаксическую структуру кортежа. Выбирающее-по-логическому-предложению можно также укоротить до

(x < 0 | x := – x) или

если x < 0 то x := – x все;

где опущенное продолжение-ВЫБИРАЮЩЕГО-предложения должно пониматься как иначе пропуск. С другой стороны, выбор можно повторить, написав:

(x > 0 | 1 + x | x < 0 | 1 – x | 1) или

если x > 0 то 1 + x инес x < 0 то 1 - x иначе 1 все,

и т.д., что должно пониматься как

(x > 0 | 1 + x | (x < 0 | 1 – x | 1)).

ВАРИАНТНЫЕ-предложения, осуществляющие выбор по целому числу или виду, отличаются тем, что главная-часть-ВАРИАНТНОГО-предложения составлена из основ. Общий образец для них таков:

(– – – | – – –, ..., – – – | – – –) или

выб – – – в – – –, ..., – – – либо – – – быв

Здесь выбор также можно повторить, используя ливыб.

Компонентами выбирающего-по-целому-предложения (или вариантного-предложения) служат просто основы, но их должно быть по крайней мере две; выбор среди этих основ следует их текстуальному порядку.

Пример:

```
проц пуст работать, отдыхать, развлекаться;
выб цел день; чит (день); день
в работать, работать, работать, работать, работать, отдыхать, развлекаться
либо печ ((„дня с номером", день, „ нет в неделе"))
быв
```

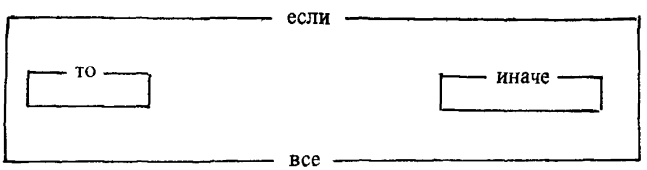
В выбирающем-по-ПРЕДСТАВИТЕЛЮ-предложении (или сопоставляющем-предложении), которое проверяет виды, каждый ВЫБИРАЮЩИЙ-вариант имеет форму (описатель идентификатор): основа или (описатель): основа. Специфицируемый этим описателем вид сравнивается с (текущим) видом проверяемого значения; упомянутый идентификатор, если он есть, обеспечивает с полной надежностью в отношении синтаксической проверки вида доступ к значению, подвергнутому проверке, внутри данной основы. Использование вида 'ПРЕДСТАВИТЕЛЬ' обеспечивает требуемую свободу для вида проверяемого значения; кроме того, 'ПРЕДСТАВИТЕЛЬ' должен содержать вид каждой из спецификаций, иначе соответствующий ВЫБИРАЮЩИЙ-вариант не будет никогда выбран.

Пример:

```
вид мальчик = ст (цел возраст, вещ рост),
вид девочка = ст (цел возраст, вещ вес);
```

```
проц об (мальчик, девочка) новорожденный;
выб новорожденный в
   (мальчик джон) : печ (рост от джон),
   (девочка мэри) : печ (вес от мэри)
все.}
```

{Иерархия блоков в условном-предложении выглядит так:



и аналогично для остальных типов выбора. Таким образом, среда и окружение выясняющего-предложения продолжают иметь силу как в главной-части-ВЫБИРАЮЩЕГО-предложения, так и в продолжении-ВЫБИРАЮЩЕГО-предложения. Однако обратная передача управления из главной-части-или продолжения-ВЫБИРАЮЩЕГО-предложения невозможна, так как выясняющее-предложение не может содержать определений-метки (, за исключением тех, которые содержатся во входящих в него ЗАКРЫТЫХ-предложениях). }

3.4.1. *Синтаксис*

```
A) ВЫБИРАЮЩЕЕ :: выбирающее по логическому; ВАРИАНТНОЕ.
B) ВАРИАНТНОЕ :: выбирающее по целому;
      выбирающее по ПРЕДСТАВИТЕЛЮ.
a) ВЫБИРАЮЩЕЕ предложение в СРЕДЕ1
      ПРИВОДИМОЕ {5D, 551a, A341h, A349a }:
   старт ВЫБИРАЮЩЕГО ОФОРМЛЕННЫЙ {91a,a},
      ОФОРМЛЕННЫЙ состав ВЫБИРАЮЩЕГО предложения
        ПРИВОДИМЫЙ в СРЕДЕ1 {b},
      финиш ВЫБИРАЮЩЕГО ОФОРМЛЕННЫЙ {91e, -}.
b) ОФОРМЛЕННЫЙ состав выбирающего по ВИДУ
      предложения ПРИВОДИМЫЙ в СРЕДЕ1 {a, 1}:
   определяющее СЛОЙ2 выясняющее предложение
      выдающее ВИД в СРЕДЕ1 {c, -},
   ОФОРМЛЕННЫЕ альтернативы выбирающего по ВИДУ
      предложения ПРИВОДИМЫЕ в СРЕДЕ1 с СЛОЕМ2 {d}.
c) определяющее новые ?ОПИСАНИЯ2 выясняющее
      предложение выдающее ВИД в СРЕДЕ1 {b, 35g}:
   кортеж с ?ОПИСАНИЯМИ2 раскрыто выдающий
      ВИД в СРЕДЕ1 с новыми ?ОПИСАНИЯМИ2 {32b}.
d) ОФОРМЛЕННЫЕ альтернативы ВЫБИРАЮЩЕГО
      предложения ПРИВОДИМЫЕ в СРЕДЕ2 {b}:
   ОФОРМЛЕННАЯ главная часть ВЫБИРАЮЩЕГО
```

предложения ПРИВОДИМАЯ в СРЕДЕ2 {e};
если ПРИВОДИМОЕ уравнивает
ПРИВОДИМОЕ1 и ПРИВОДИМОЕ2 {32e},
ОФОРМЛЕННАЯ главная часть ВЫБИРАЮЩЕГО
предложения ПРИВОДИМАЯ1 в СРЕДЕ2 {e},
ОФОРМЛЕННОЕ продолжение ВЫБИРАЮЩЕГО
предложения ПРИВОДИМОЕ2 в СРЕДЕ2 {l}.

e) ОФОРМЛЕННАЯ главная часть ВЫБИРАЮЩЕГО
предложения ПРИВОДИМАЯ в СРЕДЕ2 {d}:
вход в собственно ВЫБИРАЮЩЕЕ ОФОРМЛЕННЫЙ {91b,},
собственно выбор ВЫБИРАЮЩЕГО
ПРИВОДИМЫЙ в СРЕДЕ2 {f, g, h}.

f) собственно выбор выбирающий по логическому
ПРИВОДИМЫЙ в СРЕДЕ2 {e}:
определяющее СЛОЙ3 последовательное
предложение ПРИВОДИМОЕ в СРЕДЕ2 {32a}.

g) собственно выбор выбирающего по целому
ПРИВОДИМЫЙ в СРЕДЕ2 {e}:
образ основ ПРИВОДИМЫХ в СРЕДЕ2 {33b}.

h) собственно выбор выбирающий по ПРЕДСТАВИТЕЛЮ
ПРИВОДИМЫЙ в СРЕДЕ2 {e, h}:
выбирающий по ПРЕДСТАВИТЕЛЮ вариант
ПРИВОДИМЫЙ в СРЕДЕ2 {i};
если ПРИВОДИМОЕ уравнивает
ПРИВОДИМОЕ1 и ПРИВОДИМОЕ2 {32e}
выбирающий по ПРЕДСТАВИТЕЛЮ вариант
ПРИВОДИМЫЙ1 в СРЕДЕ2 {i}, знак а также {94f},
собственно выбор выбирающий по ПРЕДСТАВИТЕЛЮ
ПРИВОДИМЫЙ2 в СРЕДЕ2 {h}.

i) выбирающий по ПРЕДСТАВИТЕЛЮ вариант
ПРИВОДИМЫЙ в СРЕДЕ2 {h}:
определяющая СЛОЙ3 спецификация
ЗНАЧЕНИЯ в СРЕДЕ2 с СЛОЕМ3 {j, k, -},
если ЗНАЧЕНИЕ служит ПРЕДСТАВИТЕЛЕМ {64b},
основа в СРЕДЕ2 с СЛОЕМ3 ПРИВОДИМАЯ {32d}
{Здесь СЛОЙ3 :: новое СЛОВО для ВИДА; новое ПУСТО.}

j) определяющая новое СЛОВО3 для ВИДА
спецификация ВИДА в СРЕДЕ3 {i}:
упакованное кратким определяющее новое СЛОВО3
для ВИДА задание аргумента в СРЕДЕ3 {541e},
знак двоеточие {94f}.

k) определяющая новое ПУСТО спецификация
ЗНАЧЕНИЯ в СРЕДЕ3 {i}:
упакованный кратким формальный описатель
ЗНАЧЕНИЯ в СРЕДЕ3 {46b}, знак двоеточие {94f}.

l) ОФОРМЛЕННОЕ продолжение ВЫБИРАЮЩЕГО
предложения ПРИВОДИМОЕ в СРЕДЕ2 {d}:
выход собственно ВЫБИРАЮЩЕГО ОФОРМЛЕННЫЙ {91d, -},
определяющее СЛОЙ3 последовательное
предложение ПРИВОДИМОЕ в СРЕДЕ2 {32a};
продолжатель ВЫБИРАЮЩЕГО ОФОРМЛЕННЫЙ {91c, -},
ОФОРМЛЕННЫЙ состав ВЫБИРАЮЩЕГО2
предложения ПРИВОДИМЫЙ в СРЕДЕ2 {b},
если ВЫБИРАЮЩЕЕ2 может следовать за ВЫБИРАЮЩИМ {m}.

m) ЕСЛИ выбирающее по ВИДУ2 может следовать
за выбирающим по ВИДУ1 {l}:
если (ВИД1) есть (ОБЫЧНОЕ),
ЕСЛИ (ВИД2) есть (ВИД1);
если (ВИД1) начинается с (объединение),
ЕСЛИ (ВИД2) начинается с (объединение).

n)* выбирающее предложение ДЕЙСТВУЮЩЕЕ:
ВЫБИРАЮЩЕЕ предложение ДЕЙСТВУЮЩЕЕ {a}.

o)* условное предложение ДЕЙСТВУЮЩЕЕ:
выбирающее по логическому предложение ДЕЙСТВУЮЩЕЕ {a}.

p)* вариантное предложение ДЕЙСТВУЮЩЕЕ:
выбирающее по целому предложение ДЕЙСТВУЮЩЕЕ {a}.

q)* сопоставляющее предложение ДЕЙСТВУЮЩЕЕ:
выбирающее по ПРЕДСТАВИТЕЛЮ предложение
ДЕЙСТВУЮЩЕЕ {a}

{Примеры:

a) (x > 0 | x | o) ·
выб i в принстон, гренобль либо финиш быв ·
выб uir в (цел i) : печ (i), (вещ) : печ („нет") быв

b) x > 0 | x | 0

c) x > 0 · i · uir

d) | x · | x | 0

e) | x ·
в принстон, гренобль ·
в (цел i) : печ (i), (вещ) : печ („нет")

f) x

g) принстон, гренобль

h) (цел i) : печ (i), (вещ) : печ („нет")

i) (цел i) : печ (i)

j) (цел i) :

k) (вещ) :

l) либо финиш · | : x < 0 | – x | 0}

{Правило d показывает, почему 'ПРИВОДИМО выдающие ЗНАЧЕНИЕ' должны „уравниваться". Если, например, альтернативы-ВЫБИРАЮЩЕГО-предложения крепкие, то хотя бы одно из его главной-части-ВЫБИРАЮЩЕГО-предложения или продолжения-ВЫБИРАЮЩЕГО-предложения должно быть крепким, в то время как другое может быть сильным. Так, например, в (p | x | пропуск) + (p | пропуск | y) условное-предложение (p | x | пропуск)

уравнивается, делая | x крепким, а | пропуск сильным, в то время как (p | пропуск | y) уравнивается, делая | пропуск сильным, а | y крепким. Контрпример (p | пропуск | пропуск) + y показывает, что обе компоненты не могут быть сильными, поскольку иначе обозначение-операции + было бы неидентифицируемо.}

3.4.2. *Семантика*

a) Выдача W состава-ВЫБИРАЮЩЕГО-предложения C в окружении E1 определяется следующим образом:

- пусть E2 – окружение, устанавливаемое {3.2.2.b} вокруг E1 согласно выясняющему-предложению этого C;
- пусть V – выдача этого выясняющего-предложения в E2;
- W – выдача сцены, „выбранной" {b} по V из C в E2;

требуется, чтобы по области действия выдача W не была младше E1.

b) Сцена S, „выбранная" по значению V из состава-ВЫБИРАЮЩЕГО-предложения-выдающего-ЗНАЧЕНИЕ C в окружении E2, определяется следующим образом:

Случай A: 'ВЫБИРАЮЩЕЕ' есть 'выбирающее по логическому' и V есть истина:

- S – составляющая главная часть-ВЫБИРАЮЩЕГО-предложения этого C в E2;

Случай B: 'ВЫБИРАЮЩЕЕ' есть 'выбирающее по целому' и $1 \leqslant V \leqslant n$ где n – число составляющих основ составляющего собственно-выбора-ВЫБИРАЮЩЕГО этого C:

- S есть V-я такая основа в E2;

Случай C: 'ВЫБИРАЮЩЕЕ' есть 'выбирающее по ПРЕДСТАВИТЕЛЮ' и V приемлемо для {2.1.3.6.d} 'ЗНАЧЕНИЯ2' какой-нибудь составляющей спецификации-ЗНАЧЕНИЯ2 D этого C {; если таких составляющих спецификаций несколько, то не определено, какая из них выбирается в качестве D}:

- S основа, следующая за этим D в некотором {нелокализующем (3.2.2.b) } окружении, устанавливаемом вокруг E2 согласно D c V;

Остальные случаи {, когда значение V отлично от указанных}:

Если C содержит составляющее продолжение-ВЫБИРАЮЩЕГО предложения O,
то S есть O в E2;
иначе S есть пропуск-выдающий ЗНАЧЕНИЕ в E2.

3.5. Циклические предложения

{Циклические-предложения используются для динамического повторения одной и той же последовательности инструкций. Число таких повторений регулируется или некоторой конечной последовательностью равноотстоящих целых чисел, или каким-либо проверяемым каждый раз условием, или и тем и другим.

Пример 1:

цел фак := 1;
для i от n шаг – 1 до 1

```
цк фак X:= 1 кц
```

Пример 2:

```
цел a, b; чит ((a, b)) прагм здесь a ⩾ 0∧b > 0 прагм;
цел q := 0, r := a;
пока r ⩾ b прагм здесь a = b X q + r ∧ 0 ⩽ r прагм
цк (q +:= 1, r –:= b) кц
прагм здесь a = bX q + r ∧ 0 ⩽ r ∧ r < b прагм
```

(см. 9.2, где объясняются прагматы).

Идентификатор переменной цикла, например i в примере 1, определен над всем циклом. Вводимые условием-цикла определения действуют также и в теле-цикла.

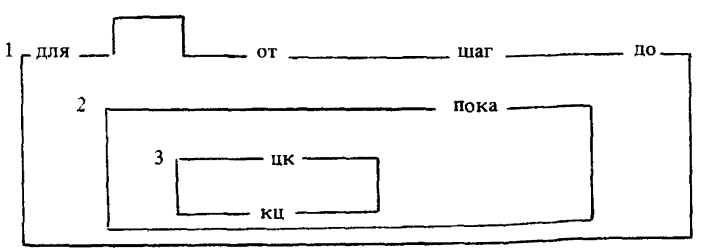
Если идентификатор переменной цикла не применяется в цикле, то заглавие-цикла можно опустить. Можно опустить настройку-нижнего-предела от 1; точно так же можно опустить и шаг 1. Настройку-верхнего-предела можно опустить, если не требуется проверять конечное значение переменной цикла. Можно опустить и условие-цикла пока истина. Например, вместо

```
для i от 1 шаг 1 до n пока истина цк печ („a") кц
```

можно написать

```
до n цк печ („a") кц
```

Иерархия блоков выглядит так:



3.5.1. *Синтаксис*

A) ЦИКЛ :: нижний предел; шаг; верхний предел.

a) циклическое предложение в СРЕДЕ1
сильно выдающее пустое значение {5D, 551a}:
ОФОРМЛЕННОЕ заглавие цикла определяющее
новое СЛОВО2 для целого в СРЕДЕ1 {b},
ОФОРМЛЕННАЯ настройка цикла в СРЕДЕ1 {c},
ОФОРМЛЕННЫЙ цикл с СЛОВОМ2 для целого в СРЕДЕ1 {e}.

b) ОФОРМЛЕННОЕ заглавие цикла определяющее
новое СЛОВО2 для целого в СРЕДЕ1 {a}:
знак для ОФОРМЛЕННЫЙ {94g, -},
определяющий СЛОВО2 идентификатор в СРЕДЕ1
с новым СЛОВОМ2 для целого выдающий целое {48a}:

если (СЛОВО2) есть (буква алеф), ПУСТО.

c) ОФОРМЛЕННАЯ настройка цикла в СРЕДЕ1 {a}:
возможно ОФОРМЛЕННАЯ
настройка нижнего предела в СРЕДЕ {d}.
возможная ОФОРМЛЕННАЯ настройка шага в СРЕДЕ1 {d}.
возможно ОФОРМЛЕННАЯ
настройка верхнего предела в СРЕДЕ1 {d}.

d) ОФОРМЛЕННАЯ настройка ЦИКЛА в СРЕДЕ1 {c}:
знак ЦИКЛ ОФОРМЛЕННЫЙ {94g, -},
основа в СРЕДЕ1 раскрыто выдающая целое {32d}.

e) ОФОРМЛЕННЫЙ ЦИКЛ с ОПИСАНИЕМ2 в СРЕДЕ1 {a}:
ОФОРМЛЕННЫЙ подчиненный условию цикл
в СРЕДЕ1 с новым ОПИСАНИЕМ2 {f};
ОФОРМЛЕННОЕ тело цикла в СРЕДЕ1 с новым
ОПИСАНИЕМ2 {h}.

f) ОФОРМЛЕННЫЙ подчиненный условию цикл в СРЕДЕ2 {e}:
ОФОРМЛЕННОЕ определяющее СЛОЙ3
условие цикла в СРЕДЕ2 {g},
ОФОРМЛЕННОЕ тело цикла в СРЕДЕ2 с СЛОЕМ3 {h}.

g) ОФОРМЛЕННОЕ определяющее СЛОЙ3
условие цикла в СРЕДЕ2 {f}:
знак пока ОФОРМЛЕННЫЙ {94g, -},
определяющее СЛОЙ3 выясняющее предложение
выдающее логическое в СРЕДЕ2 {34c, -},

h) ОФОРМЛЕННОЕ тело цикла в СРЕДЕ3 {e, f}:
знак цикл ОФОРМЛЕННЫЙ {94g, -},
определяющее СЛОЙ4 последовательное предложение
сильно выдающее пустое значение в СРЕДЕ3 {32a},
знак конец цикла ОФОРМЛЕННЫЙ {94g, -}

{Примеры:

a) для i пока i < n цк задача1 кц • до n цк задача1; задача2 кц

b) для i

c) от –5 до +5

d) от –5

e) пока i < n цк задача1 кц • цк задача1; задача2 кц

f) пока i < n цк задача1; задача2 кц

g) пока i < n

h) цк задача1; задача2 кц}

3.5.2. *Семантика*

Циклическое-предложение С в окружении Е1 исполняется посредством следующих шагов:

Шаг 1: Все составляющие настройки-ЦИКЛА из С, если они вообще есть, исполняются совместно с Е1;

• пусть f – выдача составляющей настройки-нижнего-предела из С, если она есть, а иначе f будет 1;

• пусть b – выдача составляющей настройки-шага из С, если она есть, а иначе b будет 1;
• пусть t – выдача составляющей настройки-верхнего-предела из С, если она есть, а иначе пусть t отсутствует;
• пусть Е2 будет {нелокализующим (3.2.2.b)} окружением, устанавливаемым вокруг Е1 согласно заглавию-цикла-определяющему-новое-СЛОВО2-для целого из С и с целым числом f;

Шаг 2: Пусть i – целое число, доступное {2.1.2.c} для 'СЛОВА2 для целого' внутри участка окружения Е2;
Если t не отсутствует,
то
если b < 0 и i < t или если b > 0 и i > t,
то С в Е1 {завершается и} выдает пусто;
{иначе предпринимается шаг 3;}

Шаг 3: Пусть окружение Е3 и истинностное значение w определяются следующим образом:

Случай А: С не содержит составляющего условия-цикла:
• Е3 есть Е2;
• w есть истина;

Случай В: С содержит составляющее условие-цикла Р:
• Е3 есть {возможно нелокализующее (3.3.2.b)} окружение, устанавливаемое вокруг Е2 согласно выясняющему-предложению из Р;
• w – выдача этого выясняющего-предложения в Е3;

Шаг 4:
Если w – истина,
то
• составляющее тело-цикла данного С исполняется в Е3;
• 'СЛОВО2 для целого' получает доступ к i + b внутри участка окружения Е2;
• шаг 2 предпринимается снова;
иначе
• С в Е1 {завершается и} выдает пустое.

{Циклическое-предложение
для i от u1 шаг u2 до u3 пока условие цк действие кц
эквивалентно. таким образом, замкнутому-предложению-выдающему-пустое-значение
начало цел f := u1 , цел b = u2, t = u3;
шаг2:
если $(b > 0 \wedge f \leqslant t) \vee (b < 0 \wedge f \geqslant t) \vee b = 0$
то цел i = f;
если условие
то действие; f +:= b; на шаг2
все
все
конец.

Разумеется, такой эквивалентности может не быть, если данное циклическое-предложение содержит локальные-генераторы или некоторые из использованных обозначений-операций не идентифицируют обозначения-операций в стандартной языковой обстановке (10).} •

4. ОПИСАНИЯ, ОПИСАТЕЛИ И ИНДИКАТОРЫ

{Описания служат для того, чтобы
- вводить новые индикаторы, например, идентификаторы,
- определять их виды или приоритеты и
- приписывать этим индикаторам значения и инициализировать переменные.}

4.1. О п и с а н и я

4.1.1. *Синтаксис*

A) ОБЪЕКТ :: вид; приоритет; тождество для ПРОВИДА;
переменная как имя ПРОВИДА; операция как ПРОВИД;
ПАРАМЕТР; поле вида ВИД среди ПОЛЕЙ.
{ПРОВИД :: процедура; ВИД.}

a) описание !ОПИСАНИЙ в СРЕДЕ {a, 32b}:
описание ОБЪЕКТОВ через !ОПИСАНИЯ
в СРЕДЕ {42a, 43a, 44a, e, 45a, -};
если (!ОПИСАНИЯ) есть (!ОПИСАНИЯ1 !ОПИСАНИЯ2),
описание ОБЪЕКТОВ через !ОПИСАНИЯ1
в СРЕДЕ {42a, 43a, 44a, e, 45a, -},
знак а также {94f},
описание !ОПИСАНИЙ2 в СРЕДЕ {a}.

b) групповое определение ОБЪЕКТОВ через !ПАРЫ ПАРУ
в СРЕДЕ {b, 42a, 43a, 44a, e, 45a, 46e, 541e}:
групповое определение ОБЪЕКТОВ через !ПАРЫ в СРЕДЕ
{b, c}, знак а также {94f},
групповое определение ОБЪЕКТА через ПАРУ в СРЕДЕ {c}.

c) групповое определение ОБЪЕКТА через ПАРУ
в СРЕДЕ {b, 42a, 43a, 44a,e, 45a, 46e, 541e}:
определение ОБЪЕКТА через ПАРУ
в СРЕДЕ {42b, 43b, 44c,f, 45c, 46f, 541f, -}.

d)* определение ПАРЫ: определение ОБЪЕКТА через ПАРУ
в СРЕДЕ {42b, 43b, 44c, f, 45c, 46f, 541f};
определение метки через ПАРУ в СРЕДЕ {32c}
{Примеры:
a) вид r = имя вещ, s = лит • прио∨ = 2, ∧ = 3 •
цел m = 4096 • вещ x, y •
оп∨ = (лог a, b) лог : (a | истина | b)
b) r = имя вещ, s = лит • ∨= 2, ∧ = 3 • m = 4096 •
x, y • ∨= (лог a, b) лог : (a | истина | b)
c) r = имя вещ • ∨= 2 • m = 4096 • x •

∨ = (лог a, b) лог : (a | истина | b) }

4.1.2. *Семантика*

Исполнение описания состоит в совместном исполнении его описания-ОБЪЕКТОВ и его описания, если оно есть. {Таким образом, все описания-ОБЪЕКТОВ, разделенные знаками-а-также, исполняются совместно.}

4.2 Описания видов

{Описания-видов задают определяющие-индикаторы-вида, играющие роль сокращений для описателей, построенных из более примитивных компонент, или из других описателей, или даже из самих себя.

Например,

вид массив = [m, n] вещ и

вид книга = ст (строк текст, имя книга следующая)

В последнем примере использующий-индикатор-вида книга служит не только удобным сокращением, но и по существу необходим для данного описания.}

4.2.1. *Синтаксис*

a) описание видов через !ОПИСАНИЯ в СРЕДЕ {41a}:
знак вид {94d},
групповое определение видов
через !ОПИСАНИЯ в СРЕДЕ {41b, c}:

b) определение вида через ИНДИКАНТ
для ЗНАЧЕНИЯ НОМЕР в СРЕДЕ {41c}:
если (ИНДИКАНТ) есть (выделенное СЛОВО) или
(СРЕДА) есть (новое с СЛОЕМ),
определяющий ИНДИКАНТ индикатор вида в
СРЕДЕ выдающий ЗНАЧЕНИЕ НОМЕР {48a},
знак определяется как {94d},
фактический описатель ЗНАЧЕНИЯ НОМЕР в СРЕДЕ {c}.

c) фактический описатель ЗНАЧЕНИЯ НОМЕР1 в СРЕДЕ {b}:
если (НОМЕР1) есть (I),
фактический определитель ЗНАЧЕНИЯ
в СРЕДЕ { 46c, d, g, h, o, s, -};
если (НОМЕР1) есть (НОМЕР2 I),
использующий ИНДИКАНТ2 индикатор вида в
СРЕДЕ выдающий ЗНАЧЕНИЕ НОМЕР2 {48b}

{Примеры:

a) вид r = имя вещ, s = лит

b) r = имя вещ c) имя вещ • лит

{Использование 'НОМЕРА' исключает круговые цепочки определений-видов, такие, как вид *a = b, b = a*.

Определяющие-?РАЗМЕРНОЕ-СТАНДАРТНОЕ-индикаторы-видов могут описываться только в стандартном-вступлении, где среда имеет форму 'новое с СЛОЕМ' (10.1.1.b).}

4.2.2. *Семантика*

Исполнение описания-видов {не требует действия, не выдает значен

тем самым } завершено.

4.3 Описание приоритетов

{Описания-приоритетов используются для определения приоритета обозначений-операций. Существуют приоритеты от 1 до 9.

Так как обозначения-унарных-операций фактически имеют только один уровень приоритета, более высокий, чем у всех обозначений-бинарных-операций, то описания-приоритетов для обозначений-унарных-операций не нужны.}

4.3.1. *Синтаксис*

a) описание приоритетов через !ОПИСАНИЯ в СРЕДЕ {41a}:
знак приоритет {94d},
групповое определение приоритетов
через !ОПИСАНИЯ в СРЕДЕ {41b, c}.

b) определение приоритета через ИНФИКС для
приоритета ПРИОРИТЕТ в СРЕДЕ {41c}:
определяющее ИНФИКС обозначение операции в СРЕДЕ
выдающее приоритет ПРИОРИТЕТ {48a},
знак определяется как {94d}, знак ЦИФРА {94b},
если ЦИФРА считает ПРИОРИТЕТ {c, d}.
{ЦИФРА :: цифра нуль; цифра один; цифра два; цифра три;
цифра четыре; цифра пять; цифра шесть; цифра семь;
цифра восемь; цифра девять.}

c) ЕСЛИ ЦИФРА1 считает ПРИОРИТЕТ I. {b, c}:
ЕСЛИ ЦИФРА2 считает ПРИОРИТЕТ {c, d},
если (цифра один цифра два цифра три цифра четыре
цифра пять цифра шесть цифра семь цифра восемь
цифра девять) содержит (ЦИФРУ2 ЦИФРУ1).

d) ЕСЛИ цифра один считает I ·{b, c} : ЕСЛИ истина.
{Примеры:
а) прио ∨ = 2, ∧ =3 b) ∨ = 2}

4.3.2. *Семантика*

Исполнение описания-приоритетов {не требует действий, не выдает значения и тем самым} завершено.

4.4. Описания идентификаторов

{Описания-идентификаторов задают определяющие-идентификаторы-выдающие-ВИД при помощи либо описаний-тождеств, либо описаний-переменных.

Примеры:

вещ пи = 3.1416 ·

вещ точность := 0.05.

Второй пример, который служит описанием-переменной, можно рассматривать как эквивалентную форму для описания-тождества

имя вещ точность = лок вещ := 0.05.

Исполнение описаний-идентификаторов вызывает приписывание значений их идентификаторам; в приведенных выше примерах 3.1416 приписы-

вается пи и новое локальное имя, именующее 0.05, приписывается точности.}

4.4.1. *Синтаксис*

A) ПРОВИД :: процедура: ВИД.

B) ЛОКАЛИЗУЮЩИЙ :: локальный, глобальный; первичный.

a) описание тождеств для ПРОВИДА
через !ОПИСАНИЯ в СРЕДЕ {41a}:
формальный описатель ПРОВИДА в СРЕДЕ {b, 46b},
групповое определение тождеств для ПРОВИДА
через !ОПИСАНИЯ в СРЕДЕ {41 b, c}.

b) ЛЮБОЙ описатель процедуры
в СРЕДЕ {a, 523b}: знак процедура {94d}.

c) определение тождества для ПРОВИДА
через СЛОВО для ВИДА в СРЕДЕ {41c}:
определяющий СЛОВО идентификатор в СРЕДЕ
выдающий ВИД {48a}, знак определяется как {94d},
источник вида ВИД для ПРОВИДА в СРЕДЕ {d}.

d) источник вида ВИД для ПРОВИДА в СРЕДЕ {c, f, 45c}:
если (ПРОВИД) есть (ВИД),
источник вида ВИД в СРЕДЕ {521c};
если (ПРОВИД) есть (процедура),
текст процедуры в СРЕДЕ выдающий ВИД {541a, b, -}.

e) описание переменных как имен ПРОВИДА
через !ОПИСАНИЯ в СРЕДЕ {41a}:
задание ЛОКАЛИЗУЮЩЕГО генератора в СРЕДЕ
выдающего имя ПРОВИДА {523b},
групповое определение переменных как
имен ПРОВИДА через !ОПИСАНИЯ в СРЕДЕ {41b, c}.

f) определение переменных как имен ПРОВИДА
через СЛОВО для имени ВИДА в СРЕДЕ {41c}:
определяющий СЛОВО идентификатор в СРЕДЕ
выдающий имя ВИДА {48a}, знак присвоить {94c},
источник вида ВИД для ПРОВИДА в СРЕДЕ {d};
если (ПРОВИД) есть (ВИД),
определяющий СЛОВО идентификатор в СРЕДЕ
выдающий имя ВИДА {48a}.

g)* описание идентификаторов:
описание тождеств для ПРОВИДА
через !ОПИСАНИЯ в СРЕДЕ {a};
описание переменных как имен ПРОВИДА
через !ОПИСАНИЯ в СРЕДЕ {e}.

{Примеры:

a) цел m = 4096 · проц r 10 = вещ : псч X 10

b) проц c) m = 4096

d) 4096 · вещ : псч X 10

e) вещ x, y • проц pp := вещ : псч X 10
f) pp := вещ : псч X 10 • x}
4.4.2. *Семантика*
a) Описание-тождеств D исполняется следующим образом:
• совместно исполняются составляющие источники-для ПРОВИДА из D;
Для каждого составляющего определения-тождества D1 из D
• выдача V источника-для-ПРОВИДА данного D1 приписывается {4.8.2.a} определяющему-идентификатору этого D1.
b) Описание-переменных D исполняется следующим образом:
• задание-генератора {5.2.3.1.b} G этого D и все источники-для-ПРОВИДА, если они есть, составляющих определений-переменных из D исполняются совместно;
Для каждого составляющего определения-переменной-через СЛОВО-для имени-ВИДА D1 из D,
• пусть W1 – какой-то „вариант" {c} для 'ВИДА' значения, именуемого выдачей N данного G;
• пусть N1 – вновь созданное, имя, равное N по области действия и именующее W1;
• если N1 – составное имя {2.1.3.2.b}, то N1 снабжается подыменами {2.1.3.3.e, 2.1.3.4.g};
• N1 приписывается {4.8.2.a} определяющему-идентификатору данного D1;
• выдача источника-для-ПРОВИДА, если он есть, этого D1 присваивается {5.2.1.2.b} N1.

{Фактический-описатель, общий для нескольких определений-переменных, исполняется только однажды. Например, исполнение

цел m := 10; [1 : m +:= 1] цел p, q; печ (m)

вызовет печать 11, а не 12; кроме того, p и q будут приписаны два новых локальных имени, именующих массивы с паспортами ((1, 11)) и неопределенными элементами.}

c) „Вариантом" значения V для вида M будет некоторое значение W, приемлемое для {2.1.3.6.d} M и определяемое следующим образом:
Случай A: M есть 'структура содержащая !ПОЛЯ в себе':
Для каждого 'СЛОВА для выборки ВИДА', заложенного в эти '!ПОЛЯ',
• полем, выбираемым по 'СЛОВУ' в W, будет вариант для 'ВИДА' того поля, которое выбирается по СЛОВУ в V;
Случай B: M есть '?ПОДВИЖНЫЙ МАССИВ из ВИДА1':
• паспортом значения W служит паспорт данного значения V;
• каждый элемент этого W есть вариант для 'ВИДА1' некоторого элемента значения V;
Остальные случаи:
• W – любое значение, приемлемое для M.

d) Выдачей фактического-описателя-процедуры является какая-нибудь процедура {, вид которой не существен}.

4.5. Описания операций

{Описания-операций задают определяющие-обозначения-операций.

Пример:

оп mc = (вещ a, b) вещ : (3 × a < b | a | b).

В отличие от случая, например, описаний-идентификаторов, в одну и ту же зону могут входить более одного описания-операции с одним и тем же знаком-АФФИКСА; например, предыдущий пример прекрасно может находиться в одной зоне с

оп mc = (компл маккарти, джон) компл: (псч < .5 | маккарти | джон);

в этом случае обозначение-операции mc называется „перегруженным".}

4.5.1. *Синтаксис*

A) ОПЕРАЦИЯ :: ДВУМЕСТНАЯ; ОДНОМЕСТНАЯ.

B) АФФИКС :: ИНФИКС; ПРЕФИКС.

a) описание операций как ПРОВИДА
через !ОПИСАНИЯ в СРЕДЕ {41a}:
знак операция {94d}, формальный план ПРОВИДА
в СРЕДЕ {b, 46p, -},
групповое определение операций как ПРОВИДА
через !ОПИСАНИЯ в СРЕДЕ {41b, c}.

b) формальный план процедуры в СРЕДЕ {в}: ПУСТО.

c) определение операции как ПРОВИДА через
АФФИКС для ОПЕРАЦИИ в СРЕДЕ {41c}:
определяющее АФФИКС обозначение операции в СРЕДЕ
выдающее ОПЕРАЦИЮ {48a},
знак определяется как {94d},
источник вида ОПЕРАЦИЯ для ПРОВИДА в СРЕДЕ {44d}.

{Примеры:

a) оп ∨= (лог a, b) лог : (a | истина | b)

c) ∨= (лог a, b) лог : (a | истина | b)}

4.5.2. *Семантика*

a) Исполнение описания-операций состоит в совместном исполнении его составляющих определений-операций.

b) Определение-операции исполняется приписыванием {4.8.2.a} его определяющему-обозначению-операции процедуры, выдаваемой его источником-для-ПРОВИДА.

4.6. Описатели

{Описатели специфицируют виды. Всякий описатель является или определителем, явно описывающим какой-то вид, или использующим-индикатором-вида, который употребляется вместо некоторого определителя через описание-вида. Определители формируются из символов пуст, цел, вещ, лог и лит (10.2.2) при помощи других символов, а именно имя, ст, [ ], проц и об. Например, проц (вещ) лог специфицирует вид 'процедура с параметром вида вещественное вырабатывающая логическое'.

Фактические-описатели, применяемые главным образом в генераторах, требуют, чтобы в них были заданы границы. Формальные-описатели, применяемые главным образом в формальных-параметрах и ядрах, не требуют границ. Следующий за имя описатель всегда 'виртуальный' и может, поскольку подвижность есть свойство имен, специфицировать 'подвижный МАССИВ из ВИДА'. Так как в генераторах фактические-описатели неявно следуют за 'имя', они также могут специфицировать 'подвижный МАССИВ из ВИДА'.}

4.6.1. *Синтаксис*

A) ЛЮБОЙ :: НЕФОРМАЛЬНЫЙ; формальный.
B) НЕФОРМАЛЬНЫЙ :: виртуальный; фактический.
C) !ЗНАЧЕНИЯ :: ЗНАЧЕНИЕ; !ЗНАЧЕНИЯ ЗНАЧЕНИЕ.
a) НЕФОРМАЛЬНЫЙ описатель ЗНАЧЕНИЯ
в СРЕДЕ {c, e, g, h, 523a, b}:
НЕФОРМАЛЬНЫЙ определитель ЗНАЧЕНИЯ
в СРЕДЕ { c, d, g, h, o, s, -};
использующий ИНДИКАТ индикатор вида в СРЕДЕ
выдающий ЗНАЧЕНИЕ НОМЕР {48b, -}
b) формальный описатель ЗНАЧЕНИЯ
в СРЕДЕ {e, h, p, r, u, 34k, 44a, 541a, b, e, 551a}:
если ЗНАЧЕНИЕ фиксирует ЗНАЧЕНИЕ {47a, b, c, -},
формальный определитель ЗНАЧЕНИЯ
в СРЕДЕ {c, d, h, o, s, -};
использующий ИНДИКАТ индикатор вида в СРЕДЕ
выдающий ЗНАЧЕНИЕ1 НОМЕР{48b,}
если ЗНАЧЕНИЕ фиксирует ЗНАЧЕНИЕ1 {47a, b, c, -}.
c) ЛЮБОЙ определитель имени ВИДА в СРЕДЕ {a, b, 42c}:
знак имя {94d},
виртуальный описатель ВИДА в СРЕДЕ {a}.
d) ЛЮБОЙ определитель структуры
содержащей !ПОЛЯ в себе в СРЕДЕ {a, b, 42c}:
знак структура {94d}, упакованный кратким
ЛЮБОЙ образ !ПОЛЕЙ среди !ПОЛЕЙ в СРЕДЕ {e}.
e) ЛЮБОЙ образ !ПОЛЕЙ1 среди !ПОЛЕЙ в СРЕДЕ { d, e}:
ЛЮБОЙ описатель ВИДА в СРЕДЕ {a, b},
групповое определение полей вида ВИД
среди !ПОЛЕЙ через !ПОЛЯ1 в СРЕДЕ {41b, c};
если (!ПОЛЯ1) есть (!ПОЛЯ2 !ПОЛЯ 3),
ЛЮБОЙ описатель ВИДА в СРЕДЕ {a, b},
групповое определение полей вида ВИД
среди !ПОЛЕЙ через !ПОЛЯ2 в СРЕДЕ {41b, c},
знак а также {94f},
ЛЮБОЙ образ !ПОЛЕЙ3 среди !ПОЛЕЙ в СРЕДЕ {e}.
f) определение поля вида ВИД среди !ПОЛЕЙ
через СЛОВО для выборки ВИДА в СРЕДЕ {41c}:

определяющий СЛОВО указатель поля
вида ВИД среди !ПОЛЕЙ {48c}.

g) НЕФОРМАЛЬНЫЙ определитель подвижного МАССИВА
из ВИДА в СРЕДЕ {a, 42c}:
знак подвижное {94d}, НЕФОРМАЛЬНЫЙ описатель
МАССИВА из ВИДА в СРЕДЕ {a}.

h) ЛЮБОЙ определитель МАССИВА из ВИДА в СРЕДЕ {a, b, 42c}:
индексованный ОФОРМЛЕННЫЙ
ЛЮБОЙ диапазон МАССИВА в СРЕДЕ {i, j, k, l},
ЛЮБОЙ описатель ВИДА в СРЕДЕ {a, b}.

i) ЛЮБОЙ диапазон вектора МАССИВОВ в СРЕДЕ {h, i}:
ЛЮБОЙ диапазон вектора в СРЕДЕ {j, k, l},
знак а также {94f},
ЛЮБОЙ диапазон МАССИВА в СРЕДЕ {i, j, k, l}.

j) фактический диапазон вектора в СРЕДЕ {h, i}:
нижняя граница в СРЕДЕ {m}, знак вплоть до {94f},
верхняя граница в СРЕДЕ {n};
верхняя граница в СРЕДЕ {n}.

k) виртуальный диапазон вектора в СРЕДЕ {h, i}:
возможный знак вплоть до {94f}.

l) формальный диапазон вектора в СРЕДЕ {h, i}:
возможный знак вплоть до {94f}.

m) нижняя граница в СРЕДЕ {j, 532f, g}:
основа в СРЕДЕ раскрыто выдающая целое {32d}.

n) верхняя граница в СРЕДЕ {j, 532f}:
основа в СРЕДЕ раскрыто выдающая целое {32d}.

o) ЛЮБОЙ определитель ПРОЦЕДУРЫ в СРЕДЕ {a, b, 42c}:
знак процедура {94d},
формальный план ПРОЦЕДУРЫ в СРЕДЕ {p}.

p) формальный план процедуры ?ПАРАМЕТРИЗОВАННОЙ
вырабатывающей ЗНАЧЕНИЕ в СРЕДЕ {o, 45a}:
если (?ПАРАМЕТРИЗОВАННАЯ) есть (ПУСТО),
формальный описатель ЗНАЧЕНИЯ в СРЕДЕ {b};
если (?ПАРАМЕТРИЗОВАННАЯ) есть (с !ПАРАМЕТРАМИ),
упакованный кратким групповой
описатель !ПАРАМЕТРОВ в СРЕДЕ {q, r},
формальный описатель ЗНАЧЕНИЯ в СРЕДЕ {b}.

q) групповой описатель !ПАРАМЕТРОВ
ПАРАМЕТРА в СРЕДЕ {p, q}:
групповой описатель !ПАРАМЕТРОВ в СРЕДЕ {q, r},
знак а также {94f},
групповой описатель ПАРАМЕТРА в СРЕДЕ {r}.

r) групповой описатель
параметра вида ВИД в СРЕДЕ {p, q}:
формальный описатель ВИДА в СРЕДЕ {b}.

s) ЛЮБОЙ определитель объединения !ОБЫЧНЫХ1
ОБЫЧНОГО1 воедино в СРЕДЕ {a, b, 42c}:
если неверно что ПУСТО
родственно !ОБЫЧНЫМ1 ОБЫЧНОМУ1 {47f},
знак объединение {94d}, упакованный кратким
групповой описатель !ЗНАЧЕНИЙ в СРЕДЕ {t, u},
если !ЗНАЧЕНИЯ сплетены с !ОБЫЧНЫМИ2 {47g} и
укрытые !ОБЫЧНЫЕ1 ОБЫЧНОЕ1
входят в укрытые !ОБЫЧНЫЕ2 {73l} и
укрытые !ОБЫЧНЫЕ2 входят
в укрытые !ОБЫЧНЫЕ1 ОБЫЧНОЕ1 {73l, m}.

t) групповой описатель !ЗНАЧЕНИЙ
ЗНАЧЕНИЯ в СРЕДЕ {s, t}:
групповой описатель !ЗНАЧЕНИЙ в СРЕДЕ {t, u},
знак а также {94f},
групповой описатель ЗНАЧЕНИЯ в СРЕДЕ {u}.

u) групповой описатель ЗНАЧЕНИЯ в СРЕДЕ {s, t}:
формальный описатель ЗНАЧЕНИЯ в СРЕДЕ {b}
{Примеры:
a) [1 : n] вещ • лицо
b) [ ] вещ • строк
c) имя вещ
d) ст (цел возраст,
имя лицо отец, сын)
e) имя лицо отец, сын •
цел возраст, имя
лицо отец, сын
f) возраст
g) подв [1 : n] вещ
h) [1 : m, 1 : n] вещ
i) 1 : m, 1 : n
j) 1 : n
k) :
l) :
m) 1
n) n
o) проц (лог, лог) лог
p) (лог, лог) лог
q) лог, лог
r) лог
s) об (цел, лит)
t) цел, лит
u) цел}

{О фактических-описателях-ЗНАЧЕНИЯ-НОМЕР см. 4.2.1c, о фактических-описателях-процедуры см. 4.4.1.b.

Не существует описателей, специфицирующих такие виды, как 'объединение целого объединения целого вещественного воедино воедино' или 'объединение целого вещественного целого воедино'. В действительности описатели об (цел, об (цел, вещ)) и об (цел, вещ, цел) можно написать, но в обоих случаях специфицируемым видом будет 'объединение целого вещественного воедино' (, что с таким же успехом можно выписать и как 'объединение вещественного целого воедино').}

4.6.2. *Семантика*

a) Выдача W фактического-описателя-ВИДА D в окружении E определяется следующим образом:

Если 'ВИД' есть некоторое 'СОСТАВНОЕ',

то

- пусть D1 в E1 „развертывается" {c} из D в E;
- W – выдача {определителя} D1 в {локализующем, см. 3.2.2.b,} окружении, устанавливаемом по E и вокруг E1;

иначе

- W – любое значение {приемлемое для 'ВИДА'}.

b) Выдача W фактического-определителя-СОСТАВНОГО D определяется следующим образом:

Случай A: 'СОСТАВНОЕ' является 'структурой содержащей !ПОЛЯ в себе':

- совместно исполняются составляющие описатели данного D;
- каждое поле выдачи W есть вариант {4.4.2.c}

(i) выдачи последнего составляющего описателя-ВИДА из D, находящегося перед составляющим определяющим-указателем-поля этого D, выбирающим {2.1.5.g} данное поле

(ii) для данного 'ВИДА';

Случай B: 'СОСТАВНОЕ' является 'МАССИВОМ из ВИДА':

- совместно исполняются все составляющие нижние- и верхние-границы данного D и его описатель D1;

Для i = 1, ..., n, где n – число 'векторов', содержащихся в 'МАССИВЕ',

- пусть l_i – выдача нижней-границы, если она есть, i-го составляющего диапазона-вектора этого D, а иначе l_i будет 1;
- пусть u_i – выдача верхней-границы этого диапазона-вектора,
- W состоит из

(i) паспорта $((l_1, u_1), ..., (l_n, u_n))$,

(ii) вариантов выдачи описателя D1 для 'ВИДА';

Случай C: 'СОСТАВНОЕ' является 'подвижным МАССИВОМ из ВИДА':

- W – выдача описателя данного D.

c) Сцена S, „развертываемая из" фактического-описателя-СОСТАВНОГО D в окружении E, определяется следующим образом:

Если видимый прямой наследник D1 данного D есть некоторый индикатор-вида,

то

- S – сцена, развертываемая из сцены, выдаваемой D1 в E,

иначе {D1 – определитель},

- S – составляется из D1 и E.

d) Всякий данный описатель-ЗНАЧЕНИЯ „специфицирует" вид 'ЗНАЧЕНИЕ'.

4.7. Соотношения между видами

{Некоторые виды должны фиксироваться, потому что вид никакого значения не может быть подвижным {2.1.3.6.b}. Родственные объединения не должны допускаться во избежание двусмысленности. Множество 'ПРЕДСТАВИТЕЛЕЙ' и 'ОБЫЧНЫХ' может быть сплетено заменой всех этих 'ПРЕДСТАВИТЕЛЕЙ' их 'ОБЫЧНЫМИ' компонентами.}

4.7.1. *Синтаксис*

A) НЕСОСТАВНОЕ :: ПРОСТОЕ; ИМЯ ВИДА; ПРОЦЕДУРА;
ПРЕДСТАВИТЕЛЬ; пустое значение.

B) ?ОБЫЧНОЕ :: !ОБЫЧНЫЕ; ПУСТО.

C) ?ЗНАЧЕНИЯ :: !ЗНАЧЕНИЯ; ПУСТО.

a) ЕСЛИ НЕСОСТАВНОЕ фиксирует
НЕСОСТАВНОЕ {b, e, 46b, 521c, 62a, 71n} : ЕСЛИ истина.

b) ЕСЛИ МАССИВ из ВИДА2 фиксирует
?ПОДВИЖНЫЙ МАССИВ изВИДА1 {b, e, 46b, 521c, 62a, 71n}:
ЕСЛИ ВИД2 фиксирует ВИД1 {a, b, c, -}.

c) ЕСЛИ структура содержащая !ПОЛЯ2 в себе
фиксирует структуру содержащую !ПОЛЯ1
в себе {b, e, 46b, 521c, 62a, 71n}:
ЕСЛИ !ПОЛЯ2 фиксируют !ПОЛЯ1 {d, e,}

d) ЕСЛИ !ПОЛЯ2 ПОЛЕ2 фиксируют !ПОЛЯ1 ПОЛЕ1 {c, d}:
ЕСЛИ !ПОЛЯ2 фиксируют !ПОЛЯ1 {d, e, -}
и ПОЛЕ2 фиксирует ПОЛЕ1 {e, -}.

e) ЕСЛИ СЛОВО для выборки ВИДА2
фиксирует СЛОВО для выборки ВИДА1 {c, d}:
ЕСЛИ ВИД2 фиксирует ВИД1 {a,b, c, - }.

f) ЕСЛИ ?ОБЫЧНЫЕ1 родственны ?ОБЫЧНЫМ2 {f, 46s}:
если (?ОБЫЧНЫЕ2) есть (ОБЫЧНОЕ ?ОБЫЧНЫЕ3),
ЕСЛИ ?ОБЫЧНЫЕ1 ОБЫЧНОЕ родственны ?ОБЫЧНЫМ3 {f}
или ОБЫЧНОЕ скреплено с объединением ?ОБЫЧНЫХ1
?ОБЫЧНЫХ3 воедино {71m};
если (?ОБЫЧНЫЕ2) есть (ПУСТО), ЕСЛИ ложь.

g) ЕСЛИ !ЗНАЧЕНИЯ сплетены с !ОБЫЧНЫМИ {g, 46s}:
если (!ЗНАЧЕНИЯ) есть (!ОБЫЧНЫЕ), ЕСЛИ истина;
если (!ЗНАЧЕНИЯ) есть (?ОБЫЧНЫЕ объединение !ОБЫЧНЫХ1
воедино ?ЗНАЧЕНИЯ),
ЕСЛИ ?ОБЫЧНЫЕ !ОБЫЧНЫЕ1 ?ЗНАЧЕНИЯ
сплетены с !ОБЫЧНЫМИ {g}.

{Никакой составляющий вид объединения не может приводиться укреплением к одному из других составляющих видов этого объединения или к их объединению (, правило f), так как в противном случае может возникнуть двусмысленность. Например, ядро

об (имя цел, цел) (лок цел)

двусмысленно в том, что разыменование может как появиться, так и нет перед объединением. Аналогично

вид сеп = об (середи, петер);
об (имя сеп, сеп) (лок сеп)

двусмысленно. Отметим, что из-за сплетения (, правило g,) вид, специфицируемый описателем данного ядра, точнее обозначается описателем об (имя сеп, середи, петер).}

4.8. Индикаторы и указатели полей

4.8.1. *Синтаксис*

A) ИНДИКАТОР :: идентификатор; индикатор вида;
обозначение операции.

B) ПРИМЕНЯЮЩИЙ :: определяющий; использующий.

C) ?ПАРЫ :: !ПАРЫ; ПУСТО.

D) !ПАРЫ :: ПАРА; !ПАРЫ ПАРА.

E) ПАРА :: ОПИСАНИЕ; МЕТКА; ПОЛЕ.
{ПАРА :: ОБОЗНАЧЕНИЕ для ПРИЗНАКА.}

F) ПРИЗНАК :: ВИД; ЗНАЧЕНИЕ НОМЕР; БИНАРНОЕ;
метка; выборка ВИДА.

G) ОБОЗНАЧЕНИЕ :: СЛОВО; ИНДИКАНТ; ИНФИКС; ПРЕФИКС.

a) определяющий ОБОЗНАЧЕНИЕ ИНДИКАТОР в СРЕДЕ с
новыми ?ПАРАМИ1 ОБОЗНАЧЕНИЕМ для ПРИЗНАКА ?ПАРАМИ2
выдающий ПРИЗНАК {32c, 35b, 42b, 43b, 44c, f, 45c, 541f}:
знак ОБОЗНАЧЕНИЕ {942A, D, F, K},
если ОБОЗНАЧЕНИЕ для ПРИЗНАКА не зависит
от ?ПАР1 ?ПАР2 {71a, b, c}.

b) использующий ОБОЗНАЧЕНИЕ ИНДИКАТОР в СРЕДЕ
выдающий ПРИЗНАК {42c, 46a, b, 5D, 542a, b, 544a}:
знак ОБОЗНАЧЕНИЕ {942A, D, F, K},
если ОБОЗНАЧЕНИЕ для ПРИЗНАКА
идентифицировано в СРЕДЕ {72a}.

c) определяющий СЛОВО указатель поля вида ВИД
среди ?ПАР1 СЛОВА для выборки ВИДА ?ПАР2 {46f}:
знак СЛОВО {942A},
если СЛОВО для выборки ВИДА
не зависит от ?ПАР1 ?ПАР2 {71a, b, c}.

d) использующий СЛОВО указатель поля
вида ВИД среди !ПОЛЕЙ {531a}:
знак СЛОВО {942A},
если СЛОВО для выборки ВИДА
находится в !ПОЛЯХ {72b, c, -}.

e)* ПРИМЕНЯЮЩИЙ ОБОЗНАЧЕНИЕ индикатор в СРЕДЕ
выдающий ПРИЗНАК:
ПРИМЕНЯЮЩИЙ ОБОЗНАЧЕНИЕ ИНДИКАТОР
в СРЕДЕ выдающий ПРИЗНАК {a, b}.

f)* ПРИМЕНЯЮЩИЙ СЛОВО указатель поля вида ВИД:
ПРИМЕНЯЮЩИЙ СЛОВО указатель поля
вида ВИД среди !ПОЛЕЙ {c, d}.

{Примеры:
a) x (в вещ x, y)
b) x (в x + y)
c) следующая (см. 1.1.2)
d) следующая (в следующая из проект)}

4.8.2. *Семантика*

a) Когда какое-нибудь значение или сцена V „приписывается" определяющему ОБОЗНАЧЕНИЕ-индикатору-выдающему-ПРИЗНАК в окружении E, 'ОБОЗНАЧЕНИЕ для ПРИЗНАКА' получает доступ к V внутри участка этого E {2.1.2.c}.

b) Выдача W использующего-ОБОЗНАЧЕНИЕ-индикатора-выдающего-ПРИЗНАК l в окружении E, составленном из окружения E1 и участка L, определяется следующим образом:

Если L соответствует '?ОПИСАНИЯМ ?МЕТКАМ', в которые заложено {1.1.4.1.c} это 'ОБОЗНАЧЕНИЕ для ПРИЗНАКА',

то W – значение или сцена, если они существуют, доступные для 'ОБОЗНАЧЕНИЯ для ПРИЗНАКА' внутри L, и не определено в противном случае;

иначе W – выдача этого l в E1.

{Рассмотрим замкнутое-предложение, содержащее другое такое же:

```
начало прим блок 1 прим
  цел i = 421, цел a := 5, проц p = пуст : печ (a);
  начало прим блок 2 прим
    вещ a; a := i; p
  конец
конец
```

К тому времени, когда в ходе исполнения встретится a := i, будут созданы два новых окружения, по одному для каждого блока.

Сначала идет поиск определяющего-идентификатора i в E2, младшем из них, и, поскольку он не найдется там, то начнется поиск (успешный) в старшем окружении в E1. Участок этого E1 соответствует 'букве и лат для целого букве a для имени целого букве пэ лат для процедуры вырабатывающей пустое значение'. Следовательно, выдачей данного использующего-идентификатора i будет значение 421, приписанное (a) 'букве и лат для целого внутри участка окружения E1. Однако выдача идентификатора a в a := i найдется в участке окружения E2.

Когда вызывается (5.4.3.2.b) процедура *p*, ее основа исполняется в некотором окружении E3, устанавливаемом вокруг E1, но по E2 (3.2.2.b). Это означает, что в отношении области действия E3 младше E2, а E1 – составляющее окружение этого E3. Когда *a* должно печататься, оно является выдачей идентификатора-выдающего-имя-целого, описанного во внешнем из имеющихся блоков, и эта выдача есть 5.

Таким образом, смысл индикатора, используемого, но не описанного внутри процедуры, определяется контекстом, в котором эта процедура была создана, а не тем, в котором она вызывается.}

5. ОСНОВЫ

{Основы используются для программирования примитивных действий или превращения в единичные компоненты больших конструктов из разд. 3.

Приведенные-ПОНЯТИЯ, но не доопределения, являются результатом приведений (разд. 6); в случае ЗАКРЫТЫХ-предложений любые требуемые приведения осуществляются внутри них.

Из задаваемого ниже синтаксиса следует, например, что текст из отчет + „конец" разбирается как (текст из отчет) + „конец", поскольку выборка есть 'ВТОРИЧНОЕ', а формула – 'ТРЕТИЧНОЕ'.}

5.1. Синтаксис

A) ОСНОВА {32d} :: приведенное присваивание {521a};
приведенное отношение одноименности {522a};
приведенный текст процедуры 541a, b ; переход {544a} :
пропуск { 522a} ; ТРЕТИЧНОЕ { B} .

B) ТРЕТИЧНОЕ {A, 521b, 522a } :: псевдоимя { 524a} ;
приведенная АРНАЯ формула { 542a, b} ; ВТОРИЧНОЕ { C} .

C) ВТОРИЧНОЕ {B, 531a, 542c} :: приведенная выборка {531a} ;
приведенный ЛОКАЛИЗУЮЩИЙ генератор { 523a} ;
ПЕРВИЧНОЕ {D} .

D) ПЕРВИЧНОЕ {C, 532a, 543a } :: приведенная вырезка { 532a} :
приведенный вызов { 551a };
приведенное изображаемое { 80a} ;
приведенное ядро { 551a} ;
приведенный текст формата {A341a} ;
приведенный использующий СЛОВО идентификатор {48b} :
ЗАКРЫТОЕ предложение { 31a, 33a, c, d, e, 34a, 35a}.

{Гиперправила для 'приведенной ФОРМЫ ПРИВОДИМО выдающей ЗНАЧЕНИЕ', данные в 6.1.1.a, b, c, d и e, служат входами в синтаксис приведений. Когда этот синтаксис приведений запрашивается для какой-то 'приведенной ФОРМЫ ПРИВОДИМО выдающей ЗНАЧЕНИЕ', он в конце концов возвратится (, исключая тупики,) к некоторому правилу для 'ФОРМЫ выдающей ЗНАЧЕНИЕ1' в данном разделе. Именно на эти правила даны ссылки в метаправилах, перечисленных выше. Синтаксис приведений просто преобразует 'ЗНАЧЕНИЕ' в 'ЗНАЧЕНИЕ1' для семантики; в это время не порождается никакой другой видимый наследник. }

a)* доопределение ДЕЙСТВУЮЩЕЕ:
переход ДЕЙСТВУЮЩИЙ {544a} ;
пропуск ДЕЙСТВУЮЩИЙ {552};
псевдоимя ДЕЙСТВУЮЩЕЕ {524a} .

{Видом доопределения всегда является апостериорный вид, требуемый контекстом; выдача доопределения приемлема для этого вида. Поскольку любой вид легко получить таким образом, никакие приведения здесь не разрешены. }

5.2. Основы, связанные с именами

{ Именам можно присваивать (5.2.1), их можно сравнивать с другими именами (5.2.2) и создавать (5.2.3). }

5.2.1. *Присваивания*

{ В присваиваниях значение „присваивается" имени. Например, в x := 3.14 выдаваемое источником 3.14 вещественное число присваивается имени, выдаваемому получателем x. }

5.2.1.1. Синтаксис.

a) присваивание в СРЕДЕ выдающее ИМЯ ВИДА { 5A } :
получатель выдающий ИМЯ ВИДА в СРЕДЕ { b }.
знак присвоить { 94c } ,
источник вида ВИД в СРЕДЕ { c }.

b) получатель выдающий ИМЯ ВИДА в СРЕДЕ { a } :
ТРЕТИЧНОЕ в СРЕДЕ мягко выдающее ИМЯ ВИДА { 5B }.

c) источник вида ВИД1 в СРЕДЕ { a, 44d } :
основа в СРЕДЕ сильно выдающая ВИД2 { 32d }
если ВИД2 фиксирует ВИД1 { 47a, b, c, - }.

{ Примеры:
a) x := 3.14 b) x
c) 3.14 }

5.2.1.2. Семантика.

a) Всякое присваивание A исполняется следующим образом:

- пусть N и W – { совместные } выдачи { имя и некоторое другое значение } получателя и источника этого A;
- W присваивается { b } N:
- выдачей A служит N.

b) Значение W „присваивается" имени N, видом которого является некоторое 'ИМЯ ВИДА', следующим образом:

Требуется, чтобы
- N не было псевдоименем и
- W по области действия не было младше N;

Случай A: 'ВИД' есть 'структура содержащая !ПОЛЯ в себе':
Для каждого 'СЛОВА', выбирающего поле в W,
- это поле присваивается подымени, выбираемому по 'СЛОВУ' в N;

Случай B: 'ВИД' есть 'МАССИВ из ВИДА1':
- пусть V – { старое } значение, именуемое N;
- требуется, чтобы паспорта W и V были идентичны;

Для каждого индекса I, выбирающего элемент в W,
- этот элемент присваивается подымени, выбираемому по I в N;

Случай C: 'ВИД' есть 'подвижный МАССИВ из ВИДА1':
- пусть V – { старое } значение, именуемое N;
- N начинает именовать массив, составленный из

(i) паспорта значения W,

(ii) вариантов {4.4.2.c} некоторого {, возможно скрытого,} элемента значения V;

- N снабжается подыменами {2.1.3.1.g} ;

Для каждого индекса l, выбирающего элемент в W,

- этот элемент присваивается подымени, выбираемому по l в N;

Остальные случаи {, например, если 'ВИД' есть 'ПРОСТОЕ' или некоторый 'ПРЕДСТАВИТЕЛЬ'} :

- N начинает именовать {2.1.3.2a} W

{ Если дано

подв [1 : 0] [1 : 3] цел подвфикс,

то наличие скрытого элемента {2.1.3.4.c} гарантирует, что смысл присваивания подвфикс := лок [1 : 1] [1 : 3] цел вполне определен, в то время как смысл присваивания подвфикс := лок [1 : 1] [1 : 4] цел не определен, так как граничные пары по второму измерению различны.}

5.2.2. *Отношения одноименности*

{ Отношения-одноименности могут использоваться, чтобы узнать, совпадают ли два имени одного и того же вида.

Например, после присваивания проект := („абв", нил) отношение-одноименности следующая из проект :=: имя книга (нил) выдает значение истина. Однако следующая из проект :=: нил выдает ложь, поскольку оно эквивалентно следующая из проект :=: имя имени книга (нил), здесь выдача ТРЕТИЧНОГО следующая из проект есть безо всяких приведений имя, именующее второе поле структуры, именуемой значением проект и, следовательно, не являющейся псевдоименем.}

5.2.2.1. Синтаксис.

a) отношение одноименности в СРЕДЕ
выдающее логическое {5A} :
если мягко уравнивает ПРИВОДИМО1 и ПРИВОДИМО2 {32f} ,
ТРЕТИЧНОЕ1 в СРЕДЕ
ПРИВОДИМО1 выдающее имя ВИД {5B} ,
сравнитель имен {b},
ТРЕТИЧНОЕ2 в СРЕДЕ
ПРИВОДИМО2 выдающее имя ВИД {5B}.

b) сравнитель имен {a} : знак есть {94f};
знак не есть {94f}.

{ Примеры:

a) следующая из проект :=: имя книга (нил)

b) :=: • :≠:}

{ Данным синтаксисом не порождается a1 [i] :=: a1 [j]. Тем самым предотвращается сравнение временных имен (2.1.3.6.c) посредством отношения-одноименности. }

5.2.2.2. Семантика.

Выдача W отношения-одноименности 1 определяется следующим образом:

• пусть N1 и N2 – {совместные} выдачи ТРЕТИЧНЫХ этого l;

Случай А: Знаком сравнителя-имен этого l является знак-есть:

• W – истина, если {имя} N1 есть то же, что и N2, и ложь в противном случае;

Случай В: Знаком сравнителя-имен этого l является знак-не-есть:

• W – истина, если N1 не есть то же, что и N2, и ложь в противном случае.

5.2.3. *Генераторы*

{ Исполнение генератора, например лок вещ в xx := лок вещ := 3.14, или задания-генератора, например [1: n] лит в [1 : n] лит u, v; , включает создание имени, т.е. отведение места в памяти.

Использование локального-генератора предполагает (в большинстве реализаций) отведение памяти в динамическом стеке, тогда как глабальные-генераторы предполагают отведение памяти в другой области, называемой „кучей", в которой для управления памятью можно применять технику, называемую обычно „сборкой мусора". В силу меньшей эффективности последнего режима локальные-генераторы лучше, так что в задании-генераторов описаний переменных можно опускать только лок.}

5.2.3.1. Синтаксис.

{ ЛОКАЛИЗУЮЩИЙ :: локальный; глобальный; первичный.}

a) ЛОКАЛИЗУЮЩИЙ генератор в СРЕДЕ
выдающий имя ВИДА {5C}:
знак ЛОКАЛИЗУЮЩИЙ {94d, -},
фактический описатель ВИДА в СРЕДЕ {46a}.

b) задание ЛОКАЛИЗУЮЩЕГО генератора
в СРЕДЕ ВЫДАЮЩЕГО ИМЯ ПРОВИДА {44e}:
знак ЛОКАЛИЗУЮЩИЙ {94d, -},
фактический описатель ПРОВИДА в СРЕДЕ {44b, 46a};
если (ЛОКАЛИЗУЮЩИЙ) есть (локальный),
фактический описатель ПРОВИДА в СРЕДЕ {44b, 46a};

{Примеры:

a) лок вещ b) лок вещ • вещ}

{Не существует представления для символа-первичный (см. 9.4.a).}

5.2.3.2. Семантика.

а) Выдача W ЛОКАЛИЗУЮЩЕГО-генератора или задания-ЛОКАЛИЗУЮЩЕГО-генератора G в окружении E определяется следующим образом:

• W – вновь созданное имя, начинающее именовать {2.1.3.2.a} выдачу в E фактического-описателя {4.4.2.d, 4.6.2.a} из G;

• W по области действия та же, что и окружение E1, определяемое следующим образом:

Случай А: 'ЛОКАЛИЗУЮЩИЙ' есть 'локальный':

• E1 – „локализующее окружение" {b}, доступное из окружения E;

Случай В: 'ЛОКАЛИЗУЮЩИЙ' есть 'глобальный':

· E1 является {первым окружением, созданным в ходе исполнения текущей собственно-программы, являющимся} таким, что:

(i) первичное окружение {2.2.2.a} – это окружение окружения окружения этого E1 {именно так!} и

(ii) E1 есть E или старше E;

Случай C: 'ЛОКАЛИЗУЮЩИЙ' есть 'первичный':

· E1 – первичное окружение;

· если W – составное имя {2.1.3.2.b}, то W снабжается подыменами {2.1.3.3.e, 2.1.3.4.g}.

{Единственное место, где встречаются примеры первичных-генераторов, – это стандартное- и системное-вступления (10.3.1.1.h, 10.3.1.4.b, n, o, 10.4.1.a).

Когда G есть задание-генератора-выдающего-имя-процедуры, вид выдачи W не существен.}

b) „Локализующим окружением", доступным из окружения E, является окружение E1, определяемое следующим образом:

Если E – „нелокализующее" {3.2.2.b},

то E1 – локализующее окружение, доступное из окружения этого E;

иначе E1 есть E.

{Окружение является нелокализующим, если оно устанавливается согласно последовательному-предложению или выясняющему-предложению, которое не содержит составляющего описания-вида, -идентификатора или -операции, либо согласно заглавию-цикла (3.5.1.b) или спецификации (3.4.1.j, k).}

5.2.4. *Псевдоимена*

5.2.4.1. Синтаксис.

a) псевдоимя в СРЕДЕ сильно выдающее имя ВИДА {5B}: знак нил {94f}.

{Пример:

a) нил{Нил – это аналог нуля для имен}}

5.2.4.2. Семантика.

Выдачей псевдоимени является псевдоимя.

5.3. Основы, связанные с составными значениями.

{Поля структур можно получить посредством выборок (5.3.1), а элементы массивов – посредством вырезок (5.3.2); кроме того, соответствующие действия определены и над составными именами.}

5.3.1. *Выборки*

{Выборка выбирает поле из структуры или (если это – „выборка из массива") некоторый массив из массива, элементами которого служат структуры. Например, вч из z выбирает первое вещественное поле (называемое обычно вещественной частью) выдачи этого z. Если *z* выдает имя, то вч из z также выдает имя, но если g выдает комплексное значение, то вч из g выдает вещественное значение, а не именующее его имя.}

5.3.1.1. Синтаксис.

A) ?ИМЯ :: ИМЯ; ПУСТО.

B) ?ССЫЛКА НА :: ИМЯ; ИМЯ подвижного; ПУСТО.
{ИМЯ :: имя; временное имя.}

a) выборка в СРЕДЕ выдающая ?ИМЯ ВИДА1 {5C} :
использующий СЛОВО указатель поля
вида ВИД1 среди !ПОЛЕЙ {48d}, знак из {94f},
ВТОРИЧНОЕ в СРЕДЕ слабо выдающее ?ИМЯ
структуры содержащей !ПОЛЯ в себе {5C};
если (ВИД1) есть (МАССИВ из ВИДА2),
использующий СЛОВО указатель поля
вида ВИД2 среди !ПОЛЕЙ {49d}, знак из {94f},
ВТОРИЧНОЕ в СРЕДЕ слабо выдающее ?ССЫЛКУ
НА МАССИВ из структур содержащих !ПОЛЯ в себе {5C},
если (?ИМЯ) выводится из (?ССЫЛКИ НА) { b, c, -}.

b) ЕСЛИ (временное имя) выводится из
(ИМЕНИ подвижного) { a, 532a, 66a} : ЕСЛИ истина.

c) ЕСЛИ (?ИМЯ) выводится из
(?ИМЕНИ) { a, 532a, 66a } : ЕСЛИ истина.

{ Примеры:

a) вч из z • вч из z 1}

{ Вид вч из z начинается с 'имя', так как вид z начинается с него же.

Пример:

цел возраст := 7; ст (лог пол, цел возраст) жиль;
возраст из жиль := возраст;

Отметим, что получатель возраст из жиль выдает имя, так как жиль само выдает имя. После описания-тождества

ст (лог пол, цел возраст) джек = (истина, 9),

не может быть присваивания полю возраст из джек, поскольку джек не является переменной. }

5.3.1.2. Семантика.

Выдача W выборки s определяется следующим образом:

- пусть V будет выдачей ВТОРИЧНОГО выборки S;
- требуется, чтобы V {, если оно имя,} не было псевдоименем;
- W – значение, выбираемое в {2.1.3.3.a, e, 2.1.3.4.k}, или имя, генерируемое из { 2.1.3.4.1} V по указателю-поля этого S.

{ Выборка из имени, именующего структуру, выдает существующее подымя (2.1.3.3.e) этого имени. Имя, порожденное из имени, именующего массив, посредством выборки со ВТОРИЧНЫМ-выдающим-МАССИВ-из-ВИДА (как в вч из z 1), есть имя, которое может как быть, так и не быть вновь созданным для этой цели. }

5.3.2. *Вырезки*

{ Вырезки получаются посредством индексации, например x1 [i], вырезания, например x1 [2 : n], или и того и другого, например x2 [j : n, j]

или x2 [, k]. И индексация, и вырезание могут применяться только к ПЕРВИЧНЫМ, например к x1 или (p |x1|y1), но не к вч из z1. Значением вырезки может быть или один из элементов выдачи ее ПЕРВИЧНОГО, или подмножество таких элементов; например, x1 [i] – вещественное число из вектора вещественных чисел x1, x2 [i,] представляет собой i-ю строку матрицы x2, а x2 [, k] – ее k-й столбец.}

5.3.2.1. Синтаксис.

A) ?МАССИВ :: МАССИВ: ПУСТО.

a) вырезка в СРЕДЕ выдающая ?ИМЯ ВИДА1 {5D}:
ПЕРВИЧНОЕ в СРЕДЕ слабо выдающее
?ССЫЛКУ НА МАССИВ1 из ВИДА1 {5D},
индексирующий ОФОРМЛЕННЫЙ индексатор МАССИВА1
в СРЕДЕ оставляющий ПУСТО {b, c, -},
если (?ИМЯ) выводится из (?ССЫЛКИ НА) {531b, c, -};
если (ВИД1) есть (МАССИВ2 из ВИДА2),
ПЕРВИЧНОЕ в СРЕДЕ слабо выдающее
?ССЫЛКУ НА МАССИВ1 из ВИДА2 {5D},
индексирующий ОФОРМЛЕННЫЙ индексатор МАССИВА1
в СРЕДЕ оставляющий МАССИВ2 {b, d, -},
если (?ИМЯ) выводится из (?ССЫЛКИ НА) {531b, c, -}.
{МАССИВ :: вектор; МАССИВ векторов.}

b) индексатор вектора МАССИВОВ в СРЕДЕ
оставляющий ?МАССИВ1 ?МАССИВОВ2 {a, b}:
индексатор вектора в СРЕДЕ оставляющий ?МАССИВ1
{c, d, -}, знак a также {94f},
индексатор МАССИВА в СРЕДЕ
оставляющий ?МАССИВ2 {b, c, d, -}.

c) индексатор вектора в СРЕДЕ оставляющий ПУСТО {a, b}:
индекс в СРЕДЕ {e}.

d) индексатор вектора в СРЕДЕ оставляющий вектор {a, b}:
отрезок в СРЕДЕ {f};
возможная сдвинутая нижняя граница в СРЕДЕ {g}.

e) индекс в СРЕДЕ {c}:
основа в СРЕДЕ раскрыто выдающая целое {32d}.

f) отрезок в СРЕДЕ {c}:
возможная нижняя граница в СРЕДЕ {46m},
знак вплоть до {94f},
возможная верхняя граница в СРЕДЕ {46n}.
возможная сдвинутая нижняя граница в СРЕДЕ {g}.

g) сдвинутая нижняя граница в СРЕДЕ {d, f}.
знак с {94f},
нижняя граница в СРЕДЕ {46m}.

h)* ограничение : индекс в СРЕДЕ {e}; отрезок в СРЕДЕ {f};
возможная сдвинутая нижняя граница в СРЕДЕ {g}.

i)* индексатор:

индексатор МАССИВА в СРЕДЕ
оставляющий ?МАССИВ {b, c, d}.

j)* грань : индекс в СРЕДЕ {e}; нижняя граница в СРЕДЕ {46m};
верхняя граница в СРЕДЕ {46n};
сдвинутая нижняя граница в СРЕДЕ {g}.

{Примеры:

a) x2 [i, j] · x2 [, j]

b) 1 : 2, j (в x2 [1 : 2, j]) · i, j (в x2 [i, j])

c) j (в x2 [1 : 2, j]

d) 1 : 2 · @ 0 (в x1 [@ 0])

e) j

f) 1 : 2@0

g) @ 0}

{Индекс уменьшает число измерений на одно, а отрезок не меняет его. В правиле (а) 'МАССИВ' отражает число ограничений в вырезке, а 'МАССИВ2' – число тех из них, которые являются отрезками или возможными-сдвинутыми-нижними-границами.

Если значение, из которого должна быть сделана вырезка, есть имя, то выдачей этой вырезки также будет имя. Кроме того, если вид исходного имени есть 'имя подвижного МАССИВА из ВИДА', то эта выдача будет временным именем (см. 2.1.3.6.c).}

5.3.2.2. Семантика.

а) Выдача W вырезки S определяется следующим образом:

- пусть V и $(l_1, ..., l_n)$ – {совместные выдачи} ПЕРВИЧНОГО вырезки S и индексатора {b} из S;
- требуется, чтобы V {, если оно имя,} не было псевдоименем;
- пусть $((r1, s1), ..., (r_n, s_n))$ – паспорт выдачи V или значения, именуемого V;

Для i = 1, ..., n

Случай А: l_i – целое число:

- требуется, чтобы $r_i \leqslant l_i \leqslant s_i$;

Случай В: l_i – тройка (l, u, l'):

- пусть L будет r_i, если l отсутствует, и l в противном случае;
- пусть U будет s_i, если u отсутствует, и u в противном случае;
- требуется, чтобы $r_i \leqslant L$ и $U \leqslant s_i$;
- пусть D будет 0, если l' отсутствует и L – l' в противном случае; {D – это число, которое следует вычесть из L для того, чтобы получить сдвинутую нижнюю границу;}
- l_i заменяется на (L, U, D);
- W – значение, выбираемое в {2.1.3.4.a, g, i}, или имя, генерируемое из {2.1.3.4.j} V по $(l_1, ..., l_n)$.

b) Выдача индексатора l вырезки S есть отрезок {2.1.3.4.h} или индекс {2.1.3.4.a} $(l_1, ..., l_n)$, определяемый следующим образом:

- составляющие грани вырезки S исполняются совместно;

Для i = 1, ..., n, где n – число составляющих ограничений вырезки S,

Случай А: i-е ограничение есть индекс:

- l_i – {целое число, которое есть} выдача этого индекса;

Случай B: i-е ограничение есть отрезок T:

- l_i – тройка (l, u, l′), где
- l – выдача составляющей нижней-границы из T, если она есть, а иначе отсутствует,
- u – выдача составляющей верхней-границы из T, если она есть, а иначе отсутствует,
- l′ – выдача составляющей сдвинутой-нижней границы из T, если она есть, а иначе l:

Случай C: i-е ограничение есть возможная-сдвинутая-нижняя-граница N:

- l_i – тройка (отсутствует, отсутствует, l′),

где

- l′ – выдача сдвинутой-нижней-границы из N, если она есть, а иначе отсутствует.

{ Отметим, что если $(l_1, ..., l_n)$ не содержит троек, то это индекс, выбирающий один элемент; в противном случае это отрезок, выбирающий подмножество элементов. }

{ Вырезка из имени, именующего массив, выдает существующее подымя (2.1.3.4.j) этого имени, если все составляющие ограничения этой вырезки являются индексами. В противном случае она выдает генерируемое имя, которое может как быть, так и не быть вновь созданным для этой цели. Следовательно, выдача отношения-одноименности x1 [1 : 2] :=: x1 [1 : 2] не определена, хотя x1 [1] :=: x1 [1] должно всегда выдавать истина. }

{ Различные возможные границы в выдаче вырезок иллюстрируются следующими примерами, для каждого из которых показан паспорт значения, именуемого соответствующей выдачей:

```
[0 : 9,2 : 11] цел i 3;
i3 [1, 3 : 10 @ 3] ↓ [↖(3, 10))↓;
i3 [1, 3 :10] ↓ ((1, 8))↓;
i3 [1, 3 :] ↓ ((1, 9))↓;
i3 [1, :] ↓ ((1, 10))↓;
i3 [1, ] ↓ (2, 11))↓;
i3 [, 2] ↓ (0,9))↓.
```

}

5.4. Основы, связанные с процедурами

{ Процедуры создаются из текстов-процедур (5.4.1) или переходов (5.4.4); их можно „вызвать" при помощи вызовов (5.4.3), формул (5.4.2) или посредством распроцедуривания (6.3). }

5.4.1. *Тексты процедур*

{ Текст-процедуры всегда содержит формальный-описатель, специфицирующий вид результата, и знак-признак-процедуры, а именно двоеточие. Справа от этого двоеточия располагается основа, определяющая вычисления, которые надо осуществить, когда вызывается данная процедура. Если процедура имеет параметры, то слева от упомянутого формального-описателя помещается задание-аргументов, содержащее различные требуемые формальные-параметры.

Примеры:

пуст : печ (x);
(имя вещ a, вещ b) лог : (a < b | a := b; истина | ложь).}

5.4.1.1. Синтаксис.

a) текст процедуры в СРЕДЕ1 выдающий
процедуру вырабатывающую ЗНАЧЕНИЕ {44d, 5A}:
формальный описатель ЗНАЧЕНИЯ в СРЕДЕ1 {46b},
знак признак процедуры {94f},
основа в СРЕДЕ1 сильно выдающая ЗНАЧЕНИЕ {32d}

b) текст процедуры в СРЕДЕ1 выдающий процедуру
с !ПАРАМЕТРАМИ вырабатывающую ЗНАЧЕНИЕ {44d, 5A}:
упакованное кратким
определяющее новые !ОПИСАНИЯ2 задание
аргументов в СРЕДЕ1 с новыми !ОПИСАНИЯМИ2 {e},
если !ОПИСАНИЯ2 оказались !ПАРАМЕТРАМИ {c, d, -},
формальный описатель ЗНАЧЕНИЯ в СРЕДЕ1 {46b},
знак признак процедуры {94f},
основа в СРЕДЕ1 с новыми !ОПИСАНИЯМИ2
сильно выдающая ЗНАЧЕНИЕ {32d}.

c) ЕСЛИ !ОПИСАНИЯ ОПИСАНИЕ оказались
!ПАРАМЕТРАМИ ПАРАМЕТРОМ {b, c}:
ЕСЛИ !ОПИСАНИЯ оказались !ПАРАМЕТРАМИ {c, d, -}
и ОПИСАНИЕ оказалось ПАРАМЕТРОМ {d, -}.
{ПАРАМЕТР :: параметр вида ВИД.}

d) ЕСЛИ СЛОВО для ВИДА оказалось параметром вида
ВИД {b, c}:
ЕСЛИ истина.

e) определяющее новые !ОПИСАНИЯ2
задание аргументов в СРЕДЕ2 {b, e, 34j}:
формальный описатель ВИДА в СРЕДЕ {46b},
групповое определение параметров вида ВИД
через !ОПИСАНИЯ2 в СРЕДЕ2 {41b, c};
если (!ОПИСАНИЯ2) есть (!ОПИСАНИЯ3 !ОПИСАНИЯ4),
формальный описатель ВИДА в СРЕДЕ2 {46b},
групповое определение параметров вида ВИД
через !ОПИСАНИЯ3 в СРЕДЕ2 {41b, c},
знак а также {94f},
определяющее новые !ОПИСАНИЯ4
задание аргументов в СРЕДЕ2 {e}.

f) определение параметра вида ВИД
через СЛОВО2 для ВИДА в СРЕДЕ2 {41c}:
определяющий СЛОВО2
идентификатор в СРЕДЕ2 выдающий ВИД {48a}.

g)* формальный параметр вида ВИД:
определение параметра вида ВИД через
СЛОВО для ВИДА в СРЕДЕ {f}.

{Примеры:

a) вещ : псч × 10 b) (лог a, b) лог : (a | b | ложь)

e) лог a, b • лог a, лог b f) a}

5.4.1.2. Семантика.

Выдачей текста-процедуры T в окружении E является процедура, составленная из

(i) T и

(ii) окружения, необходимого для {7.2.2.c} T в E.

5.4.2. *Формулы*

{Формулы бывают бинарные и унарные, например x + i или абс x. Порядок исполнения формулы определяется приоритетами ее обозначений-операций; первыми исполняются унарные формулы, затем – бинарные, от высшего приоритета к низшему.}

5.4.2.1. Синтаксис.

A) БИНАРНОЕ :: приоритета ПРИОРИТЕТ.

B) УНАРНОЕ :: приоритета III III III I.

C) АРНОЕ :: БИНАРНОЕ; УНАРНОЕ.

D) ?НОМЕР :: НОМЕР; ПУСТО.

a) БИНАРНАЯ формула в СРЕДЕ выдающая ЗНАЧЕНИЕ {c, 5B}
БИНАРНЫЙ ?НОМЕРА операнд выдающий ВИД1 в СРЕДЕ {c, -},
использующее ИНФИКС обозначение операции в СРЕДЕ
выдающее процедуру с параметром вида ВИД1
параметром вида ВИД2 вырабатывающую ЗНАЧЕНИЕ {48b},
если ИНФИКС для БИНАРНОГО
идентифицирован в СРЕДЕ {72a},
БИНАРНЫЙ НОМЕРА операнд выдающий ВИД2 в СРЕДЕ {c, -}.

b) УНАРНАЯ формула в СРЕДЕ выдающая ЗНАЧЕНИЕ {c, 5B}:
использующее ПРЕФИКС обозначение операции в СРЕДЕ
выдающее процедуру с параметром вида ВИД
вырабатывающую ЗНАЧЕНИЕ {48b},
УНАРНЫЙ операнд выдающий ВИД в СРЕДЕ {c}.

c) АРНЫЙ операнд выдающий ВИД в СРЕДЕ {a, b}:
приведенная АРНАЯ формула в СРЕДЕ {a, b}
крепко выдающая ВИД {61b}:
если (АРНОЕ) есть (УНАРНОЕ),
ВТОРИЧНОЕ в СРЕДЕ крепко выдающее ВИД {5C}.

b)* формула выдающая ЗНАЧЕНИЕ:
АРНАЯ формула в СРЕДЕ выдающая ЗНАЧЕНИЕ {a, b}

e)* обозначение ИНФИКСОМ бинарной операции
выдающее ДВУМЕСТНУЮ:
ПРИМЕНЯЮЩЕЕ ИНФИКС обозначение операции
в СРЕДЕ выдающее ДВУМЕСТНУЮ {48a, b}.

f)* обозначение ПРЕФИКСОМ унарной операции
выдающее ОДНОМЕСТНУЮ:
ПРИМЕНЯЮЩЕЕ ПРЕФИКС обозначение операции

в СРЕДЕ выдающее ОДНОМЕСТНУЮ {48a, b}.

g)* операнд вида ВИД:

АРНЫЙ операнд выдающий ВИД в СРЕДЕ {c}.

{ Примеры:

a) – x + 1 b) – x

c) – x · 1 }

5.4.2.2. Семантика.

Выдача W формулы F в окружении E определяется следующим образом:

- пусть R – процедура, выдаваемая обозначением-операции формулы F;
- пусть V_1, ..., V_n {n есть 1 или 2} – {совместные} выдачи операндов формулы F в {локализующем, см. 3.2.2.b,} окружении E1, устанавливаемом вокруг E;
- W – выдача вызова {5.4.3.2.b} процедуры R в E1 с V_1, ..., V_n;
- требуется, чтобы W по области действия не была младше E.

{ Отметим, что a ↑ b не в точности то же, что и a^b в обычных обозначениях; действительно, как (–1 ↑ 2 + 4 = 5), так и (4 – 1 ↑ 2 = 3) имеют значение истина, поскольку первый, символ-минус является обозначением-унарной-операции, а второй – обозначением-бинарной-операции. }

5.4.3. *Вызовы*

{ Вызовы используются для задания исполнения параметризованных процедур с фактическими-параметрами.

Примеры:

sin (x) · (p | sin| cos) (x). }

5.4.3.1. Синтаксис.

a) вызов в СРЕДЕ выдающий ЗНАЧЕНИЕ {5D}:
ПЕРВИЧНОЕ в СРЕДЕ раскрыто выдающее процедуру
с !ПАРАМЕТРАМИ вырабатывающую ЗНАЧЕНИЕ {5D},
упакованные кратким
фактические !ПАРАМЕТРЫ в СРЕДЕ {b, c}.

b) фактические !ПАРАМЕТРЫ ПАРАМЕТР в СРЕДЕ {a, b}:
фактические !ПАРАМЕТРЫ в СРЕДЕ {b, c}.
знак a также {94f},
фактический ПАРАМЕТР в СРЕДЕ {c}.

c) фактический параметр вида ВИД в СРЕДЕ {a, b}:
основа в СРЕДЕ сильно выдающая ВИД {32d}.

{ Примеры:

a) вывод (станд вывод, x) (см. 10.3.3.1.a)

b) станд вывод, x c) x }

5.4.3.2. Семантика.

a) Выдача W вызова C в окружении E определяется следующим образом:

- пусть {процедура} R и V_1, ..., V_n будут {совместными} выдачами ПЕРВИЧНОГО из C в E и составляющих фактических-параметров этого C в {локализующем, см. 3.2.2.b,} окружении E1, устанавливаемом вокруг E;

• W – выдача вызова {b} процедуры R в E1 с V_1, ..., V_n;
• требуется, чтобы W по области действия не была младше E.

b) Выдача W „вызова" процедуры R в окружении E1, возможно со значениями {параметров} V_1, ..., V_n, определяется следующим образом:
• пусть E2 – окружение, устанавливаемое {3.2.2.b} по E1 вокруг окружения процедуры R согласно заданию-аргументов упакованного-задания-аргументов, если оно есть, текста-процедуры из R со значениями, если они есть, V_1, ..., V_n;
• W – выдача в E2 основы текста-процедуры из R.

{ Рассмотрим последовательное-предложение
проц замельзон = (цел n, проц (цел) вещ f) вещ:
начало длин вещ s := длин 0;
для i до n цк s +:= удл f (i) ↑ 2 кц;
укр длин корень (s)
конец;
замельзон (m, (цел j) вещ : x1 [j]).
В этом контексте последний вызов дает тот же эффект, что и ядро:
вещ:
цел n = m, проц (цел) вещ f = (цел j) вещ : x1 [j];
начало длин вещ s := длин 0;
для i до n цк s +:= удл f (i) ↑ 2 кц;
укр длин корень (s)
конец).

Таким образом, передача фактических-параметров аналогична исполнению описаний-тождества (4.4.2.a); см. также установление (3.2.2.b) и приписывание (4.8.2.a). }

5.4.4. *Переходы*

{ Переход может прекратить исполнение кортежа и заставить некоторый другой помеченный кортеж исполняться вместо него.

Примеры:

у := если x ⩾ 0 то корень (x) иначе на принстон все •
на сен пьер де шартрез.

С другой стороны, если контекст требует вида 'процедура вырабатывающая ЗНАЧЕНИЕ', то вместо этого выдается процедура, основой которой служит данный переход, как в проц пуст m := на норт бервик. }

5.4.4.1. Синтаксис.

a) переход в СРЕДЕ сильно выдающий ЗНАЧЕНИЕ {5A}:
возможное указание {b},
использующий СЛОВО идентификатор в СРЕДЕ
выдающий метку {48b}.

b) указание {a}: знак на ОФОРМЛЕННЫЙ {94f, -};
знак иди ОФОРМЛЕННЫЙ {94f, -},
символ верхний предел ОФОРМЛЕННЫЙ {94g, -}.

{ Примеры:

а) на коотвейк • go to варшава • зандвоорт

b) на · go to }

5.4.4.2. Семантика.

Переход-в-СРЕДЕ-выдающий-ЗНАЧЕНИЕ J в окружении E исполняется следующим образом:

· пусть сцена, выдаваемая в E идентификатором-метки этого J, составлена из кортежа S2 и окружения E1;

Случай A: 'ЗНАЧЕНИЕ' не является никакой 'процедурой вырабатывающей ЗНАЧЕНИЕ1':

· пусть S1 – кортеж наименьшего {1.1.3.2.g} последовательного-предложения, содержащего S2;

· исполнение S1 в E1 или любого кортежа, исполняемого вместо него в E1, прекращается {2.1.4.3.e};

· S2 исполняется в E1 „вместо" S1 в E1;

Случай B: 'ЗНАЧЕНИЕ' есть 'процедура вырабатывающая ЗНАЧЕНИЕ1':

· J в E {завершается и} выдает процедуру, составленную из:

(i) нового текста-процедуры-в-СРЕДЕ-выдающего-ЗНАЧЕНИЕ, основа которого подобна {1.1.3.2.k} этому J,

(ii) окружения E1.

5.5. Основы, связанные со значениями любого вида

5.5.1. *Ядра*

{ Ядра можно использовать, чтобы обеспечить сильную позицию, например, имя вещ (xx) в имя вещ (xx) := 1, имя книга (нил) в следующая из проект :=: имя книга (нил) и строк (p|c|r) в s +:= строк (p|c|r). }

5.5.1.1. Синтаксис.

а) ядро в СРЕДЕ выдающее ЗНАЧЕНИЕ {5D}:
формальный описатель ЗНАЧЕНИЯ в СРЕДЕ {46b},
ЗАКРЫТОЕ предложение в СРЕДЕ сильно выдающее
ЗНАЧЕНИЕ {31a, 33a, c, d, e, 34a, 35a, -}

{ Пример:
а) имя книга (нил) }

{ Выдачей ядра является, в силу предысполнения (2.1.4.1.c), выдача его ЗАКРЫТОГО-предложения. }

5.5.2. *Пропуски*

5.5.2.1. Синтаксис.

а) пропуск в СРЕДЕ сильно выдающий ЗНАЧЕНИЕ {5A}:
знак пропуск {94f}.

{ Пример:
а) пропуск }

5.5.2.2. Семантика.

Выдачей пропуска является некоторое {неопределенное} значение, по области действия такое же, как и первичное окружение.

{ Вид выдачи пропуска-выдающего-ЗНАЧЕНИЕ есть 'ЗНАЧЕНИЕ'. Пропуск-выдающий-пустое значение служит пустым оператором и его можно,

например, использовать после метки, которая отмечает конец последовательного-предложения. }

6. ПРИВЕДЕНИЕ

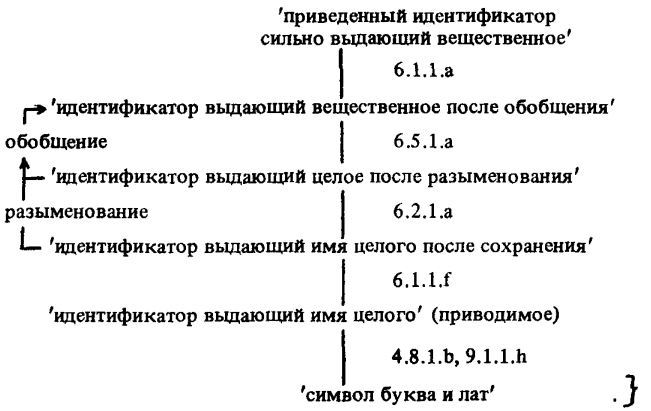
{ Приведения порождают приводимое из приведенного в соответствии с тремя критериями: априорного вида приводимого до применения какого-либо приведения, апостериорного вида приведенного, требуемого после выполнения приведений, и синтаксической позиции, или „приводимости" этого приведенного. Приведения могут применяться друг к другу последовательно.

Существует шесть возможных приведений, называемых „распроцедуривание", „разыменование", „объединение", „обобщение", „векторизация" и „опустошение". Каждое приведение, исключая „объединение", оказывает соответствующее динамическое влияние на связанные с ним значения. Следовательно, при помощи приведений неявно можно запрограммировать ряд примитивных действий. }

6.1. Приведенные

{ Всякое приведенное есть конструкт, дерево порождения которого может начинаться с последовательности приведений, оканчивающейся приводимым. Порядок (завершения) исполнения этих приведений идет поэтому от приводимого к приведенному (, что и определило выбор этих парапонятий). Например, і в вещ (і) является приведенным, дерево порождения которого включает 'после обобщения' и 'после разыменования' именно в этом порядке на пути от приведенного к приводимому. Разыменование должно завершиться прежде, чем произойдет обобщение.

Соответствующее дерево порождения (с опущенным 'в СРЕДЕ', 'использующий СЛОВО' и невидимыми поддеревьями) выглядит так:



}

6.1.1. *Синтаксис*

A) УСИЛЕНИЕ {a, 66a} :: УКРЕПЛЕНИЕ {B}; обобщение {65a, b, c, d};
векторизация {66a}; опустошение {67a, b}.

B) УКРЕПЛЕНИЕ { A, b } :: РАСКРЫТИЕ { C }; объединение {64a}.

C) РАСКРЫТИЕ { B, c, d, 62a, 63a, 64a, 65a, b, c, d } ::
сохранение { f }; разыменование {62a};
распроцедуривание {63a}.

D) СМЯГЧЕНИЕ {e, 63b} :: сохранение {f};
мягкое распроцедуривание {63b}.

E) ФОРМА : : РАСКРЫВАЕМОЕ; ПРЯМОЕ.

F) РАСКРЫВАЕМОЕ :: выборка в СРЕДЕ; вырезка в СРЕДЕ;
вызов в СРЕДЕ; текст процедуры в СРЕДЕ;
АРНАЯ формула в СРЕДЕ;
использующий СЛОВО идентификатор в СРЕДЕ.

G) ПРЯМОЕ :: присваивание в СРЕДЕ; ядро в СРЕДЕ;
отношение одноименности в СРЕДЕ; изображаемое в СРЕДЕ;
ЛОКАЛИЗУЮЩИЙ генератор в СРЕДЕ; текст формата в СРЕДЕ.

a) приведенная ФОРМА сильно
выдающая ЗНАЧЕНИЕ {5A, B, C, D, A341i}:
если (ФОРМА) есть (РАСКРЫВАЕМОЕ),
РАСКРЫВАЕМОЕ выдающее значение после УСИЛЕНИЯ {A};
если (ФОРМА) есть (ПРЯМОЕ),
ПРЯМОЕ выдающее ЗНАЧЕНИЕ после УСИЛЕНИЯ {A},
если неверно что (ЗНАЧЕНИЕ после УСИЛЕНИЯ)
есть (пустое значение после распроцедуривания).

b) приведенная ФОРМА крепко
выдающая ВИД {5A, B, C, D, 542c}:
ФОРМА выдающая ВИД после УКРЕПЛЕНИЯ {B}.

:) приведенная ФОРМА раскрыто
выдающая ЗНАЧЕНИЕ {5A, B, C, D}:
ФОРМА выдающая ЗНАЧЕНИЕ после РАСКРЫТИЯ {C}.

l) приведенная ФОРМА слабо выдающая
?ИМЯ СОСТАВНОГО {5A, B, C, D}:
ФОРМА выдающая ?ИМЯ СОСТАВНОГО после РАСКРЫТИЯ {C},
если неверно что (?ИМЯ) есть (ПУСТО)
и (РАСКРЫТИЕ) есть (разыменование)

) приведенная ФОРМА мягко выдающая ВИД {5A, B, C, D}:
ФОРМА выдающая ВИД после СМЯГЧЕНИЯ {D}.

ФОРМА выдающая ЗНАЧЕНИЕ
после сохранения {C, D, 67a, b}:
ФОРМА выдающая ЗНАЧЕНИЕ.

)* приведенное ПРИВОДИМО выдающее ЗНАЧЕНИЕ:
приведенная ФОРМА ПРИВОДИМО
выдающая ЗНАЧЕНИЕ {a, b, c, d, e}.

)* приводимое выдающее ЗНАЧЕНИЕ:

ФОРМА выдающая ЗНАЧЕНИЕ.

{Примеры:

a) 3.14 (в x := 3.14)

b) 3.14 (в x + 3.14)

c) sin (в sin (x))

d) x1 (в x1 [2] := 3.14)

e) x (в x := 3.14)}

{Ссылки на 'ФОРМУ выдающую ЗНАЧЕНИЕ' (правило f) содержатся в подразделе 5.1.A, B, C, D. Заметим, что 'ФОРМА выдающая ЗНАЧЕНИЕ' может быть тупиком. Тупики в этой главе не отмечаются.}

{Существуют пять сортов синтаксических позиций, а именно:

- „сильные" позиции, т.е. фактические-параметры, например x в sin (x), источники, например x в y := x, ЗАКТЫТЫЕ-предложения ядер, например (нил) в имя книга (нил), и операторы, например, y := x в (y := x; x := 0);

„крепкие" позиции, т.е. операнды, например x в x + y;

- „раскрытые" позиции, т.е. выясняющие-предложения, например x > 0 в (x > 0 | x | 0), грани, например i в x1 [i], и ПЕРВИЧНЫЕ вызовов, например sin в sin (x);
- „слабые" позиции, т.е. ВТОРИЧНЫЕ выборок и ПЕРВИЧНЫЕ вырезок, например x1 в x1 [i];
- „мягкие" позиции, т.е. получатели, например x в x := y, и одно из ТРЕТИЧНЫХ отношений-одноименности, например x в xx :=: x.

Сильные позиции появляются также при уравнивании (3.2.1.e).

В сильных позициях могут встретиться все шесть приведений, в крепких позициях запрещены векторизация, обобщение и опустошение, в раскрытых и слабых позициях, кроме того, запрещено объединение, а в мягких позициях разрешено только распроцедуривание. Однако ФОРМА-выдающая-СОСТАВНОЕ-после-разыменования не может быть прямым наследником приведенной-ФОРМЫ-слабо-выдающей-СОСТАВНОЕ (правило d), поскольку в противном случае x := x1 [i] было бы двусмысленным синтаксически (хотя в этом случае не семантически). Точно так же ПРЯМОЕ-выдающее-пустое-значение-после-распроцедуривания не может быть прямым наследником приведенного-ПРЯМОГО-сильно-выдающего-пустое-значение (правило a), так как в противном случае было бы двусмысленным (проц пусть энгельфриет; проц пуст рейпенс = пропуск: энгельфриет := рейпенс: пропуск).}

6.2. Разыменование

{Разыменование служит для получения значения, именуемого некоторым именем, как в x := y, где y выдает имя, именующее вещественное число, и именно это число присваивается имени, выдаваемом получателем x. Априорный вид y, рассматриваемого как приводимое, есть 'имя вещественного', а его апостериорный вид, когда y рассматривается как приведенное, есть 'вещественное'.

6.2.1. *Синтаксис*

a) ФОРМА выдающая ВИД1 после разыменования {61C}:
ФОРМА выдающая ИМЯ ВИДА2 после РАСКРЫТИЯ {61C},
если ВИД1 фиксирует ВИД2 {47a, b, c, -}.

{Пример:
a) х (в вещ (х))}

6.2.2. *Семантика*

Выдача W ФОРМЫ-выдающей-ВИД-после-разыменования F определяется следующим образом:

- пусть {имя} N – выдача ФОРМЫ-после-РАСКРЫТИЯ F;
- требуется, чтобы N не было псевдоименем;
- W – значение, именуемое этим N.

6.3. Распроцедуривание

{Распроцедуривание используется, когда надо вызвать процедуру без параметров. Например, в х := псч ↯ псевдослучайное число ↯ вызывается процедура, выдаваемая псч, и присваивается выдаваемое ею вещественное число; апостериорным видом псч является 'вещественное'. С точки зрения синтаксиса из априорного вида удаляется начальное 'процедура вырабатывающая'.}

6.3.1. *Синтаксис*

a) ФОРМА выдающая ЗНАЧЕНИЕ после
распроцедуривания {61C, 67a}:
ФОРМА выдающая процедуру вырабатывающую ЗНАЧЕНИЕ
после РАСКРЫТИЯ {61C}.

b) ФОРМА выдающая ВИД после
мягкого распроцедуривания {61D}:
ФОРМА выдающая процедуру вырабатывающую ВИД
после СМЯГЧЕНИЯ {61D}.

{Примеры:
a) псч (в вещ (псч))
b) х или у (в х или у := 3.14, см. 1.1.2)}

6.3.2. *Семантика*

Выдача W ФОРМЫ-выдающей-ЗНАЧЕНИЕ-после-распроцедуривания или ФОРМЫ-выдающей-ЗНАЧЕНИЕ-после-мягкого-распроцедуривания F в окружении E определяется следующим образом:

- пусть {процедура} R будет выдачей в E прямого наследника из F;
- W – выдача вызова {5.4.3.2.b} из R в E;
- требуется, чтобы W по области действия не была младше E.

6.4. Объединение

{Объединение не изменяет вида значения, выдаваемого конструктом во время счета, а просто создает для него бо́льшую свободу. Такое значение должно быть приемлемо не только для одного какого-то вида, но для целого множества видов. Однако после объединения это значение может использоваться примитивным действием только после динамической проверки его сопоставляющим-предложением {3.4.1.g}; в самом деле, с конструктом

вида 'ПРЕДСТАВИТЕЛЬ' нельзя запрограммировать никакие примитивные действия (кроме, конечно, присваивания переменной-вида-ПРЕДСТАВИТЕЛЬ).

Пример:

об (лог, лит) t, v;
t := "a"; t := истина; v := t.}

6.4.1. *Синтаксис*

a) ФОРМА выдающая ПРЕДСТАВИТЕЛЬ после объединения {61B}:
ФОРМА выдающая ЗНАЧЕНИЕ после РАСКРЫТИЯ {61C},
если ЗНАЧЕНИЕ служит ПРЕДСТАВИТЕЛЕМ {b}.

b) ЕСЛИ ЗНАЧЕНИЕ1 служит ЗНАЧЕНИЕМ2 {a, 34i, 71m}:
если ЗНАЧЕНИЕ1 эквивалентно ЗНАЧЕНИЮ2 {73a},
ЕСЛИ ложь;
если неверно что
ЗНАЧЕНИЕ1 эквивалентно ЗНАЧЕНИЮ2 {73a},
ЕСЛИ укрытые !ОБЫЧНЫЕ1 входят
в укрытые !ОБЫЧНЫЕ2 {73l, m, n},
если (!ОБЫЧНЫЕ1) есть (ЗНАЧЕНИЕ1)
или (объединение !ОБЫЧНЫХ1 воедино) есть (ЗНАЧЕНИЕ1),
если (!ОБЫЧНЫЕ2) есть (ЗНАЧЕНИЕ2)
или (объединение !ОБЫЧНЫХ2 воедино) есть (ЗНАЧЕНИЕ2).

{Примеры:

а) x (в uir := x) ·
и (в об (лит. цел, пуст) (u), в зоне: содержащей
об (цел, пуст) u := пустое)}

6.5. О б о б щ е н и е

{Обобщение преобразует целые числа в вещественные, вещественные числа в комплексные (в обоих случаях с тем же размером), значения вида 'БИТОВОЕ' в неупакованные векторы истинностных значений и значения вида 'СЛОГОВОЕ' в неупакованные векторы литер.

Например, в z := 1 выдача 1 обобщается до вещественного числа 1.0, а затем до комплексного числа (1.0, 0.0); синтаксически здесь априорный вид, специфицируемый цел, заменяется видом, специфицируемым вещ, а затем видом, специфицируемым компл.}

6.5.1. *Синтаксис*

A) БИТОВОЕ :: структура содержащая ?МЕРНУЮ букву алеф
для выборки вектора из логических в себе.

B) СЛОГОВОЕ :: структура содержащая ?МЕРНУЮ букву алеф
для выборки вектора из литерных в себе.

C) ?МЕРНОЕ :: ДОЛГОЕ ?ДОЛГОЕ; КРАТКОЕ ?КРАТКОЕ; ПУСТО.

D) ДОЛГОЕ :: буква д буква л буква и буква н.

E) КРАТКОЕ :: буква к буква о буква р.

F) ?ДОЛГОЕ :: ДОЛГОЕ ?ДОЛГОЕ; ПУСТО.

G) ?КРАТКОЕ :: КРАТКОЕ ?КРАТКОЕ; ПУСТО.

a) ФОРМА выдающая ?РАЗМЕРНОЕ вещественное

после обобщения {b, 61A}:
ФОРМА выдающая ?РАЗМЕРНОЕ
целое после РАСКРЫТИЯ {61C}.
{?РАЗМЕРНОЕ :: длинное ?ДЛИННОЕ;
короткое ?КОРОТКОЕ; ПУСТО.}

b) ФОРМА выдающая структуру содержащую
букву эр лат букву е лат для выборки
?РАЗМЕРНОГО вещественного
букву и лат букву эм лат для
выборки ?РАЗМЕРНОГО вещественного
в себе после обобщения {61A};
ФОРМА выдающая ?РАЗМЕРНОЕ вещественное
после РАСКРЫТИЯ {61C};
ФОРМА выдающая ?РАЗМЕРНОЕ вещественное
после обобщения {a}.

c) ФОРМА выдающая вектор из логических
после обобщения {61A};
ФОРМА выдающая БИТОВОЕ после РАСКРЫТИЯ {61C}.

d) ФОРМА выдающая вектор из литерных
после обобщения {61A}:
ФОРМА выдающая СЛОГОВОЕ после РАСКРЫТИЯ {61C}.

{Примеры:
a) 1 (в x := 1)
b) 1.0 (в compl z := 1.0) · 1 (в compl z := 1)
c) 2r 101 (в [ ] лог (2r 101))
d) r (в [ ] лит (r), см. 1.1.2)}

6.5.2. *Семантика*

Выдача W ФОРМЫ-выдающей-ВИД-после-обобщения F определяется следующие образом:

- пусть V – выдача прямого наследника F;
  Случай A: 'ВИД' есть '?РАЗМЕРНОЕ вещественное':
  - W – вещественное число, обобщаемое {2.1.3.1.e} из V;

  Случай B: 'ВИД' есть 'структура содержащая букву эр лат букву е лат для выборки ?РАЗМЕРНОГО вещественного букву и лат букву эм лат для выборки ?РАЗМЕРНОГО вещественного в себе'
  - W – {комплексное число, которое есть} структурное значение с полями, равными V и вещественному числу 0 того же размера {2.1.3.1.b}, что и V;

  Случай C: 'ВИД' есть 'вектор из логических' или 'вектор из литерных':
  - W – единственное поле из V.

6.6. Векторизация

{Векторизация позволяет построить массив из одного элемента. Если последний является именем, то результат векторизации также может быть именем, именующим этот массив.

Пример:

[1 : 1] вещ b1 := 4.13 }

6.6.1. *Синтаксис*

а) ФОРМА выдающая ?ИМЯ МАССИВА1 из ВИДА
после векторизации { 61A } :
если (МАССИВ1) есть (вектор),
ФОРМА выдающая ?ССЫЛКУ НА ВИД
после УСИЛЕНИЯ { 61A },
если (?ИМЯ) выводится из (?ССЫЛКИ НА) { 531b, c, - };
если (МАССИВ1) есть (вектор МАССИВОВ2).
ФОРМА выдающая ?ССЫЛКУ НА МАССИВ2 из ВИДА
после УСИЛЕНИЯ { 61A },
если (?ИМЯ) выводится из (?ССЫЛКИ НА) { 531b, c, - }.

{ Примеры:

а) 4.13 (в [1 : 1] вещ b1 := 4.13 ·
x1 (в [1 : 1, 1 : n] вещ b2 := x1) }

6.6.2. *Семантика*

а) Выдача W ФОРМЫ-выдающей-?ИМЯ-МАССИВА1-из-ВИДА-после-векторизации F определяется следующим образом:

· пусть V – выдача ФОРМЫ-после-УСИЛЕНИЯ из F;
Случай A: '?ИМЯ' есть 'ПУСТО':
· W – массив, „построенный" { b } из V для 'МАССИВА1';
Случай B: '?ИМЯ' есть 'ИМЯ':
· Если V – псевдоимя,
то W – тоже псевдоимя;
иначе W – имя, „построенное" { c } из V для 'МАССИВА1'.

b) Массив W, „построенный" из значения V для некоторого 'МАССИВА1', определяется следующим образом:
Случай A: 'МАССИВ1' есть 'вектор':
· W составляется из
(i) паспорта ((1, 1)),
(ii) { одного элемента } V;
Случай B: 'МАССИВ1' есть 'вектор МАССИВОВ2':
· пусть паспорт значения V будет $((l_1, u_1), ..., (l_n, u_n))$;
· W составляется из
(i) паспорта $((1, 1), (l_1, u_1), ..., (l_n, u_n))$,
(ii) элементов этого V;
· элемент, выбираемый в V по индексу $(i_1, ..., i_n)$, является элементом, выбираемым в W по индексу $(1, i_1, ..., i_n)$.

c) Имя N1, „построенное" из имени N для некоторого 'МАССИВА1', определяется следующим образом:
· N1 – { не обязательно вновь созданное } имя, равное по области действия имени N и именующее массив, построенный { b } для 'МАССИВА из значения, именуемого N;
Случай A: 'МАССИВ1' есть 'вектор':
· { единственным } подыменем этого N1 является N;

Случай В: 'МАССИВ1' есть 'вектор МАССИВОВ2':

- подыменем этого N1, выбираемым по $(1, i_1, ..., i_n)$, является подымя имени N, выбираемого по $(i_1, ..., i_n)$.

6.7. Опустошение

{Опустошение используют, чтобы отбросить выдачу некоторой основы, первичное назначение которой – вызвать побочный эффект; ее апостериорным видом становится тогда просто 'пустое значение'. Например, в х := 1; у := 1; присваивание у := 1 опустошается, а в проц t = цел : целч (псч Х 100); t ; использующий-идентификатор t опустошается после распроцедуривания, предписывающего вызов процедуры.

Присваивания и другие ПРЯМЫЕ опустошаются без всякого распроцедуривания, так что в проц пуст р; р := финиш это присваивание р := финиш не предусматривает неожидаемого здесь вызова процедуры финиш.}

6.7.1. *Синтаксис*

A) НЕРАСП {РОЦЕДУРИВАЕМОЕ} :: ПРОСТОЕ; СОСТАВНОЕ;
ПРЕДСТАВИТЕЛЬ; ИМЯ НЕРАСП;
процедура с ПАРАМЕТРАМИ вырабатывающая ЗНАЧЕНИЕ.

a) РАСКРЫВАЕМОЕ выдающее пустое значение
после опустошения {61A}:
РАСКРЫВАЕМОЕ выдающее НАРАСП
после распроцедуривания {63a};
РАСКРЫВАЕМОЕ выдающее НЕРАСП после сохранения {61f}.

b) ПРЯМОЕ выдающее пустое значение
после опустошения {61A}:
ПРЯМОЕ выдающее ВИД после сохранения {61f}.

{Примеры:

a) псч (в пропуск; псч;)
след псч (пред псч)
(в пропуск; след псч (пред псч);)

b) проц пуст (рр)
(в проц проц пуст рр = проц пуст:
(печ (1); пуст : печ (2));
проц пуст (*рр*);)}

6.7.2. *Семантика*

Исполнение ФОРМЫ-выдающей-пустое-значение-после-опустошения состоит из исполнения ее прямого наследника и выдает пустое.

7. ВИДЫ И СРЕДЫ

{Идентификация свойств в среде представляет собой статический двойник динамического определения (4.8.2.b) значения в некотором окружении: поиск проводится с самого нового (самого младшего) уровня по направлению к предыдущим (более старшим) уровням.

Виды составляются из примитивных видов, таких, как 'логическое', с

помощью 'ПРОЛОГОВ', таких, как 'структура содержащая', и могут быть рекурсивными. Рекурсивные виды, выписываемые различным образом, могут тем не менее быть эквивалентными. Синтаксис проверяет эквивалентность таких видов, доказывая невозможность обнаружения какого-либо несовпадения между их соответствующими подструктурами или составляющими видами.

Предотвращается ряд небезопасных использований свойств. Никакой идентификатор или индикатор-вида не описывается более одного раза в каждой зоне. Виды операндов всякой формулы не определяют более одной операции. Рекурсия в видах не приводит к динамическому созданию объектов неограниченного размера и не допускает двусмысленных приведений.}

7.1. Независимость свойств

{Следующий синтаксис определяет, могут ли два свойства (т.е. две 'ПАРЫ'), подобные тем, которые соответствуют вещ x и цел x, быть заложены в один и тот же 'СЛОЙ'.}

7.1.1. *Синтаксис*

A) ПРИСТАВКА :: процедура вырабатывающая; ИМЯ.

B) КОРЕНЬ :: ПРОСТОЕ; СОСТАВНОЕ; ПРЕДСТАВИТЕЛЬ;
пустое значение; процедура с ПАРАМЕТРАМИ
вырабатывающая ЗНАЧЕНИЕ.

C)* ?ПРИСТАВКИ :: ПРИСТАВКА ?ПРИСТАВКИ; ПУСТО.
{ПАРА :: ОПИСАНИЕ; МЕТКА; ПОЛЕ.
ПРИЗНАК :: ВИД; ЗНАЧЕНИЕ НОМЕР; БИНАРНОЕ;
метка; выборка ВИДА.
ОБОЗНАЧЕНИЕ :: СЛОВО; ИНДИКАНТ; ИНФИКС; ПРЕФИКС.
АФФИКС :: ИНФИКС; ПРЕФИКС.}

a) ЕСЛИ ПАРА1 не зависит от !ПАР2 ПАРЫ2 {a, 48a, c, 72a}:
ЕСЛИ ПАРА1 не зависит от !ПАР2 {a, c,}
и ПАРА1 не зависит от ПАРЫ2 {c}.

b) ЕСЛИ ПАРА не зависит от ПУСТО {48a, c, 72a}: ЕСЛИ истина.

c) ЕСЛИ ОБОЗНАЧЕНИЕ1 для ПРИЗНАКА1 не зависит
от ОБОЗНАЧЕНИЯ2 для ПРИЗНАКА2 {a, 48a, c, 72a}:
если неверно что (ОБОЗНАЧЕНИЕ1) есть (ОБОЗНАЧЕНИЕ2)
ЕСЛИ истина;
если (ОБОЗНАЧЕНИЕ1) есть (ОБОЗНАЧЕНИЕ2)
и (ОБОЗНАЧЕНИЕ1) есть (АФФИКС),
ЕСЛИ ПРИЗНАК1 не зависит от ПРИЗНАКА2 {d}.

d) ЕСЛИ ПРИЗНАК1 не зависит от ПРИЗНАКА2 {c}:
если ПРИЗНАК1 связан с ПРИЗНАКОМ2 {e, f, g, h, i, j, -},
ЕСЛИ ложь;
если неверно что ПРИЗНАК1 связан с
ПРИЗНАКОМ2 {e, f, g, h, i, j, -}, ЕСЛИ истина.

e) ЕСЛИ ОДНОМЕСТНАЯ связана с ДВУМЕСТНОЙ {d}: ЕСЛИ ложь.

f) ЕСЛИ ДВУМЕСТНАЯ связана с ОДНОМЕСТНОЙ {d}: ЕСЛИ ложь.

g) ЕСЛИ ОПЕРАЦИЯ связана с БИНАРНЫМ {d}: ЕСЛИ ложь.
h) ЕСЛИ БИНАРНОЕ связано с ОПЕРАЦИЕЙ {d}: ЕСЛИ ложь.
i) ЕСЛИ процедура с параметром вида ВИД1 параметром
вида ВИД2 выбаратывающая ЗНАЧЕНИЕ1 связана с
процедурой с параметром вида ВИД3 параметром
вида ВИД4 вырабатывающей ЗНАЧЕНИЕ2 {d}:
ЕСЛИ ВИД1 крепко связан с ВИДОМ3 {k}
и ВИД2 крепко связан с ВИДОМ4 {k}
j) ЕСЛИ процедура с параметром вида ВИД1 вырабатывающая
ЗНАЧЕНИЕ1 связана с процедурой с параметром
вида ВИД2 вырабатывающей ЗНАЧЕНИЕ2 {d} :
ЕСЛИ ВИД1 крепко связан с ВИДОМ2 {k}.
k) ЕСЛИ ЗНАЧЕНИЕ1 крепко связано со ЗНАЧЕНИЕМ2 {i, j}:
ЕСЛИ !ОБЫЧНЫЕ1 скреплены со ЗНАЧЕНИЕМ2 {l, m}
или !ОБЫЧНЫЕ2 скреплены со ЗНАЧЕНИЕМ1 {l, m,},
если (!ОБЫЧНЫЕ1) есть (ЗНАЧЕНИЕ1)
или (объединение !ОБЫЧНЫХ1 воедино) есть (ЗНАЧЕНИЕ1),
если (!ОБЫЧНЫЕ2) есть (ЗНАЧЕНИЕ2)
или (объединение !ОБЫЧНЫХ2 видов) есть (ЗНАЧЕНИЕ2).
l) ЕСЛИ !ОБЫЧНЫЕ ОБЫЧНОЕ секреплены со ЗНАЧЕНИЕМ {k, l}:
ЕСЛИ !ОБЫЧНЫЕ скреплены со ЗНАЧЕНИЕМ {l, m}
или ОБЫЧНОЕ скреплено со ЗНАЧЕНИЕМ {m}.
m) ЕСЛИ ЗНАЧЕНИЕ1 скреплено со ЗНАЧЕНИЕМ2 {k, l, n, 47f}:
ЕСЛИ ЗНАЧЕНИЕ1 эквивалентно ЗНАЧЕНИЮ2 {73a}
или ЗНАЧЕНИЕ1 служит ЗНАЧЕНИЕМ2 {64b}
или ЗНАЧЕНИЕ2 есть корень ЗНАЧЕНИЯ1 {n}.
n) ЕСЛИ ЗНАЧЕНИЕ2 есть корень ЗНАЧЕНИЯ1 {m}:
если (ЗНАЧЕНИЕ1) есть (ПРИСТАВКА ЗНАЧЕНИЕ3),
ЕСЛИ ЗНАЧЕНИЕ5 скреплено со ЗНАЧЕНИЕМ2 {m},
если ЗНАЧЕНИЕ5 фиксирует ЗНАЧЕНИЕ3 {47a, b, c};
если (ЗНАЧЕНИЕ1) есть (КОРЕНЬ), ЕСЛИ ложь.

{ Чтобы предотвратить двусмысленное использование индикаторов, как в вещ x, цел x; x := 0, на определяющие-индикаторы, входящие в некоторую данную зону, налагаются определенные ограничения. Они обеспечиваются синтаксической проверкой „независимости" свойств, заложенных в соответствующий 'СЛОЙ' (правила a, b, c). Достаточное условие независимости пары свойств, каждое из которых является некоторым 'ОБОЗНАЧЕНИЕМ для ПРИЗНАКА', не выполненное в приведенном выше примере, состоит в различии этих 'ОБОЗНАЧЕНИЙ' (правило c). Для 'ОБОЗНАЧЕНИЙ', не являющихся 'АФФИКСОМ', это условие также и необходимо, так что даже вещ x, цел x; пропуск не есть последовательное-предложение.

Для двух свойств 'АФФИКС для ПРИЗНАКА1' и 'АФФИКС для ПРИЗНАКА2' проверка на независимость несколько сложнее, как это можно видеть из последовательного-предложения

оп + = (цел i) лог : истина, оп + = (цел i, j) цел : 1,

```
    оп + = (цел i, лог j) цел : 2, прио + = 6;
    0 + + 0 ¢ = 2 ¢.
```

Двусмысленности можно обнаружить в

```
    прио + = 6, + = 7;  1 + 2 × 3 ¢ 7 или 9? ¢.
```

в

```
    оп z = (цел i) цел : 1, вид z = цел;
      z i ¢ формула или описание? ¢;  пропуск
```

и в

```
      оп? = (об(имя вещ, лит) a) цел : 1, оп ? = (вещ a) цел : 2;
        ? лок вещ ¢ 1 или 2? ¢.
```

В таких случаях проверяется, независимы ли эти два 'ПРИЗНАКА' (правила c, d). Всякое 'ЗНАЧЕНИЕ НОМЕР' всегда зависит от любого 'ПРИЗНАКА' (правило d). 'ОДНОМЕСТНАЯ' всегда не зависит от 'ДВУМЕСТНОЙ' (правила d, e, f) и обе они не зависят от 'БИНАРНОГО' (т.е. от 'приоритета ПРИОРИТЕТ') (правила d, g, h). В случае двух 'ОПЕРАЦИЙ', обеих 'ОДНОМЕСТНЫХ' или 'ДВУМЕСТНЫХ', двусмысленность может возникнуть, если виды их соответствующих параметров „крепко связаны", т.е. если вид операнда может крепко приводиться к виду параметра (виды пары операндов могут крепко приводиться к видам параметров) какой-нибудь 'ОПЕРАЦИИ' (правила i, j). В примере с двумя определениями для ? указанные две 'ОПЕРАЦИИ' связаны, поскольку крепко связаны виды, специфицируемое описателями об (имя вещ, лит) и вещ; специфицируемый имя вещ вид крепко приводим к любому из них.

Можно показать, что два вида крепко связаны, если один из них или какое-то составляющее 'ОБЫЧНОЕ' одного из них может крепко приводиться к другому (правила k, l), что потребует некоторой последовательности из нуля или более раскрывающих приведений и затем не более одного объединения (6.4.1.а). Возможна ли такая последовательность приведений между двумя видами, определяется предикатом 'скреплены' (правила m, n).

'ПАРА1' делает, кроме того, недоступной 'ПАРУ2' во внешнем 'СЛОЕ', если эта 'ПАРА2' зависит от 'ПАРЫ1'; например,

```
начало цел x;
    нач вещ x; ¢ здесь 'ПАРА1' есть 'буква икс лат для
      имени вещественного' ¢
    пропуск
  кон
конец
```

и аналогично

```
начало оп ? = (цел i) цел : 1, цел k := 2;
  нач оп ? = (имя цел i) цел : 3;
   ? k ¢ вырабатывает 3, а ? 4 не может встретиться в
    этом контексте, поскольку его обозначение-операции
    здесь недоступно ¢
  кон
конец.}
```

7.2. Идентификация в средах

{Этот раздел обеспечивает, что для каждого использующего-индикатора существует соответствующая 'ПАРА' в некотором подходящем 'СЛОЕ' его среды.}

7.2.1. *Синтаксис*

{?ПАРЫ :: !ПАРЫ; ПУСТО.
!ПАРЫ :: ПАРА; !ПАРЫ ПАРА.
ПАРА :: ОПИСАНИЕ; МЕТКА; ПОЛЕ.
ПРИЗНАК :: ВИД; ЗНАЧЕНИЕ НОМЕР;
БИНАРНОЕ; метка; выборка ВИДА.
ОБОЗНАЧЕНИЕ :: СЛОВО; ИНДИКАНТ; ИНФИКС; ПРЕФИКС.}

a) ЕСЛИ ПАРА идентифицирована в СРЕДЕ
с новыми ?ПАРАМИ {a, 48b, 542a}:
если ПАРА находится в ?ПАРАХ {b, c, -}, ЕСЛИ истина;
если ПАРА не зависит от ?ПАР {71a, b, c},
ЕСЛИ ПАРА идентифицирована в СРЕДЕ {a, -}.

b) ЕСЛИ ПАРА1 находится в !ПАРАХ2 ПАРЕ? {a, b, 48d}:
ЕСЛИ ПАРА1 находится в ПАРЕ2 {c, -}
или ПАРА1 находится в !ПАРАХ2 {b, c, -}.

c) ЕСЛИ ОБОЗНАЧЕНИЕ для ПРИЗНАКА1 находится
в ОБОЗНАЧЕНИИ для ПРИЗНАКА2 {a, b, 48d}:
если (ПРИЗНАК1) есть (метка) или (ПРИЗНАК1) есть
(БИНАРНОЕ) или (ПРИЗНАК1) есть (выборка ВИДА),
ЕСЛИ (ПРИЗНАК1) есть (ПРИЗНАК2);
если (ПРИЗНАК1) есть (ЗНАЧЕНИЕ1 ?НОМЕР)
и (ПРИЗНАК2) есть (ЗНАЧЕНИЕ2 ?НОМЕР),
ЕСЛИ ЗНАЧЕНИЕ1 эквивалентно ЗНАЧЕНИЮ2 {73a}.

{Всякая среда, исключая первичную (которая есть просто 'новое'), есть некоторая 'СРЕДА с СЛОЕМ' (т.е. 'СРЕДА с новыми ?ПАРАМИ'). Идентификация 'ПАРЫ' происходит сначала путем поиска ее в данном 'СЛОЕ' (правило a). Если эта 'ПАРА' есть 'ОБОЗНАЧЕНИЕ для метки' или 'ОБОЗНАЧЕНИЕ для БИНАРНОГО', то простое обнаружение соответствующей 'ПАРЫ' служит достаточной проверкой (правило c). Если же эта 'ПАРА' есть ' ОБОЗНАЧЕНИЕ для ЗНАЧЕНИЯ ?НОМЕР', то необходимо обратиться к механизму эквивалентности (правило c). Если данная 'ПАРА' не найдется в указанном 'СЛОЕ', то поиск продолжается в 'СРЕДЕ' (без этого 'СЛОЯ'), при условии что она не зависит от всех 'ПАР' этого 'СЛОЯ', а иначе поиск обрывается (правило a). Отметим, что правила b и c несут двойную нагрузку, поскольку они применяются также и для проверки законности использующих-указателей-поля (4.8.1.d).}

7.2.2. *Семантика*

a) Если некоторый блок-в-СРЕДЕ R (3.0.1.f) содержит использующий-индикатор l (4.8.1.b), для которого существует наследник если-ПАРА-идентифицирована-в-СРЕДЕ-с-СЛОЕМ, но не существует наследника если-ПАРА-идентифицирована-в-СРЕДЕ, то R – „определяющий блок" данного l. {Эта

'СРЕДА' всегда будет средой, действующей как раз за пределами данного блока.}

b) Всякой использующий-ОБОЗНАЧЕНИЕ-индикатор-выдающий-ПРИЗНАК l, определяющий блок-в-СРЕДЕ { a } которого есть R, „идентифицирует" содержащийся в R определяющий-СЛОВО-индикатор-в-СРЕДЕ-с-СЛОЕМ-выдающий-ПРИЗНАК.

{Например, в

(# 1 # вещ i = 2,0; (# 2 # цел i = 1: (# 3 # вещ x; печ (i)

(# 3 #) # 2 #) # 1 #)

три блока. В силу синтаксиса использующий-идентификатор i в печ (i) вынужден быть использующим-букву-и-лат-идентификатором-в-СРЕДЕ-с-новой-буквой-и-лат-для-вещественного-новой-буквой-и-лат-для-целого-новой-буквой-икс-лат-для-имени-вещественного-выдающим-целое (4.8.1.b). Его определяющим блоком является определяющее-новую-букву-и-лат-для-целого-последовательное-предложение-в-СРЕДЕ-с-новой-буквой-и-лат-для вещественного (3.2.1.a) с номером # 2 # он идентифицирует определяющий-идентификатор i, содержащийся в цел i (а не в вещ i), и его видом является 'целое'.}

{Благодаря наличию аналогичного механизма некоторую БИНАРНУЮ-формулу (5.4.2.1.a) можно назвать „идентифицирующей" то определяющее-обозначение-операции-выдающее-БИНАРНОЕ (4.8.1.a), которое определяет ее приоритет.}

c) Окружение E, „необходимое для" конструкта C в окружении E1, определяется следующим образом:

Если E1 – первичное окружение (2.2.2.a),

то E есть E1;

иначе пусть E1 будет составлено из участка L, соответствующего каким-то '?ПАРАМ', и другого окружения E2;

Если C содержит любой использующий-СЛОВО-индикатор-выдающий-ПРИЗНАК,

- не идентифицирующий { b } никакого определяющего-индикатора, содержащегося в C,
- не являющийся прямым наследником индикатором-вида для формального или виртуального-описателя и
- такой, что предикат 'если ОБОЗНАЧЕНИЕ для ПРИЗНАКА находится в ?ПАРАХ' {7.2.1.b} выполняется,

то E есть E1;

иначе {L не необходимо для C и} E – окружение, необходимое для C в E2.

{Окружение, необходимое для конструкта, используется в семантике текстов-процедуры (5.4.1.2) и в „устанавливании" (3.2.2.b). Например, в

2 # проц пуст pp; цел n; (# 1 # проц p = пуст: печ (n); pp := p),

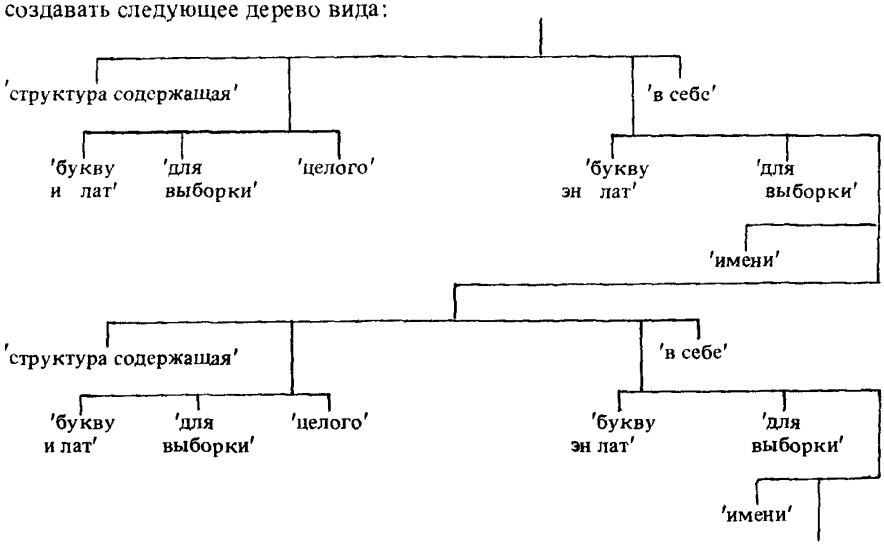
если E1 и E2 – окружения, устанавливаемые в результате исполнения последовательных-предложений, отмеченных примечаниями # 1 # и # 2 #, то E2 – окружение, необходимое в E1 для текста-процедуры пуст: печ (n), так что процедура, выдаваемая p в E1, составлена из этого текста-процедуры и E2 (5.4.1.2). Следовательно, область действия данной процедуры есть об-

ласть действия окружения E2 (2.1.3.5.c), а отсюда вытекает, что присваивание (5.2.1.2.b), вызываемое посредством pp ; = p, корректно определено.

7.3. Эквивалентность видов

{В этом разделе определяется эквивалентность или неэквивалентность различных 'ЗНАЧЕНИЙ'. Обсуждение эквивалентных 'ЗНАЧЕНИЙ' см. в 2.1.1.2.}

{Один из способов увидеть рекурсивные виды – рассматривать их как бесконечные деревья. Такое „дерево вида" получается повторной подстановкой в некотором выписывании 'ВИДА' из соответствующего 'ЦИ определения ВИДА' вместо каждого 'использования ЦИ'. Таким образом, выписывание 'ци I определение структуры содержащей букву и лат для выборки целого букву эн лат для выборки имени использования ци I в себе' будет создавать следующее дерево вида:



(и т. д.)

Два выписывания эквивалентны тогда и только тогда, когда они обусловливают идентичные деревья вида. Настоящий синтаксис эквивалентности проверяет эквивалентность двух выписываний посредством, так сказать, одновременного развертывания этих двух деревьев до тех пор, пока не найдется какая-нибудь разница (что приведет к тупику) или пока не станет явным, что никакой разницы найтись не может. В некоторой степени структура деревьев вида отображается в растущем дереве порождения.}

7.3.1. *Синтаксис*

A) УКРЫТИЕ :: укрытое; ЦИ помнит ВИД УКРЫТИЕ; инь УКРЫТИЕ; ян УКРЫТИЕ; запомненные ЗНАЧЕНИЕ1 ЗНАЧЕНИЕ2 УКРЫТИЕ.

B) ПРОЛОГ :: ПРОСТОЕ; ПРИСТАВКА {71A}; структура содержащая;

?ПОДВИЖНЫЙ МАССИВ из; процедура с; объединение;
пустое значение.

C) ?ЭПИЛОГ :: ЗНАЧЕНИЕ; !ПОЛЯ в себе;
!ПАРАМЕТРАМИ вырабатывающая ЗНАЧЕНИЕ; !ОБЫЧНЫХ
воедино; ПУСТО.

D) !ЧАСТИ :: ЧАСТЬ; !ЧАСТИ ЧАСТЬ.

E) ЧАСТЬ :: ПОЛЕ; ПАРАМЕТР.

a) ЕСЛИ ЗНАЧЕНИЕ1 эквивалентно ЗНАЧЕНИЮ2 {64b, 71m, 72c:
ЕСЛИ укрытое ЗНАЧЕНИЕ1 эквивалентно
укрытому ЗНАЧЕНИЮ2 {b}.

b) ЕСЛИ УКРЫТИЕ1 ЗНАЧЕНИЕ1 эквивалентны
УКРЫТИЮ2 ЗНАЧЕНИЮ2 {a, b, e, i, j, n}:
если (УКРЫТИЕ1) содержит (запомненные
ЗНАЧЕНИЕ1 ЗНАЧЕНИЕ2) или (УКРЫТИЕ2) содержит
(запомненные ЗНАЧЕНИЕ2 ЗНАЧЕНИЕ1), ЕСЛИ истина;
если неверно что (УКРЫТИЕ1) содержит (запомненные
ЗНАЧЕНИЕ1 ЗНАЧЕНИЕ2) или (УКРЫТИЕ2) содержит
(запомненные ЗНАЧЕНИЕ2 ЗНАЧЕНИЕ1),
ЕСЛИ (ПРОЛОГ3) есть (ПРОЛОГ4) и
запомненные ЗНАЧЕНИЕ1 ЗНАЧЕНИЕ2 УКРЫТИЕ3
?ЭПИЛОГ3 эквивалентны
УКРЫТИЮ4 ?ЭПИЛОГУ4 {b, d, e, k, q, -},
если УКРЫТИЕ3 ПРОЛОГ3 ?ЭПИЛОГ3 развертываются
из УКРЫТИЯ1 ЗНАЧЕНИЯ1 {c}
и УКРЫТИЕ4 ПРОЛОГ4 ?ЭПИЛОГ4 развертываются
из УКРЫТИЯ2 ЗНАЧЕНИЯ2 {c}.

c) ЕСЛИ УКРЫТИЕ2 ПРОЛОГ ?ЭПИЛОГ развертываются
из УКРЫТИЯ1 ЗНАЧЕНИЯ {b, c}:
если (ЗНАЧЕНИЕ) есть (ПРОЛОГ ?ЭПИЛОГ),
ЕСЛИ (ПРОЛОГ) меняет УКРЫТИЕ1 на
УКРЫТИЕ2 {74a, b, c, d, -};
если (ЗНАЧЕНИЕ) есть (ЦИ определение ВИДА),
если неверно что (УКРЫТИЕ1) содержит (ЦИ помнит),
ЕСЛИ УКРЫТИЕ2 ПРОЛОГ ?ЭПИЛОГ развертываются
из ЦИ помнит ВИД УКРЫТИЯ ВИДА {c};
если (ЗНАЧЕНИЕ) есть (использование ЦИ)
и (УКРЫТИЕ1) есть (ПОНЯТИЕ ЦИ помнит ВИД УКРЫТИЕ3)
и (ПОНЯТИЕ) содержит (инь)
и (ПОНЯТИЕ) содержит (ян),
ЕСЛИ УКРЫТИЕ2 ПРОЛОГ ?ЭПИЛОГ развертываются
из УКРЫТИЯ1 ВИДА {c}.

d) ЕСЛИ УКРЫТИЕ1 !ПОЛЯ1 в себе эквивалентны
УКРЫТИЮ2 !ПОЛЯМ2 в себе {b}:
ЕСЛИ УКРЫТИЕ1 !ПОЛЯ1 эквивалентны
УКРЫТИЮ2 !ПОЛЯМ2 {f, g, h, i}.

e) ЕСЛИ УКРЫТИЕ1 !ПАРАМЕТРАМИ1
вырабатывающая ЗНАЧЕНИЕ1
эквивалентны УКРЫТИЮ2 !ПАРАМЕТРАМИ2
вырабатывающей ЗНАЧЕНИЕ2 { b }:
ЕСЛИ УКРЫТИЕ1 !ПАРАМЕТРЫ1 эквивалентны
УКРЫТИЮ2 !ПАРАМЕТРАМ2 { f, g, h, j } и УКРЫТИЕ1
ЗНАЧЕНИЕ1 эквивалентны УКРЫТИЮ2 ЗНАЧЕНИЮ2 { b }.

f) ЕСЛИ УКРЫТИЕ1 !ЧАСТИ1 ЧАСТЬ1 эквивалентны
УКРЫТИЮ2 !ЧАСТЯМ2 ЧАСТИ2 { d, e, f }:
ЕСЛИ УКРЫТИЕ1 !ЧАСТИ1 эквивалентны
УКРЫТИЮ2 !ЧАСТЯМ2 { f, g, h, i, j } и УКРЫТИЕ1
ЧАСТЬ1 эквивалентны УКРЫТИЮ2 ЧАСТИ2 { i, j }.

g) ЕСЛИ УКРЫТИЕ1 !ЧАСТИ1 ЧАСТЬ1 эквивалентны
УКРЫТИЮ2 ЧАСТИ2 { d, e, f } : ЕСЛИ ложь.

h) ЕСЛИ УКРЫТИЕ1 ЧАСТЬ1 эквивалентны УКРЫТИЮ2
!ЧАСТЯМ2 ЧАСТИ2 { d, e, f } : ЕСЛИ ложь.

i) ЕСЛИ УКРЫТИЕ1 СЛОВО1 для выборки ВИДА1 эквивалентны
УКРЫТИЮ2 СЛОВУ2 для выборки ВИДА2 { d, f }:
ЕСЛИ (СЛОВО1) есть (СЛОВО2) и
УКРЫТИЕ1 ВИД1 эквивалентны УКРЫТИЮ2 ВИДА2 { b }.

j) ЕСЛИ УКРЫТИЕ1 параметр вида ВИД1 эквивалентны
УКРЫТИЮ2 параметру вида ВИД2 { e, f }:
ЕСЛИ УКРЫТИЕ1 ВИД1 эквивалентны УКРЫТИЮ2 ВИДУ2 { b }.

k) ЕСЛИ УКРЫТИЕ1 !ОБЫЧНЫХ1 воедино эквивалентны
УКРЫТИЮ2 !ОБЫЧНЫХ2 воедино { b }:
ЕСЛИ УКРЫТИЕ1 !ОБЫЧНЫЕ1 входят в УКРЫТИЕ2
!ОБЫЧНЫЕ2 { l, m n } и УКРЫТИЕ2 !ОБЫЧНЫЕ2
входят в УКРЫТИЕ1 !ОБЫЧНЫЕ1 { l, m, n } и
число !ОБЫЧНЫХ1 равно числу !ОБЫЧНЫХ2 { o, p }.

l) ЕСЛИ УКРЫТИЕ1 !ОБЫЧНЫЕ1 ОБЫЧНОЕ1 входят
в УКРЫТИЕ2 !ОБЫЧНЫЕ2 { k, l, m, 46s, 64b }:
ЕСЛИ УКРЫТИЕ1 !ОБЫЧНЫЕ1 входят в УКРЫТИЕ2
!ОБЫЧНЫЕ2 { l, m, n } и УКРЫТИЕ1 ОБЫЧНОЕ1
входят в УКРЫТИЕ2 !ОБЫЧНЫЕ2 { m, n },

m) ЕСЛИ УКРЫТИЕ1 ОБЫЧНОЕ1 входят в УКРЫТИЕ2
!ОБЫЧНЫЕ2 ОБЫЧНОЕ2 { k, l, m, 46s, 64b }:
ЕСЛИ УКРЫТИЕ1 ОБЫЧНОЕ1 входят в УКРЫТИЕ2
!ОБЫЧНЫЕ2 { m, n } или УКРЫТИЕ1 ОБЫЧНОЕ1
входят в УКРЫТИЕ2 ОБЫЧНОЕ2 { n }.

n) ЕСЛИ УКРЫТИЕ1 ОБЫЧНОЕ1 входит
в УКРЫТИЕ2 ОБЫЧНОЕ2 { k, l, m, 64b };
ЕСЛИ УКРЫТИЕ1 ОБЫЧНОЕ1 эквивалентны
УКРЫТИЮ2 ОБЫЧНОМУ2 { b } .

o) ЕСЛИ число !ОБЫЧНЫХ1 ОБЫЧНОГО1 равно
числу !ОБЫЧНЫХ2 ОБЫЧНОГО2 k, o }:

ЕСЛИ число !ОБЫЧНЫХ1 равно числу !ОБЫЧНЫХ2 {o, p, -}.

p) ЕСЛИ число ОБЫЧНОГО1 равно числу
ОБЫЧНОГО2 { k, o } : ЕСЛИ истина.

q) ЕСЛИ УКРЫТИЕ1 ПУСТО эквивалентны
УКРЫТИЕ2 ПУСТО { b } : ЕСЛИ истина.

{ Правило a вводит 'УКРЫТИЯ', используемые в ходе определения эквивалентности как ассоциативная память. Таких 'УКРЫТИЙ' два, по одному для каждого вида. Правило b приводит к немедленному заключению, если рассматриваемые 'ЗНАЧЕНИЯ' уже запомнены (см. ниже) в какомнибудь подходящем 'УКРЫТИИ' в форме 'запомненные ЗНАЧЕНИЕ1 ЗНАЧЕНИЕ2'. Если это не так, то указанные два 'ЗНАЧЕНИЯ' сначала запоминаются в 'УКРЫТИИ' (в том, которое слева), а затем каждое 'ЗНАЧЕНИЕ' развертывается (правило c) и разбивается на его 'ПРОЛОГ' и его '?ЭПИЛОГ', например 'имя вещественного' разбивается на 'имя' и 'вещественное'.

Если 'ПРОЛОГИ' различны, то вопрос решен (правило b); в противном случае данные '?ЭПИЛОГИ' анализируются в соответствии с их структурами (которые должны совпадать, если соответствующие 'ПРОЛОГИ' идентичны). Во всех случаях, кроме того, когда 'ПРОЛОГИ' были 'объединение', эквивалентность проверяется исследованием соответствующих составляющих согласно схеме

правило	'?ЭПИЛОГ'	составляющие
d	'ПОЛЯ в себе'	'!ПОЛЯ'
e	'ПАРАМЕТРАМИ вырабатывающая ЗНАЧЕНИЕ'	'!ПАРАМЕТРЫ' и 'ЗНАЧЕНИЕ'
f	'!ПОЛЯ ПОЛЕ'	'!ПОЛЯ' и 'ПОЛЕ'
f	'!ПАРАМЕТРЫ ПАРАМЕТР'	'ПАРАМЕТРЫ' и 'ПАРАМЕТР'
i	'СЛОВО для выборки ВИДА'	'СЛОВО' и 'ВИД'
j	'параметр вида ВИД'	'ВИД'

Для объединения соответствующие '?ЭПИЛОГИ' имеют форму '!ОБЫЧНЫХ1 воедино' и '!ОБЫЧНЫХ2 воедино'. Поскольку внутри эквивалентных объединений 'ОБЫЧНЫЕ' могут переставляться, как в специфицируемых об (вещ, цел) и об (цел, вещ) видах, то для них эквивалентность определяется проверкой того, что '!ОБЫЧНЫЕ1' входят (в теоретико-множественном смысле) в '!ОБЫЧНЫЕ2' и '!ОБЫЧНЫЕ2' входят в 'ОБЫЧНЫЕ1'; при этом, конечно, проверка на вхождение рекурсивно обращается к проверке эквивалентности (правила k, l, m, n, o, p).

Всякое 'ЗНАЧЕНИЕ' развертывается (правило c) в форму 'ПРОЛОГ ?ЭПИЛОГ' посредством определения того, что:

(i) оно уже имеет эту форму: в таком случае в его 'УКРЫТИЕ' могут помещаться маркеры ('инь' и 'ян') для того, чтобы позднее определить правильность построения этого 'ЗНАЧЕНИЯ' (см. 7.4);

(ii) оно есть какое-то 'ЦИ определение ВИДА': в этом случае в его 'УКРЫТИЕ' помещается 'ЦИ помнит ВИД' (при условии, что это конкрет-

ное 'ЦИ' уже не там), а затем развертывается этот 'ВИД';

(iii) оно есть некоторое 'использование ЦИ': в этом случае в его 'УКРЫТИИ' уже должно быть какое-то 'ЦИ помнит ВИД'. Тогда развертывается этот 'ВИД' *после проверки на правильность построения* (см. 7.4), состоящей в определении того, что в этом 'УКРЫТИИ' перед рассматриваемым 'ЦИ помнит ВИД' находится по крайней мере одно 'инь' и по крайней мере одно 'ян'.}

{ Прежде чем пара '?ЭПИЛОГОВ' проверяется на эквивалентность, в 'УКРЫТИИ' запоминается, что исходная пара 'ЗНАЧЕНИЙ' уже проверялась. Это делается, чтобы обязательно выйти прямо на 'ЕСЛИ истина', если данные 'ЗНАЧЕНИЯ' когда-нибудь снова нужно будет проверять на эквивалентность на более низких уровнях дерева порождения. Поскольку число пар составляющих тех 'ЗНАЧЕНИЙ', которые могут выводиться из двух любых данных 'ЗНАЧЕНИЙ', конечно, рассматриваемый процесс заканчивается.

Остается показать, что этот процесс корректен. Рассмотрим неограниченное (возможно, бесконечное) дерево порождения, которое могло бы получиться, если бы в данном синтаксисе не было сокращающих выходов (что получится, если опустить первую альтернативу вместе с первым звеном второй альтернативы в правиле b). Если два 'ЗНАЧЕНИЯ' неэквивалентны, то в их деревьях вида есть кратчайший путь от корневой вершины к какой-то вершине, показывающей разницу. Очевидно, что изображение этого кратчайшего пути в неограниченном дереве порождения не может содержать повторяемой проверки на эквивалентность никакой пары 'ЗНАЧЕНИЙ' и, значит, ни один из сокращающих выходов на 'ЕСЛИ истина' в ограниченном дереве порождения не может входить в этот кратчайший путь. Следовательно, этот путь до различия должен также присутствовать и в (ограниченном) дереве порождения, порождаемом настоящим синтаксисом. Если указанный процесс проверки не показывает разницы в этом ограниченном дереве, то ни за какое число шагов нельзя обнаружить никакой разницы; т.е. данные 'ЗНАЧЕНИЯ' эквивалентны.}

7.4. Правильность построения

{ Вид правильно построен, если

(i) исполнение фактического-описателя, специфицирующего этот вид является конечным действием (т.е. любое значение этого вида можно хранить в конечной памяти) и

(ii) он не является сильно приводимым из самого себя (поскольку это вело бы к двусмысленностям при приведении).}

7.4.1. *Синтаксис*

a) ЕСЛИ (ПОНЯТИЕ) меняет УКРЫТИЕ на УКРЫТИЕ {73c}:
если (ПОНЯТИЕ) есть (ПРОСТОЕ) или
(ПОНЯТИЕ) есть (?ПОДВИЖНЫЙ МАССИВ из) или
(ПОНЯТИЕ) есть (объединение) или (ПОНЯТИЕ) есть
(пустое значение), ЕСЛИ истина.

b) ЕСЛИ (ПРИСТАВКА) меняет УКРЫТИЕ

на инь УКРЫТИЕ {73c} : ЕСЛИ истина.

c) ЕСЛИ (структура содержащая) меняет УКРЫТИЕ
на ян УКРЫТИЕ {73c} : ЕСЛИ истина.

d) ЕСЛИ (процедура c) меняет УКРЫТИЕ
на инь ян УКРЫТИЕ {73c} : ЕСЛИ истина.

{ В качестве побочного продукта выявления эквивалентности видов они проверяются на правильность построения (7.3.1.c). Все нерекурсивные виды правильно построены. Для рекурсивных видов необходимо, чтобы каждый цикл в каждом выписывании такого вида (от 'ЦИ определения ВИДА' до 'использования ЦИ') проходил по крайней мере через один 'ПРОЛОГ', который есть инь, обеспечивая условие (i), и один (возможно, тот же самый) 'ПРОЛОГ', который есть ян, обеспечивая условие (ii). Инь 'ПРОЛОГАМИ' являются 'ПРИСТАВКА' и 'процедура c'. Ян 'ПРОЛОГАМИ' являются 'структура содержащая' и 'процедура c'. Остальные 'ПРОЛОГИ', включая '?ПОДВИЖНЫЙ МАССИВ из' и 'объединение', не являются ни инь, ни ян. Это означает, что все виды, специфицируемые посредством a, b и c в

вид a = ст (цел n, имя a след), b = ст (проц b след), c = проц (c) c,

правильно построены. Однако

вид d = [1 : 10] d, e = об (цел, e)

не есть описание-вида. }

8. ИЗОБРАЖЕНИЯ

{ Изображения, например 3.14 или „абв", представляют собой конструкты, выдача которых не зависит ни от каких действий. В некоторых других языках они иногда называются „литералами" или „константами". }

8.0.1. *Синтаксис*

a) изображаемое в СРЕДЕ выдающее ЗНАЧЕНИЕ {5D, A341i} :
возможная последовательность пояснений {92a},
изображение ЗНАЧЕНИЯ {810a, 811a, 812a, 813a, 814a, 815a, 82a, b, c, 83a, -}.

{ Смысл изображения не зависит ни от какой среды. }

8.1. Изображения простого

{ Изображения-простого – это изображения арифметических значений, истинностных значений, литер и пустого значения, например 1, 3.14, истина, „а" и пустое. }

8.1.0.1. Синтаксис.

A) РАЗМЕРНОЕ :: длинное; короткое.

B)* ЧИСЛО :: натуральное число; рациональное число; действительное число.

a) изображение РАЗМЕРНОГО ЧИСЛОВОГО {a, 80a} :
символ РАЗМЕРНОЕ {94d},
изображение ЧИСЛОВОГО {a, 811a, 812a}.

b)* изображение простого:

изображение ПРОСТОГО {a, 811a, 812a, 813a, 814a};
изображение пустого значения {815a}.
{Пример:
a) длин 0}
8.1.0.2. Семантика.
Выдача W изображения-ЧИСЛОВОГО есть „естественное значение" {8.1.1.2, 8.1.2.2.a, b} его составляющего ЧИСЛА;
• требуется, чтобы W была не больше, чем наибольшее из различаемых {2.1.3.1.d} значений вида 'ЧИСЛОВОЕ'.
{Изображение-ЧИСЛОВОГО выдает арифметическое значение {2.1.3.1.a}, но выдаваемые различными изображениями-ЧИСЛОВОГО арифметические значения не обязательно различны (, например, 12.3.4 и 1.234_{10} + 2).}
8.1.1. *Изображения целого*
8.1.1.1. Синтаксис.
a) изображение целого {80a, 810a}: натуральное число {b}.
b) натуральное число {a, 812c, d, f, i, A341h}: последовательность десятичных цифр {c}.
c) десятичная цифра {b}: символ ЦИФРА {94b}.
{Примеры:
a) 4096 b) 4096
c) 4}
8.1.1.2. Семантика.
Естественным значением натурального-числа N является целое число, десятичное представление которого в эталонном языке есть это N {9.3.b}.
8.1.2. *Изображения вещественного*
8.1.2.1. Синтаксис.
a) изображение вещественного {80a, 810a}:
рациональное число {b}; действительное число {e}.
b) рациональное число {a, f}:
возможная целая часть {c}, дробная часть {d}.
c) целая часть {b}: натуральное число {811b}.
d) дробная часть {b}:
символ точка {94b}, натуральное число {811b}.
e) действительное число {a}: мантисса {f}, порядок {g}.
f) мантисса {e}: натуральное число {811b};
рациональное число {b}.
g) порядок {e}:
запись десятичного основания {h}, степень десяти {i}.
h) запись десятичного основания {g}:
символ на десять в степени {94b};
символ буква e {94a} либо символ буква e лат {94a}.
i) степень десяти {g}: возможные плюс или минус {j},
натуральное число {811b}.
j) плюс или минус {i}:
символ плюс {94c}; символ минус {94c}.

{ Примеры:

a) 0.00123 · 1.23e–3 b) 0.00123

c) 0 d) .00123

e) 1.23e–3 f) 123 · 1.23

g) e–3 h) $_{10}$ · *e*

i) –3 j) + · –}

8.1.2.2. Семантика.

a) Естественное значение V рационального-числа N определяется следующим образом:

· пусть I – естественное значение натурального числа его составляющей целой-части, если она есть, а иначе 0;

· пусть F – естественное значение натурального-числа его дробной-части P, деленное на 10 в степени количества десятичных-цифр, содержащихся в P;

· V – сумма, в смысле численного анализа, I и F.

b) Естественное значение действительного-числа N определяется следующим образом:

· пусть S – естественное значение ЧИСЛА его мантиссы;

· пусть E – естественное значение составляющего натурального-числа его порядка;

Случай A: Составляющий возможный-плюс-или-минус этого N содержит символ-минус:

· V – произведение, в смысле численного анализа, S и числа 1/10, возведенного в степень E;

Случай B: Прямой наследник рассматриваемого возможного-плюса-или-минуса содержит символ-плюс или пуст:

· V – произведение, в смысле численного анализа, S и числа 10, возведенного в степень E.

8.1.3. *Изображения логического*

8.1.3.1. Синтаксис.

a) изображение логического {80a} :

символ истина {94b}; символ ложь {94b}.

{ Примеры

a) истина · ложь }

8.1.3.2. Семантика.

Выдача изображения-логического есть истина (ложь), если его прямой наследник есть символ-истина (символ-ложь).

8.1.4. *Изображения литерного*

{ Изображения-литерного состоят из элемента-строки, заключенного между двумя символами-кавычка, например "a". Чтобы изобразить кавычку, используют символ-образ-кавычки (представляемый как ""), например" " " ." Поскольку синтаксис нигде не допускает, чтобы изображения-литерного или-строкового следовали одно за другим, это не приводит к двусмысленности. }

8.1.4.1. Синтаксис.

a) изображение литерного {80a} :

символ кавычка {94b}, элемент строки {b},
символ кавычка {94b}.

b) элемент строки {a, 83b} : элемент основного набора {c};
символ образ кавычки {94b};
добавочный элемент строки {d}.

c) элемент основного набора {b, 92c} : символ БУКВА {94a};
символ ЦИФРА {94b}; символ точка {94b};
символ открыть {94f}; символ закрыть {94f};
символ запятая {94b}; символ пробел {94b};
символ плюс {94c}; символ минус {94c}.

d) Для понятия 'добавочный элемент строки' {b, для которого в тексте настоящего стандарта не задано никакого гиперправила,} можно добавить порождающее правило, каждая из альтернатив которого является символом {1.1.3.1.f}, отличным от любого терминального порождения 'элемента основного набора' {c} и не являющимся 'символом кавычка'.

{Примеры:

a) "a" b) *a* • " " • ?

c) a • 1 • • (•) • , • . • + • –}

8.1.4.2. Семантика.

a) Выдача изображения-литерного есть естественное значение символа, наследного для его элемента-строки.

b) Естественным значением каждого отдельного символа, наследного для элемента-строки, является отличная от других литера.

{Литеры не имеют специфического смысла, за исключением того, что некоторые из них интерпретируются особым образом описаниями обмена (10.3). Элементы-основного-набора, включающие все литеры, необходимые для обмена, образуют некоторое минимальное множество, обеспечиваемое, как ожидается, во всех реализациях (2.2.2.c).}

8.1.5. *Изображение пустого значения*

{Изображение-пустого-значения можно использовать, чтобы присвоить пустое значение переменной-вида-ПРЕДСТАВИТЕЛЬ, например об ([ ] вещ, пуст) u := пустое.}

8.1.5.1. Синтаксис.

a) изображение пустого значения {80a} : символ пустое {94b}.

{Пример:

a) пустое}

8.1.5.2. Семантика.

Выдача всякого изображения-пустого-значения есть пустое.

8.2. Изображения битового

8.2.1. *Синтаксис*

A) ДВОИЧНОЕ :: двоичное; четверичное; восьмеричное; шестнадцатеричное.

a) изображение структуры содержащей ДОЛГУЮ
?ДОЛГУЮ букву алеф для выборки

вектора из логических в себе {a, 80a} :
символ длинное {94d} , изображение структуры
содержащей ?ДОЛГУЮ букву алеф для выборки
вектора из логических в себе {a, c} .

b) изображение структуры содержащей КРАТКУЮ
?КРАТКУЮ букву алеф для выборки
вектора из логических в себе {b, 80a} :
символ короткое {94d} , изображение структуры
содержащей ?КРАТКУЮ букву алеф для выборки
вектора из логических в себе {b, c} .

c) изображение структуры содержащей букву алеф для выборки
вектора из логических в себе {a, b, 80a} :
ДВОИЧНОЕ основание {d, e, f, g},
символ буква эр лат {94a},
последовательность ДВОИЧНЫХ цифр {h, i, j, k} ;
ДВОИЧНОЕ основание {d, e, f, g },
символ буква я {94a} ,
последовательность ДВОИЧНЫХ цифр {h, i, j, k} .

d) двоичное основание {c, A347b} : символ цифра два {94b} .

e) четверичное основание {c, A347b} :
символ цифра четыре {94b} .

f) восьмеричное основание {c, A347b} :
символ цифра восемь {94b} .

g) шестнадцатеричное основание {c, A347b} :
символ цифра один {94b} , символ цифра шесть {94b} .

h) двоичная цифра {c, i} :
символ цифра нуль {94b} ; символ цифра один {94b} .

i) четверичная цифра {c, j} : двоичная цифра {h} ;
символ цифра два {94b} ; символ цифра три {94b} .

j) восьмеричная цифра {c, k} : четверичная цифра {i} ;
символ цифра четыре {94b} ; символ цифра пять {94b} ;
символ цифра шесть {94b} ; символ цифра семь {94b}

k) *шестнадцатеричная цифра* {c} : *восьмеричная цифра* {j} ;
символ цифра восемь {94b} ; символ цифра девять {94b} ;
символ буква а лат {94a} ; символ буква бе лат {94a} ;
символ буква цэ лат {94a} ; символ буква де лат {94a} ;
символ буква е лат {94a} ; символ буква эф лат {94a} ;
символ буква а {94a} ; символ буква б {94a} ;
символ буква ц {94a} ; символ буква д {94a} ;
символ буква є {94a} ; символ буква ф {94a} .

1)* изображение битового : изображение БИТОВОГО {a, b, c} .
{ БИТОВОЕ :: структура содержащая букву алеф для
выборки вектора из логических в себе.}

m)*машинная цифра: ДВОИЧНАЯ цифра {h, i, j, k} .

{ Примеры:
a) длин 2r 101 · длин 2я101
b) кор 16r ffff · кор 16я ффff
c) 8r 231 · 8я231}

8.2.2. *Семантика*

a) Выдача V изображения-битового D определяется следующим образом:

- пусть W – естественное логическое значение {b} его составляющей последовательности-ДВОИЧНЫХ-цифр;
- пусть m – длина этого W;
- пусть n – значение Д размер бит {10.2.1.j}, где Д означает длин (кор), повторенное столько раз, сколько символов-длинное (символов-короткое) содержится в D;
- требуется, чтобы m было не больше n;
- V – структура {вида 'БИТОВОЕ'}, единственным полем которой является массив. имеющий

(i) паспорт ((1, n)) и

(ii) n элементов, выбираемых по (i), принимающих значение ложь, если $1 \leqslant i \leqslant n - m$, и $(i + m - n)$-е истинностное значение из {последовательности} W в противном случае.

b) Естественное логическое значение последовательности-ДВОИЧНЫХ-цифр S есть кратчайшая последовательность истинностных значений, которая, рассматривается как двоичное число (истина соответствует 1, а ложь – 0), совпадает с естественным целочисленным значением {c} последовательности S.

c) Естественное целочисленное значение последовательности-двоичных-(-четверичных-, -восьмеричных-, -шестнадцатеричных-) цифр S есть целое число, двоичным (четверичным, восьмеричным, шестнадцатеричным) представлением которого в эталонном языке является S {9.3.b}, где представления a(а), b(б), c(ц), d(д), e(е) и f(ф) рассматриваются как цифры, имеющие значения 10, 11, 12, 13, 14 и 15 соответственно.

8.3. Изображения строки

{ Изображения-строки дают удобный способ спецификации „строк" т.е. массивов вида 'вектор из литерных'.

Пример:

строк сообщение := "все в порядке". }

8.3.1. *Синтаксис*

a) изображение вектора из литерных {80a}:
   символ кавычка {94b}, возможная строка {b},
   символ кавычка {94b}.
b) строка {a} : элемент строки {814b}.
   последовательность элементов строки {814b}.
c)* изображение строки : изображение вектора из литерных {a}.

{ Примеры:
a) "абв" b) абв}

8.3.2. *Семантика*

Выдача изображения-строки D есть массив V, определяемый следующим образом:

- пусть n – число элементов-строки, содержащихся в D;
- паспорт V есть ((1, n));
- для i = 1, ..., n элементом этого V с индексом (i) является естественное значение {8.1.4.2.b} i-го составляющего символа строки этого D.

{"a" есть изображение-литерного, а не изображение-строки. Однако во всех сильных позициях, например строк S := "a", оно может векторизоваться до некоторого массива (6.6). Во всех остальных местах, где требуется массив, можно использовать ядро (5.5.1.1.a),

например,

об (лит, строк) cs := строк ("a").}

9. ЗНАКИ И СИМВОЛЫ

9.1. Знаки

{Знаки (9.1.1.f) – это символы (9.1.1.h), возможно предшествуемые пояснениями (9.2.1.a). Следовательно, пояснения могут встречаться между символами повсюду, где синтаксис порождает непрерывный ряд знаков. Однако в нескольких местах синтаксис порождает исключительно символы, а не знаки, а именно в изображениях (8), текстах-формата (10.3.4.1.1.a) и, конечно, внутри пояснений. Поэтому на таких местах пояснения встречаться не могут.}

9.1.1. *Синтаксис*

a) старт ВЫБИРАЮЩЕГО ОФОРМЛЕННЫЙ {34a}:
если (ВЫБИРАЮЩЕЕ) есть (выбирающее по логическому),
знак если ОФОРМЛЕННЫЙ {94f, -};
если (ВЫБИРАЮЩЕЕ) есть (ВАРИАНТНОЕ),
знак выбрать ОФОРМЛЕННЫЙ {94f, -}.

b) вход в собственно ВЫБИРАЮЩЕЕ ОФОРМЛЕННЫЙ {34e}:
если (ВЫБИРАЮЩЕЕ) есть (выбирающее по логическому),
знак то ОФОРМЛЕННЫЙ {94f, -};
если (ВЫБИРАЮЩЕЕ) есть (ВАРИАНТНОЕ),
знак в ОФОРМЛЕННЫЙ {94f, -}.

c) продолжатель ВЫБИРАЮЩЕГО ОФОРМЛЕННЫЙ {341}:
если (ВЫБИРАЮЩЕЕ) есть (выбирающее по логическому),
знак иначе если ОФОРМЛЕННЫЙ {94f, -};
если (ВЫБИРАЮЩЕЕ) есть (ВАРИАНТНОЕ),
знак либо выбрать ОФОРМЛЕННЫЙ {94f, -}.

d) выход собственно ВЫБИРАЮЩЕГО ОФОРМЛЕННЫЙ {341}:
если (ВЫБИРАЮЩЕЕ) есть (выбирающее по логическому),
знак иначе ОФОРМЛЕННЫЙ {94f, -};
если (ВЫБИРАЮЩЕЕ) есть (ВАРИАНТНОЕ),
знак либо ОФОРМЛЕННЫЙ {94f, -}.

e) финиш ВЫБИРАЮЩЕГО ОФОРМЛЕННЫЙ {34a}:
если (ВЫБИРАЮЩЕЕ) есть (выбирающее по логическому),
знак все ОФОРМЛЕННЫЙ {94f, -}:
если (ВЫБИРАЮЩЕЕ) есть (ВАРИАНТНОЕ),
знак конец выбора ОФОРМЛЕННЫЙ {94f, -}.
f) знак ПОНЯТИЕ:
возможная последовательность пояснений {92a},
символ ПОНЯТИЕ {94a, b, c, d, e, f, g, h}.
g)* знак : знак ПОНЯТИЕ {f}.
h)* символ : символ ПОНЯТИЕ {94a, b, c, d, e, f, g, h}.

9.2. Примечания и прагматы

{Всякое пояснение есть или примечание, или прагмат. Семантика пояснений не задается, и, следовательно, смысл (2.1.4.1.a) любой программы полностью не зависит от их присутствия. В действительности предполагается, что все примечания должны полностью игнорироваться реализацией, их единственное назначение – помочь человеку, интерпретирующему программу.

С другой стороны, прагматы могут снабжать реализацию некоторым количеством информации, затрагивающей какие-то аспекты смысла данной программы, не определенные настоящим стандартом, например:

- действие, которое следует предпринять при переполнении (2.1.4.3.h) или при нарушении правила об областях действия (, как в 5.2.1.2.b), например прагм включить проверку переполнения прагм, прагм выключить проверку переполнения прагм, прагм включить проверку областей действия прагм или прагм выключить проверку областей действия прагм;
- действие, которое следует предпринять по завершении процесса трансляции, например прагм только транслировать прагм, прагм выдать содержимое памяти прагм или прагм запустить прагм;
- что язык, который надо реализовать, является некоторым подъязыком или надъязыком Алгола 68, например прагм нерекурсивная прагм (для текста-процедуры, предполагаемой нерекурсивной);
- что трансляция может проверить истинность или попытаться доказать правильность некоторого утверждения, например:

цел a, b: чит ((a, b)) прагм здесь $a \geqslant 0 \wedge b > 0$ прагм:
цел q := 0, r := a;
пока $r \geqslant b$ прагм здесь $a = b \times q + r \wedge 0 \leqslant r$ прагм
цк (q +:= 1, r –:= b) кц
прагм здесь $a = b \times q + r \wedge 0 \leqslant r < b$ прагм.

Прагматы можно также использовать, чтобы указать реализации, что данный входной текст надо дополнить каким-то другим текстом или отредактировать, например:

- надо вызвать некоторую, предварительно оттранслированную порцию данной собственно-программы, например прагм с модулем из библиотеки прагм;

• данный входной текст продолжается на некотором другом носителе, например прагм читать из другого файла прагм;

• достигнут конец входного текста, например прагм конец прагм.

9.2.1. *Синтаксис*

A) ПОЯСНЕНИЕ :: прагмат; примечание.

a) пояснение {80a, 91f, A341b, h,
A348a, b, c, A349a, A34A b} : ПОЯСНЕНИЕ {b}.

b) ПОЯСНЕНИЕ {a}:
символ ПОЯСНЕНИЕ ОФОРМЛЕННЫЙ {94h, -},
возможная последовательность элементов
ПОЯСНЕНИЯ ОФОРМЛЕННЫХ {c},
символ ПОЯСНЕНИЕ ОФОРМЛЕННЫЙ {94h, -}.
{ОФОРМЛЕННЫЙ :: краткий; выделенный; стиля НОМЕР.}

c) элемент ПОЯСНЕНИЯ ОФОРМЛЕННЫЙ {b}:
элемент основного набора {814c};
добавочный элемент ПОЯСНЕНИЯ ОФОРМЛЕННЫЙ {d}.

d) Для каждого понятия, обозначаемого 'добавочным элементом ПОЯСНИЯ ОФОРМЛЕННЫМ' {c, для которого не задано никакого гиперправила в настоящем стандарте }, можно добавить порождающее правило, каждой альтернативой которого будет какой-нибудь символ {1.1.3.1.f}, отличный от любого терминального порождения понятия 'элемент основного набора' {8.1.4.1.c} и такой, что никакое терминальное порождение 'добавочного элемента ПОЯСНЕНИЯ ОФОРМЛЕННОГО' не совпадает с соответствующим 'символом ПОЯСНЕНИЕ ОФОРМЛЕННЫМ' {94h, -}.

{Так, например, примечанием может быть прим ♯ прим, но не ♯♯♯.}

{Примеры:

a) прагм листинг прагм •
♯ исходная программа будет распечатана ♯

c) 1 ◦ ?}

9.3. Представления

a) Всякий конструкт в строгом языке должен быть представлен в каком нибудь „языке представления", таком, как „эталонный язык", используемый в настоящем стандарте. Другие языки представления, специально приспособленные к предполагаемым склонностям интерпретирующего их человека, можно назвать языками „публикации".

{Назначение настоящего эталонного языка – использование его для представления собственно-программы и их наследных конструктов. Однако в разд. 10 он также применяется для определения стандартной языковой обстановки. Язык представления, приспособленный для использования в машине, называется языком реализации. Требования к языку реализации содержатся в приложении 2.}

b) Всякий „конструкт в языке представления" получается из терминального порождения T {1.1.3.2.f} соответствующего конструкта в строгом языке {1.1.3.2.e} заменой всех символов в T их представлениями, ко-

торые для данного эталонного языка заданы в подразделе 9.4.

{Так, например, собственно-программа в строгом языке с терминальным порождением

'символ начало стиля I'
'символ пропуск'
'символ конец стиля I'

создает в эталонном языке собственно-программу

начало пропуск конец.}

9.4. Эталонный язык

a) Настоящий эталонный язык предусматривает представления для различных символов, включая произвольно большое число символов-ОБОЗНАЧЕНИЕ {, где ОБОЗНАЧЕНИЕ :: СЛОВО; ИНДИКАНТ; ИНФИКС; ПРЕФИКС.}. Представления для некоторых из них заданы ниже {9.4.1}, и к ним можно добавить подходящие представления для символов-буква - АЛФАВИТА-стиля-НОМЕР и символов-префиксный-стиля-НОМЕР, а также любые терминальные порождения 'добавочных элементов ПОЯСНЕНИЯ ОФОРМЛЕННЫХ' {9.2.1.d} и 'добавочных элементов строки' {8.1.4.1.d}. Ни для одного из них не предусмотрено представлений {, но эти представления могут вводиться отдельными реализациями, чтобы весь набор имеющихся у них знаков был доступен для использования в качестве литер, чтобы обеспечить дополнительные или расширенные алфавиты для построения символов-СЛОВО и символов-ИНДИКАНТ и предусмотреть дополнительные символы для использования в качестве обозначений-операций}. Однако нет{, и не должно быть,}никакого представления для символа-буква-алеф и символа-первичный, за исключением представлений стандартного-вступления и других вступлений {10.1.3.Шаг 6}. {Относительно прочих символов-ОБОЗНАЧЕНИЕ см. 9.4.2. Имеются также отдельные порождаемые синтаксисом символы, для которых вообще не предусмотрено представлений, например символ-прагмат-краткий. Это не препятствует представлению таких символов в других языках представления.}

b) Если для некоторого символа задано более одного представления, то может выбираться любое из них. Более того, для какой-нибудь реализации эталонного языка достаточно обеспечить только одно из них. Также не обязательно предусматривать представление для каждого конкретного символа-ПРЕФИКСНЫЙ или символа-НЕПРЕФИКСНЫЙ при том условии, что обеспечивается представление по крайней мере для одной версии {10.1.3.Шаг 3} каждого обозначения-операции, описанного в стандартном-вступлении.

{Для некоторых разных символов заданы одинаковые или почти одинаковые представления; например, представление „:" задано для символа-признак-процедуры, символа-двоеточие и символа-вплоть-до, а „ : " — для символа-метка. Вне пределов примечаний, прагматов или изображений-строки из синтаксиса однозначно вытекает, какой из этих четырех символов представлен вхождением любого знака, подобного одному из этих представлений. В этом случае также можно без какой бы то ни было дву-

смысленности использовать для любого из них представление „ .. ", и для реализаций с ограниченным набором знаков это действительно может оказаться необходимым. Можно отметить, что в таких реализациях не возникнет никакой двусмысленности и в случае, когда „(/" и „/)" будут представлять символ-открыть-индексы-стиля-II и символ-закрыть-индексы-стиля-II соответственно.

Кроме того, некоторые из заданных представлений кажутся составными: так например, представление „:=" для символа-присвоить кажется состоящим из „ : ", представления для символа-признак-процедуры и т.п., и „:=" , представления для символа-равно и символа-определяется-как. Тем не менее из синтаксиса следует, что за пределами примечаний, прагматов и изображений-строки „ := " может встречаться только в качестве представления символа-присвоить (, поскольку „ = " не может употребляться в качестве представления обозначения-унарной-операции). Точно так же и другие заданные составные представления не приводят к двусмысленности.}

c) Тот факт, что о заданных {9.4.1.a} представлениях символов-буква-АЛФАВИТА обычно говорится как о малых буквах, не означает, что нельзя использовать соответствующие большие буквы.

d) Пробел, переход на новую строчку или страницу – это „особенности типографского набора". Эти особенности не принимаются во внимание и, *когда они появляются между символами какого-нибудь конструкта в эталонном языке*, они не оказывают влияния на смысл этого конструкта. Однако пробелы, содержащиеся в пределах изображения-строки или изображения-литерного, являются одним из представлений символа-пробел {9.4.1.b}, а не особенностью типографского набора. Когда представление символа в эталонном языке состоит из нескольких знаков {, например, до, :=}, эти знаки образуют один {неделимый }символ, и, если явно не оговорено противное {9.4.2.2.a, c}, особенности типографского набора не могут разделять их.

9.4.1. *Представления символов*

a) Символы для букв

символ	представление
символ буква а {814c, 82к, 942B, A346b}	а
символ буква б {814c, 82к, 942B, A344b}	б
символ буква в { 814c, 942B }	в
символ буква г { 814c, 942B }	г
символ буква д { 814c, 82к, 942B, A342f }	д
символ буква е { 812h, 814c, 82k, 942B, A343e}	е
символ буква ж { 814c, 942B}	ж
символ буква з {814c, 942B }	з
символ буква и {814c, 942B, A345b}	и
символ буква й {814c, 942B}	й
символ буква к {814c, 942B, A341f}	к
символ буква л {814c, 942B, A341f}	л

символ буква м {814c, 942B}	м
символ буква н {814c, 942B, A341h}	н
символ буква о {814c, 942B}	о
символ буква п {814c, 942B, A341f}	п
символ буква р {814c, 942B, A347c}	р
символ буква с {814c, 82к, 942B, A348a}	с
символ буква т {814c, 942B}	т
символ буква у {814c, 942B, A341f}	у
символ буква ф {814c, 942B}	ф
символ буква х {814c, 942B, A341f}	х
символ буква ц {814c, 942B}	ц
символ буква ч {814c, 942B}	ч
символ буква ш {814c, 942B}	ш
символ буква щ {814c, 942B}	щ
символ буква ъ {814c, 942B}	ъ
символ буква ы {814c, 942B}	ы
символ буква ь {814c, 942B}	ь
символ буква э {814c, 942B}	э
символ буква ю {814c, 942B}	ю
символ буква я {814c, 942B}	я
символ буква а лат {814c, 82k, 942B, A346b}	a
символ буква бе лат {814c, 82k, 942B, A344b}	b
символ буква це лат {814c, 82k, 942B, A348a}	c
символ буква де лат {814c, 82k, 942B, A342f}	d
символ буква е лат {812h, 814c, 82k, 942B, A343e}	e
символ буква эф лат {814c, 82k, 942B, A349a}	f
символ буква ге лат {814c, 942Б, A43Aa}	g
символ буква аш лат {814c, 942B}	h
символ буква и лат {814c, 942B, A345b}	i
символ буква йот лат {814c, 942B}	j
символ буква ка лат {814c, 942B, A341f}	k
символ буква эль лат {814c, 942B, A341f}	l
символ буква эм лат {814c, 942B}	m
символ буква эн лат {814c, 942B, A341h}	n
символ буква о лат {814c, 942B}	o
символ буква пэ лат {814c, 242B, A341f}	p
символ буква ку лат {814c, 942B, A341f}	q
символ буква эр лат {814c, 82c, 942B, A347c}	r
символ буква эс лат {814c, 942B, A341l}	s
символ буква тэ лат {814c, 942B}	t
символ буква у лат {814c, 942B}	u
символ буква ве лат {814c, 942B}	v
символ буква дубль ве лат {814c, 942B}	w
символ буква икс лат {814c, 942B, A341f}	x
символ буква игрек лат {814c, 942B, A341f}	y

символ буква зэт лат {814c, 942B, A342d}	z

b) Символы для изображений

символ	представление
символ цифра нуль {811c, 814c, 82h, 942C}	0
символ цифра один {43b, 811c, 814c, 82g, h, 942C}	1
символ цифра два {43b, 811c, 814c, 82d, i, 942C}	2
символ цифра три {43b, 811c, 814c, 82i, 942C}	3
символ цифра четыре {43b, 811c, 814c, 83e, j, 942C}	4
символ цифра пять {43b, 811c, 814c, 82j, 942C}	5
символ цифра шесть {43b, 811c, 814c, 82g, j, 942C}	6
символ цифра семь {43b, 811c, 814c, 82j, 942C}	7
символ цифра восемь {43b, 811c, 814c, 82f, k, 942C}	8
символ цифра девять {43b, 811c, 814c, 82k, 942C}	9
символ точка {812d, 814c, A343d}	.
символ на десять в степени {812h}	/
символ истина {813a}	true истина
символ ложь {813a}	falle ложь
символ кавычка {814a, 83a}	"
символ образ кавычки {814b}	" "
символ пробел {814c}	␣
символ запятая {814c}	,
символ пустое {815a}	empty пустое

c) Символы для обозначения операций

символ	представление
символ или {942H}	∨
символ и {942H}	∧
символ амперсенд {942H}	&
символ не равно {942H}	≠
символ меньше {9421}	<
символ не больше {942H}	⩽
символ не меньше {942H}	⩾
символ больше {9421}	>
символ разделить {9421}	/
символ от до {942H}	÷
символ процент {942H}	%
символ элемент {942H}	□
символ меньшее целое {942H}	⌊
символ большее целое {942H}	⌈
символ плюс и на {942H}	⊥
символ не {942H}	¬
символ тильда {942H}	~
символ вниз {942H}	↓
символ вверх {942H}	↑
символ плюс {812j, 814c, 942H, A342e}	+
символ минус {812j, 814c, 942H, A342e}	

символ равно {942I}		=
символ умножить {942I}		X
символ звездочка {942I}		*
символ присвоить направо {942J}		=:
символ присвоить {44f, 521a, 942J}		:=

d) Символы для описаний

символ		представление
символ определяется как {42b, 43b, 44c, 45c}	=	
символ длинное {810a, 82a}	long	длин
символ короткое {810a, 82b}	short	кор
символ имя {46c}	ref	имя имени
символ локальный {523a, b}	loc	лок
символ глобальный {523a, b}	heap	глоб
символ структура {46d}	struct	ст структ
символ подвижное {46g}	flex	подв
символ процедура {44b, 46o}	proc	проц
символ объединение {46s}	union	об
символ операция {45a}	op	оп
символ приоритет {43a}	prio	прио
символ вид {42a}	mode	вид

e) Стандартные виды

символ		представление
символ целое {942E}	int	цел
символ вещественное {942E}	real	вещ
символ логическое {942E}	bool	лог
символ литерное {942E}	char	лит
символ формат {942E}	format	формат
символ пустое значение {942E}	void	пуст
символ комплексное {942E}	compl	компл
символ битовое {942E}	bits	бит
символ слоговое {942E}	bytes	слог
символ строковое {942E}	string	строк
символ сема {942E}	sema	сема
символ файл {942E}	file	файл
символ канал {942E}	channel	канал

f) Синтаксические символы

символ		представление
символ начало выделенный {133d}	begin	
символ конец выделенный {133d}	end	
символ начало краткий {133d, A348b, A34Ab}	(	
символ конец краткий {133d, A348b, A34Ab}	)	

символ начало стиля I {133d}		начало
символ конец стиля I {133d}		конец
символ начало стиля II {133d}		нач
символ конец стиля II {133d}		кон
символ а также {133c, 33b, f, 34h, 41a, b, 46e, i, q, t, 532b, 541e, 543b, A348b, A34Ac, d}	,	
символ продолжать {32b}	;	
символ завершить {32b}	exit	выход
символ метка {32c}	:	
символ параллельно {33c}	par	пар
символ открыть {814c}	(	
символ закрыть {814c}	)	
символ если выделенный {91a}	if	
символ то выделенный {91b}	then	
символ иначе если выделенный {91c}	elif	
символ иначе выделенный {91d}	else	
символ все выделенный {91e}	fi	
символ выбрать выделенный {91a}	case	
символ в выделенный {91b}	in	
символ либо выбрать выделенный {91c}	ouse	
символ либо выделенный {91d}	out	
символ конец выбора выделенный {91e}	esac	
символ если краткий {91a}	(	
символ то краткий {91b}	|	
символ иначе если краткий {91c}	|:	
символ иначе краткий {91d}	|	
символ все краткий {91e}	)	
символ выбрать краткий {91a}	(	
символ в краткий {91b}	|	
символ либо выбрать краткий {91c}	|:	
символ либо краткий {91d}	|	
символ конец выбора краткий {91e}	)	
символ если стиля I {91a}		если
символ то стиля I {91b}		то
символ иначе если стиля II {91c}		инес
символ иначе стиля I {91d}		иначе
символ все стиля I {91e}		все
символ выбрать стиля {91a}		выб
символ в стиля I {91b}		в
символ либо выбрать стиля I {91c}		ливыб
символ либо стиля I {91d}		либо
символ конец выбора стиля I {91e}		быв
символ двоеточие {34j, k}	:	
символ открыть индексы краткий {133e}	[	

символ закрыть индексы краткий {133e}	]		
символ открыть индексы стиля I {133e}	(		
символ закрыть индексы стиля I {133e}	)		
символ вплоть до {46j, k, l, 532f}	:		
символ с {532g}	@	at	с
символ есть {522b}	:=:	is	есть
символ не есть {522b}	:≠:	:/=:	isnt несть
символ нил {524a}	∘	nil	нил
символ из {531a}		of	из
символ признак процедуры {541a, b}	:		
символ на выделенный {544b}	goto		
символ на краткий {544b}			на
символ иди выделенный {544b}	go		
символ пропуск {552a}	skip	пропуск	скип
символ форматор {A341a}	$		ф

g) Символы для циклов

символ	представление	
символ для выделенный {35b}	for	
символ нижний предел выделенный {35d}	from	
символ с шагом выделенный {35d}	by	
символ верхний предел выделенный {35d, 544b}	to	
символ пока выделенный {35g}	while	
символ цикл выделенный {35h}	do	
символ конец цикла выделенный {35h}	od	
символ для краткий {35b}		для
символ нижний предел краткий {35d}		от
символ с шагом краткий {35d}		шаг через
символ верхний предел краткий {35d}		до
символ пока краткий {35g}		пока
символ цикл краткий {35h}		цк
символ конец цикла краткий {35h}		кц

h) Символы для пояснений

символ	представление	
символ примечание краткий {92b}	¢	
символ примечание выделенный {92b}	comment	
символ примечание стиля I {92b}	co	
символ примечание стиля II {92b}	#	
символ примечание стиля III {92b}		прим
символ примечание стиля IIII {92b}		#
символ прагмат выделенный {92b}	pragmat	
символ прагмат стиля I {92b}	pr	
символ прагмат стиля II {92b}		прагм

9.4.2 *Символы прочих обозначений*

9.4.2.1. Метасинтаксис.

A) СЛОВО {D, F, K, 48a, b, c, d} :: БУКВА {B} ;
СЛОВО БУКВА { B }; СЛОВО ЦИФРА {C}.

B) БУКВА {A} :: буква АЛФАВИТА {94a};
буква алеф { - }; буква АЛФАВИТА стиля НОМЕР { -}.

C) ЦИФРА {A} :: цифра нуль {94b} ; цифра один {94b} ;
цифра два { 94b} ; цифра три { 94b} ;
цифра четыре { 94b} ; цифра пять { 94b} ;
цифра шесть {94b} ; цифра семь {94b} ;
цифра восемь { 94h} ; цифра девять {94b}.

D) ИНДИКАНТ {48a, b } :: выделенное СЛОВО {A, -};
?РАЗМЕРНОЕ СТАНДАРТНОЕ {E}.

E) СТАНДАРТНОЕ {D} : целое {94e} ; вещественное { 94e} ;
логическое {94e } ; литерное { 94e } ; формат {94e} ;
пустое значение {94e} , комплексное { 94e} ;
битовое {94e} ; слоговое {94e} ;
строковое {94e} , сема {94e} ; файл {94e } ; канал { 94e}.

F) ИНФИКС {48a, b} :: выделенное СЛОВО { A, -} ;
ЗНАК { G } ?ПРИСВОЕНИЕ { J } ;
ЗНАК { G } перед НЕПРЕФИКСНЫМ { I } ?ПРИСВОЕНИЕ {J}.

G) ЗНАК {F} :: ПРЕФИКСНЫЙ {H} ; НЕПРЕФИКСНЫЙ { I}.

H) ПРЕФИКСНЫЙ { G, K } :: или {94c} ; и {94c} ; амперсенд {94c} ;
не равно {94c} ; не больше {94c} ; не меньше {94c };
от до {94c} , процент {94c} ; элемент {94c} ;
меньшее целое {94c}; большее целое {94c} ;
плюс и на {94c} ; не { 94c} ;
тильда {94c} ; вниз {94c} ;
вверх {94c} ; плюс {94c} ; минус {94c} ; префиксный стиля
НОМЕР { - }.

I) НЕПРЕФИКСНЫЙ {F, G, K } :: меньше {94c} ; больше {94c} ;
разделить {94c }; равно {94c }; умножить {94c} ;
звездочка { 94c }.

J) ?ПРИСВОЕНИЕ {F, K} :: перед присвоить { 94c } ;
перед присвоить направо {94c} ; ПУСТО.

K) ПРЕФИКС {48a, b} :: выделенное СЛОВО {A, -} ;
ПРЕФИКСНЫЙ {H} ?ПРИСВОЕНИЕ {J}; ПРЕФИКСНЫЙ {H}
перед {9422e } НЕПРЕФИКСНЫМ {I} ?ПРИСВОЕНИЕ { J }.

L) АЛФАВИТ {B} :: а; б; в; г; д; е; ж; з; и; й; к; л; м; н; о; п; р; с; т; у; ф; х;
ц; ч; ш; щ; ъ; ы; ь; э; ю; я;
а лат; бе лат; це лат; де лат; е лат; эф лат; ге лат; аш лат; и лат;
йот лат; ка лат; эль лат; эм лат; эн лат; о лат; пэ лат; ку лат;
эр лат; эс лат; тэ лат; у лат; ве лат; дубль ве лат; икс лат; игрек
лат; зэт лат.

M)*ЗНАКОВЫЙ :: ЗНАК {G} ; ЗНАК {G} перед

НЕПРЕФИКСНЫМ {1}.

{Метапонятие „АЛФАВИТ" предусмотрено в дополнение к метапонятию „ЛИТЕРА", чтобы способствовать определению вариантов Алгола 68 (1.1 5 b).}

9.4.2.2. Представления.

a) Представление каждого символа-СЛОВО, не заданное выше {9.4.1}, состоит из знаков, соответствующих по порядку 'БУКВАМ' или 'ЦИФРАМ', содержащимся в этом 'СЛОВЕ'. Эти знаки могут разделяться особенностями типографского набора{9.4.d}. Знаком, соответствующим каждой 'БУКВЕ' ('ЦИФРЕ'), является представление данного символа-БУКВА (символа-ЦИФРА). {Например, представлением символа-буква-х-цифра-один является х 1, что можно написать и как х 1. Символы-СЛОВО используются для идентификаторов и указателей-поля.}

b) Представление каждого символа-выделенное-СЛОВО, если оно вообще существует, состоит из знаков, соответствующих по порядку 'БУКВАМ' или 'ЦИФРАМ', содержащимся в этом 'СЛОВЕ' {, но без разделяющих их особенностей типографского набора}. Знак, соответствующий каждой 'БУКВЕ' ('ЦИФРЕ'), аналогичен знаку, представляющему соответствующий символ-БУКВА (символ-ЦИФРА), что воспроизводится в настоящем стандарте соответствующей буквой (цифрой). {Приемлемы также и другие способы указания этой аналогии, распознаваемые без дополнительных разъяснений, например, лицо, лицо, ЛИЦО, **лицо** и 'лицо' могут все быть представлениями для символа-выделенные-буква-л-буква-и-буква-ц-буква-о.}

Однако представление никакого символа-выделенное-СЛОВО не может быть таким же, как любое представление любого другого символа}; и, следовательно, может существовать конечное множество символов-выделенное-СЛОВО, которые не имеют представления, так, например, нет представления для символа-выделенные-буква-в-буква-е-буква-щ, потому что вещ есть представление для символа-вещественное; отметим, что число свободных символов-выделенное-СЛОВО остается произвольно большим.} Если согласно принятым соглашениям, данная последовательность знаков может быть либо представлением одного символа-выделенное-СЛОВО, либо конкатенацией представлений двух или более других символов, то она всегда должна разбираться как один этот символ {; включение пробела может всегда вынудить другую интерпретацию; например имявещ — один символ, а имя вещ всегда должны быть двумя}. Символы-выделенное-СЛОВО используются для индикаторов-вида и для обозначений-операции.}

c) Представление каждого символа-РАЗМЕРНОЕ-? РАЗМЕРНОЕ-СТАНДАРТНОЕ состоит из представления соответствующего символа-РАЗМЕРНОЕ с возможно следующими за ними особенностями типографского набора, за которыми идет представление соответствующего символа-?РАЗМЕРНОЕ-СТАНДАРТНОЕ. Например, представление символа-длинное-вещественное будет длин вещ или, возможно, 'длин' 'вещ' (, но, согласно сказанному в п. b выше, не длинвещ и не 'длинвещ', так как последние могли бы быть представлениями для символа-выделенные-буква-д-буква-л-буква-и-

буква-н-буква-в-буква-е-буква-щ). Символы-?РАЗМЕРНОЕ-СТАНДАРТНОЕ используются для индикаторов-вида.}

d) Представление каждого символа-ЗНАКОВЫЙ-перед-присвоить (символа-ЗНАКОВЫЙ-перед-присвоить-направо) состоит из знака или знаков, представляющих соответствующий символ-ЗНАКОВЫЙ, за которыми, без вмешательства особенностей типографского набора, идут знаки, представляющие символ-присвоить (символ-присвоить-направо). {Например, представлением символа-плюс-перед-присвоить является +:=. Символы-ЗНАКОВЫЙ-перед-?ПРИСВОЕНИЕМ используются для обозначений-операции.}

e) Представление каждого символа-ЗНАК-перед-НЕПРЕФИКСНЫМ состоит из знака, представляющего соответствующий символ-ЗНАК, за которым без вмешательства особенностей типографского набора, идет знак, представляющий соответствующий символ-НЕПРЕФИКСНЫЙ. {Например, представлением символа-от-до-перед-умножить является ÷×. Символы-ЗНАК-перед-НЕПРЕФИКСНЫМ2 могут быть только обозначениями-бинарной-операции }

10. СТАНДАРТНАЯ ЯЗЫКОВАЯ ОБСТАНОВКА

{Данная „стандартная языковая обстановка" охватывает составляющие ВНЕШНИЕ-вступления, системные-задачи и собственные заключения всякого текста-программы }

10.1 Тексты программ

{Программист имеет дело с собственно-программами (10.1 1.g). Последние всегда включены в текст-программы (10.1 1.a), содержащий также стандартное-вступление, библиотечное-вступление, зависящее от реализации системное-вступление и системные-задачи, соответствующие операционной обстановке, возможно некоторые другие собственно-программы, одно или несколько собственных вступлений (по одному для каждой из собственно-программ) и одно или несколько собственных-заключении.}

10.1.1. *Синтаксис*

A) ВНЕШНЕЕ :: стандартное,
библиотечное; системное; собственное.

B) СТОП :: буква эс лат буква тэ лат буква о лат
буква пэ лат для метки буква с буква т буква о буква п
для метки.

a) текст программы:
знак начало ОФОРМЛЕННЫЙ {94f},
вступления в новом {ПУСТО} с СЛОЕМ1 {b},
знак параллельно {94f},
УПАКОВКА задач в новом {ПУСТО} с СЛОЕМ1 {d},
знак конец ОФОРМЛЕННЫЙ {94f}.

b) вступления в СРЕДЕ1 {a}:
стандартное вступление с !ОПИСАНИЯМИ1 в СРЕДЕ1 {c},

библиотечное вступление с ?ОПИСАНИЯМИ2 в СРЕДЕ1 {c},
системное вступление с ?ОПИСАНИЯМИ3 в СРЕДЕ1 {c},
если (СРЕДА1) есть (новое ПУСТО с новыми
!ОПИСАНИЯМИ1 ?ОПИСАНИЯМИ2 ?ОПИСАНИЯМИ3).

c) ВНЕШНЕЕ вступление с ?ОПИСАНИЯМИ1 в СРЕДЕ1 {b, f}:
кортеж с ?ОПИСАНИЯМИ1 сильно
выдающий пустое значение в СРЕДЕ1 {32b},
знак продолжать {94f};
если (?ОПИСАНИЯ1) есть (ПУСТО), ПУСТО.

d) задачи в СРЕДЕ1 {a}:
список системных задач в СРЕДЕ1 {e},
знак а также {94f},
список УПАКОВОК задач пользователя в СРЕДЕ1 {f}.

e) системная задача в СРЕДЕ1 {d}:
основа в СРЕДЕ1
сильно выдающая пустое значение {32d}.

f) задача пользователя в СРЕДЕ1 {d}:
собственное вступление с !ОПИСАНИЯМИ в СРЕДЕ2 {c},
УПАКОВКА собственно программы в СРЕДЕ2 {g},
знак продолжать {94f},
собственное заключение в СРЕДЕ2 {i},
если (СРЕДА2) есть (СРЕДА1 с
новыми !ОПИСАНИЯМИ {и} СТОПОМ).

g) собственно программа в СРЕДЕ2 {f}:
групповое определение меток через ?МЕТКИ3
в СРЕДЕ2 с новыми ?МЕТКАМИ3 {h},
ЗАКРЫТОЕ предложение в СРЕДЕ2 с
новыми ?МЕТКАМИ3 сильно выдающее
пустое значение {31a, 33a, c, 34a, 35a}.

h) групповое определение меток
через ?МЕТКИ в СРЕДЕ {g, h}:
если (?МЕТКИ) есть (ПУСТО), ПУСТО;
если (?МЕТКИ) есть (МЕТКА1 ?МЕТКИ1),
определение метки через МЕТКУ1 в СРЕДЕ {32c},
групповое определение меток
через ?МЕТКИ в СРЕДЕ {h}.

i) собственное заключение в СРЕДЕ2 {f}:
кортеж с СТОПОМ сильно выдающий
пустое значение в СРЕДЕ2 {32b}.

{Примеры:

a) (с стандартное-вступление с; с библиотечное-
вступление с; с системное-вступление с;
пар начало с системная-задача-1 с,
с системная-задача-2 с,
(с собственное-вступление с;

(старт: начать: нач пропуск кон);
с собственное-заключение с),
(с другая-задача-пользователя с)
конец)

b) с стандартное-вступление (10.2, 10.3) с;
с библиотечное-вступление с;
с системное-вступление (10.4.1) с;

d) с системная-задача-1 (10.4.2.a) с,
с системная-задача-2 с,
(с собственное-вступление с;
(старт: начать: нач пропуск кон);
с собственное-заключение с),
(с другая-задача-пользователя с)

f) с собственное-вступление (10.5.1) с;
(старт: начать: нач пропуск кон);
с собственное-заключение (10.5.2) с

g) старт: начать: нач пропуск кон

h) старт: начать:

i) stop: стоп: снять (станд ввод);
снять (станд вывод); снять (станд обмен)}

10.1.2. *Соответствие языковой обстановке*

a) Программа в строгом языке должна быть подобна {1.1.3.2.k } некоторому тексту-программы, составляющие ВНЕШНИЕ-вступления и собственные-заключения которого задаются ниже в настоящем разделе.

{Удобно говорить о стандартном-вступлении, библиотечном-вступлении, собственно-программе и т.п. некоторой программы, когда рассматриваются фрагменты этой программы, соответствующие составляющему стандартному-вступлению и т.п. соответствующего текста-программы. }

b) Составляющим стандартным-вступлением всех текстов-программы является стандартное-вступление, представление которого получается {10.1.3} из форм, данных в подразделах 10.2 и 10.3.

c) Составляющее библиотечное-вступление в настоящем стандарте не задано ни для какого текста-программы {, но должно быть задано для каждой реализации; синтаксисом 'текста программы' гарантируется, что никакое описание, содержащееся в любом библиотечном-вступлении, не может противоречить никакому описанию, содержащемуся в стандартном-вступлении, указанном выше. }

d) Составляющим системным-вступлением (списком-системных-задач) всех текстов-программ является системное-вступление (список-системных задач), представление которого получается из форм, данных в подразделе 10.4, с возможным добавлением других форм, не задаваемых в настоящем стандарте {, но которые задаются, чтобы удовлетворить требованиям операционной обстановки в каждой реализации }.

e) Каждым составляющим собственным-вступлением (собственным-заключением) всех текстов-программ является собственно-вступление

(собственно-заключение), представление которого получается из форм, данных в подразделе 10.5, с возможным добавлением других форм, не задаваемых в настоящем стандарте {, но которые задаются в каждой реализации}.

10.1.3. *Способ описания стандартной языковой обстановки*

Представление ВНЕШНЕГО-вступления, системной-задачи или собственного-заключения получается за счет внесения изменений в каждую из форм в соответствующем разделе этой главы посредством следующих шагов:

Шаг 1: Если данная форма F начинается с {символа-операция} оп и за ним идет один из знаков P, Q, R или E, то F заменяется рядом новых форм – копий формы F, в которых этот {следующий за оп} знак в каждой соответствующей новой форме заменяется (все другие вхождения этого знака в F заменяются) на:

Случай A: Знак есть P:

• –, +, ≮ X, * ≯ или /
(–, +, X или /);

Случай B: Знак есть Q:

• ≮ minusab, минпр, – . = ≯,
≮ plusab, плюспр, + := ≯,
≮ timesab, умпр, X := , * := ≯ или
≮ divab, делпр, / := ≯
(– := , + := , X := или / :=);

Случай C: Знак есть R:

• ≮ <, lt, мш ≯, ≮ ≤, < =, le, нб ≯,
≮ =, eq, рв ≯, ≮ ≠, / =, ne, нр ≯,
≮ ≥, > =, ge, нм ≯ или ≮ >, gt, бш ≯
(<, ≤, =, ≠, ≥ или >);

Случай D: Знак есть E:

• ≮ =, eq, рв ≯ или ≮ ≠, / =, ne, нр ≯
(= или ≠);

Шаг 2: Если в некоторой форме, возможно полученной на предыдущем шаге, встречается знак ⁇ , за которым идет какой-нибудь ИНДИКАТОР (указатель-поля) l, то это вхождение ⁇ удаляется, а каждый ИНДИКАТОР (указатель-поля), подобный (1.1.3.2.k) этому l и содержащийся в любой из форм, заменяется копией одного и того же ИНДИКАТОРА (указателя-поля), не встречающегося нигде в данной программе, после чего шаг 2 предпринимается снова.

Шаг 3: Если данная форма F, возможно измененная или полученная на предыдущих шагах, начинается с {символа-операция} ор, за которым идет разделенная символами-а-также-цепочка символов-АФФИКС, и эта цепочка заключена между ≮ и ≯, то эта F заменяется рядом разных ее „версий", причём каждая из них является копией формы F и в ней данная цепочка вместе с охватывающими ее знаками ≮ или ≯ заменена одним из входящих в нее символов-

АФФИКС {: однако никакая реализация не обязана обеспечивать более одной из таких версий 9.4.b)}.

Шаг 4: Если в данной форме, возможно измененной или полученной на предыдущих шагах, встречается заключенная между ⊰ и ⊱ последовательность S символов и в S встречаются L int, Д цел, L real, Д вещ, L compl, Д компл, L bits, Д бит, L bytes или Д слог, то S заменяется разделенной символами-а-также цепочкой достаточного числа последовательностей, причем n-я из них является копией последовательности S и в ней каждой вхождение Д (Д, L, L, У, С) заменено повторенным n – 1 раз длин (длин, long, long, удл, укр); за этой цепочкой идут символ-а-также и другая разделенная символами-а-также цепочка достаточного числа последовательностей, причем m-я из них является копией последовательности S и в ней каждое вхождение Д (Д, L, L, У, С) заменено повторенным m раз кор (кор, short, short, укр, удл); после этого удаляются охватывающие S знаки ⊰ и ⊱.

Шаг 5: Если в данной форме F, возможно измененной или полученной на предыдущих шагах, встречаются L int (Д цел, L real, Д вещ, L compl, Д компл, L bits, Д бит, L bytes, Д слог), то эта F заменяется последовательностью достаточного числа новых форм, причем n-я из них является копией формы F и в ней каждое вхождение Д (Д, L, L, У, С) заменено повторенным n – 1 раз длин (длин, long, long, удл, укр) а каждое вхождение длин Д (длин Д, long L, long L) заменено повторенным n раз длин (длин, long, long); за этой последовательностью идет другая последовательность достаточного числа новых форм; причем m-я из них является копией формы F и в ней каждое вхождение Д (Д, L, L, У, С) заменено повторенным m раз кор (кор, short, short, укр, удл), а каждое вхождение длин Д (длин Д, long L, long L) заменено повторенным m – 1 раз кор (кор, short, short).

Шаг 6: Каждое вхождение F (ПЕРВ) в любую форму, возможно измененную или полученную на предыдущих шагах, заменяется некоторым представлением символа-буква-алеф (символа-первичный) {9.4.a}

Шаг 7: Если в любой форме, возможно измененной или полученной на предыдущих шагах, встречается последовательность представлений, начинающаяся и оканчивающаяся знаками с, то эта последовательность, называемая „псевдопримечанием", заменяется некоторым представлением описателя или замкнутого-предложения, неформально задаваемого данной последовательностью.

Шаг 8: Если в любой форме, возможно измененной или полученной на предыдущих шагах, встречается текст-процедуры, вызов которого включает действия с вещественными числами, то этот текст-процедуры, можно заменить любым другим текстом-процедуры, вызов которого имеет приблизительно тот же эффект ;{степень такого

приближения оставлена в настоящем стандарте не определенной (см. также 2.1.3.1.e)} .

Шаг 9: В случае ВНЕШНЕГО-вступления в его конце добавляется форма, состоящая из символа-пропуск, за которым идет символ-продолжать {пропуск;} .

Шаг 10: Если в любой форме, возможно измененной или полученной на предыдущих шагах, встречается последовательность

$$A1\#P1\#A2\#P2\#...An\#Pn\#A_{n+1} ,$$

где A_1 – идентификаторы, а P_i – последовательности-элементов-пояснений, то эта последовательность заменяется на последовательность

$$A1A2...AnA_{n+1}\#A1P1A2P2...AnPnA_{n+1}\#$$

Шаг 11: Если некоторая форма F, возможно измененная или полученная на предыдущих шагах, представляет собой описание некоторого индикатора 1, представление которого содержит буквы русского алфавита, перечисленные в п. 2.1 приложения 2, то дополнительно вводится еще одна форма, получаемая из F заменой всех таких букв в первом вхождении индикатора 1 в F на соответствующие буквы латинского алфавита.

{Термин „достаточное число", использованный выше в шагах 4 и 5, подразумевает, что никакая собственно-программа не может иметь другого смысла или не порождаться синтаксисом только из-за недостаточности этого числа.}

Повсюду {в описаниях обмена}, где в изображении-литерного или изображении-строки встречаются представления $_{10}$ (\, ⊥), они должны интерпретироваться как представления элементов-строки {8.1.4.1.b}, применяемые для указания {символов} „на десять в степени" (некоторой альтернативной формы „на десять в степени"{, если она есть}, „плюс и на") на внешних носителях. { Ясно, что эти представления выбраны из-за их подобия представлениям символа-на-десять-в-степени (9.4.1.b) и символа-и-плюс-и-на (9.4.1.c), но на носителях, для которых эти литеры не доступны, надо выбирать другие элементы-строки (и символ-буква-е-лат с символом-буква-и-лат – очевидные кандидаты).}

{ Все описания в данном разделе предназначены для того, чтобы четко определить их действие. То же действие вполне можно получить и более эффективными способами.}

10.2. Стандартное вступление

{ Описания данного стандартного-вступления включают „запросы к обстановке", поставляющие информацию о конкретных особенностях данной реализации (2.2.2.c), „стандартные виды", „стандартные обозначения операций и функции", „операции синхронизации" и (данные в подразделе 10.3) „описания обмена". }

10.2.1. *Запросы к обстановке*

а) цел число длин цел = с 1 плюс число добавочных удлинений целых чисел {2.1.3.1.d} с;

цел int lengths = число длин цел;

b) цел число кор цел = с 1 плюс число добавочных укорочений целых чисел {2.1.3.1.d} с;
цел int shorths = число кор цел;

c) Д цел Д макс цел = с наибольшее Д цел значение {2.2.2.b} с;
Д цел L max int = Д макс цел;

d) цел число длин вещ = с 1 плюс число добавочных удлинений вещественных чисел {2.1.3.1.d} с;
цел real lengths = число длин вещ;

e) цел число кор вещ = с 1 плюс число добавочных укорочений вещественных чисел {2.1.3.1.d} с;
цел real shorths = число кор вещ;

f) Д вещ макс вещ = с набольшее Д вещ значение {2.2.2.b} с;
Д вещ L max real = Д макс вещ;

g) Д вещ Д точность вещ = с наименьшее Д вещ значение, такое, что как Д 1 + Д точность вещ > Д 1, так и Д1 – Д точность вещ < Д1 {2.2.2.b} с;
Д вещ L small real = Д точность вещ;

h) цел число длин бит = с 1 плюс число добавочных размеров {j} битовых с;
цел bits lengths = число длин бит;

i) цел число кор бит = с 1 плюс число добавочных размеров {j} коротких битовых с:
цел bits shorths = число кор бит;

j) цел Д размер бит = с число элементов в Д бит; см. Д бит {10.2.2.g}; это число увеличивается (уменьшается вместе с „размером", т.е. с числом 'длин' (взятым с обратным знаком числом 'кор'), из которых составлено 'Д', до достижения определенного размера, а именно „числа добавочных размеров" (взятого с обратным знаком „числа добавочных размеров коротких") битовых, после чего оно остается постоянным с;
цел L bits width = Д размер бит;

k) цел число длин слог = с 1 плюс число добавочных размеров {m} слоговых с;
цел bytes lengths = число длин слог;

l) цел число кор слог = с1 плюс число добавочных размеров коротких {m} слоговых с;
цел bytes shorths = число кор слог;

m) цел Д размер слог = с число элементов в Д слог; см. Д слог {10.2.2.h}; это число увеличивается (уменьшается) вместе с „размером", т.е. с числом 'длин' (взятым с обратным знаком числом 'кор'), из которых составлено 'Д', вплоть до достижения определенного размера, а именно „числа добавочных размеров" (взятого с обратным знаком „числа добавочных размеров коротких") слоговых, после чего оно остается постоянным с;
цел L bytes width = Д размер слог;

n) оп ⫷ абс, abc ⫸ = (лит а) цел: с целочисленный эквивалент {2.1.3.1g} литеры а с;

o) оп ⩤ пред, repr ⩥ = (цел a) лит: c та литера 'x', если она существует, для для которой абс x = a c;

p) цел макс лит = c наибольший целочисленный эквивалент {2.1.3.1.g} литеры c:
цел max abs char = макс лит;

q) лит заполнитель = c некоторая литера c;
лит null character = заполнитель;

r) лит да = c литера, используемая для представления 'истина' во время обмена {10.3.3.1.a, 10.3.3.2.a} c;
лит flip = да;

s) лит нет = c литера, используемая для представления 'ложь' во время обмена c;
лит flop = нет;

t) лит литера ошибки = c литера, используемая во время обмена для представления непреобразуемых арифметических значений {10.3.2.1.b, c, d, e, f} c;
лит errorchar = литера ошибки;

u) лит пробел = " "; лит blank = пробел;

10.2.2. *Стандартные виды*

a) вид пуст = c фактический-описатель, специфицирующий вид 'пустое значение' c;
вид void = пуст;

b) вид лог = c фактический-описатель, специфицирующий вид 'логическое' c;
вид bool = лог;

c) вид Д цел = c фактический-описатель, специфицирующий вид 'Д цел' c;
вид L int = Д цел c;

d) вид Д вещ = c фактический описатель, специфицирующий вид 'Д вещ' c;
вид L real = Д вещ;

e) вид лит = c фактический-описатель, специфицирующий вид 'литерное' c;
вид char = лит;

f) вид L compl = ст (Д вещ re, im);
вид Д компл = L compl;

g) вид Д бит = ст ([1 : Д размер бит] лог Д F); {см. 10.2.1.j} {Этот указатель-поля скрыт от пользователя для того, чтобы он не мог проникнуть внутрь данной структуры; в частности, он не может индексировать данное поле.}
вид L bits = Д бит;

h) вид Д слог = ст ([1 : Д размер слог] лит Д F); {см. 10.2.1.m}
вид L bytes = Д слог;

i) вид строк = подв [1 : 0] лит;
вид string = строк;

10.2.3. *Стандартные обозначения операций и функций*

10.2.3.0. Стандартные приоритеты.

a) прио минпр = 1, minusab = 1, плюспр = 1, plusab = 1, умпр = 1, timesab = = 1, делпр = 1, divab = 1, цедпр = 1, overab = 1, модпр = 1, modab = 1, прип = 1, plusto = 1,
−:= = 1, +:= = 1, ×:= = 1, *:= = 1, /:= = 1, ÷:= =1,
%:= =1, ÷×:= = 1, ÷*:= = 1, %×:= = 1, %*:= = 1, +=:= 1,
∨ = 2, или = 2, or = 2, ∧ = 3, & = 3, и = 3, and = 3,
= = 4, рв = 4, eq = 4, ≠ = 4, / = = 4, нр = 4, ne = 4,
< = 5, мш = 5, lt = 5, ⩽ = 5, < = = 5, нб = 5, le = 5,
⩾ = 5, > = = 5, нм = 5, ge = 5, > = 5, бм = 5, gt = 5,
− = 6, + = 6,
× = 7, * = 7, / = 7, ÷ = 7, % = 7, цед = 7, over = 7,
÷ × = 7, ÷ * = 7, % × = 7, % * = 7, мод = 7, mod = 7,
□ = 7, элем = 7, elem = 7,
↑ = 8, ** = 8, ↓ = 8,
вверх = 8, up = 8, вниз = 8, down = 8,
лев = 8, shl = 8, прав = 8, shr = 8, нигр = 8, lwb = 8,
вегр = 8, upb = 8, ∟ = 8, ⌜ = 8,
⊥ = 9, +× = 9, +* = 9, им = 9, i = 9;

10.2.3.1. Массивы и связанные с ними операции.

a) вид ⍦ массив = с фактический-описатель, специфицирующий вид, объединенный из {2.1.3.6.a} достаточного набора видов, каждый из которых начинается с вектор с;

b) оп ⋖ нигр, lwb, ∟ ⋗ = (цел n, массив *a*) цел: с нижняя граница в *n*-й граничной паре паспорта значения 'a', если эта граничная пара существует с;

c) оп ⋖ вегр, upb, ∟ ⋗ = (цел n, массив *a*) цел: с верхняя граница в n-й граничной паре паспорта значения 'a', если эта граничная пара существует с;

d) оп ⋖ нигр, lwb, ∟ ⋗ = (массив *a*) цел: 1∟a;

e) оп ⋖ вегр, upb, ⌜ ⋗ = (массив a) цел: 1∟a;

{ Термин „достаточный набор", использованный выше в (a), а также в 10.3.2.2.b и d, подразумевает, что никакая имевшаяся в виду собственно-программа не может не порождаться (никакая не имевшаяся в виду собственно-программа может порождаться) синтаксисом только за счет недостаточности видов в этом наборе. }

10.2.3.2. Операции над логическими операндами.

a) оп ⋖ ∨, или, or ⋗ = (лог a, b) лог: (a | истина | b);

b) оп ⋖ ∧ &, и, and ⋗ = (лог a, b) лог: (a | b | ложь);

c) оп ⋖ ¬, ~, не, not ⋗ = (лог a) лог: (a | ложь | истина);

d) оп ⋖ =, рв, eq ⋗ = (лог a, b) лог: (a ∧ b) ∨ (¬ a ∧ ¬ b);

e) оп ⋖ ≠, /=, нр, ne ⋗ = (лог a), b) лог: ¬ (a = b);

f) оп ⋖ абс, abc ⋗ = (лог a) цел: (a | 1 | 0);

10.2.3.3. Операции над целыми операндами.

a) оп ⋖ <, мш, lt ⋗ = (Д цел a, b) лог: с истина, если значение 'a' меньше { 2.1.3.1.e } значения 'b', а иначе ложь с;

b) оп ≮≤, <=, нб, le ≯ = (Д цел a, b) лог: ¬ (b < a);
c) оп ≮ =, рв, eq ≯ = (Д цел a, b) лог: a ≤ b ∧ b ≤ a ;
d) оп ≮≠, / =, нр, ne ≯ = (Д цел a, b) лог: ¬ (a = b);
e) оп ≮≥, > = нм, ge ≯ = (Д цел a, b) лог: b ≤ a;
f) оп ≮>, бш, gt ≯ = (Д цел a, b) лог: b < a;
g) оп – = (Д цел a, b) Д цел: с значением 'a' минус {2.1.3.1.e} значение 'b' с;
h) оп – = (Д цел a) Д цел: Д 0 – a;
i) оп + = (Д цел a, b) Д цел: a – – b;
j) оп + = (Д цел a) Д цел: a;
k) оп ≮ абс, abs ≯ = (Д цел a) Д цел: (a < Д 0 | – a | a);
l) оп ≮ ×, * ≯ = (Д цел a, b) Д цел:
начало Д цел s:= Д0, i := абс b;
пока i ≥ Д 1
цк s := s + a; i := i – Д 1 кц;
(b < Д 0 | – s | s)
конец;
m) оп ≮ ÷, %, цед, over ≯ = (Д цел a, b) Д цел:
если b ≠ Д 0
то Д цел q := Д 0, r := абс a;
пока (r := r – абс b) ≥ Д 0 цк q := q + Д 1 кц;
(a < Д 0 ∧ b ≥ Д 0 ∨ a ≥ Д 0 ∧ b < Д 0 | – q | q)
все;
n) оп ≮ ÷ ×, ÷ *, % ×, %*, мод, mod ≯ = (Д цел a, b) Д цел:
(Д цел r = a – a ÷ b × b; r < 0 | r + абс b | r);
o) оп / = (Д цел a, b) Д вещ: (Д вещ (a) / Д вещ (b);
p) оп ≮ ↑, * *, вверх, up ≯ = (Д цел a, цел b) Д цел:
(b ≥ 0 | Д цел p := Д 1; до b цк p := p × a кц; p);
q) оп ≮ удл, leng ≯ = (Д цел a) длин Д цел: *с* длинное Д цел значение, удлиненное из {2.1.3.1.e} значения 'a' с;
r) оп ≮ укр, shorten ≯ = (длин Д цел a) Д цел: с Д цел значение, если оно существует, которое можно удлинить до {2.1.3.1.e} значения 'a' с;
s) оп ≮ нчт, odd ≯ = (Д цел a) лог: абс a ÷ × Д 2 = Д 1;
t) оп ≮ знак, sign ≯ = (Д цел a) цел:
(a > Д 0 | 1 | : a < Д 0 | – 1 | 0);
u) оп ≮ ⊥, + ×, + *, i, им ≯ = (Д цел a, b) Д компл: (a, b);

10.2.3.4. Операции над вещественными операндами.

a) оп ≮ <, мш, lt ≯ = (Д вещ a, b) лог: с истина, если значение 'a' меньше {2.1.3.1.e} значения 'b', а иначе ложь с;
b) оп ≮ ≤, <=, нб, le ≯ = (Д вещ a, b) лог: ¬ (b < a);
c) оп ≮ =, рв, eq ≯ = (Д вещ a, b) лог: a ≤ b ∧ b ≤ a;
d) оп ≮ ≠, / =, нр, ne ≯ = (Д вещ a, b) лог: ¬ (a = b);
e) оп ≮ ≥, > =, нм, ge ≯ = (Д вещ a, b) лог: b ≤ a;
f) оп ≮ >, бш, gt ≯ = (Д вещ a, b) лог: b < a;

g) оп – = (Д вещ a, b) Д вещ:
с значение 'a' минус {2.1.3.1.e}
значение 'b' с;
h) оп – = (Д вещ a) Д вещ: Д 0 – a;
i) оп + = (Д вещ a, b) Д вещ: a – – b;
j) оп + = (Д вещ a) Д вещ: a;
k) оп ⊲ абс, abs ⊳ = (Д вещ a) Д вещ: (a < Д 0 | –a | a);
l) оп ⊲ ×, * ⊳ = (Д вещ a, b) Д вещ: с значение 'a', умноженное на {2.1.3.1.e}
значение 'b' с;
m) оп / = (Д вещ a, b) Д вещ: с значение 'a', деленное на {2.1.3.1.e} значение 'b' с;
n) оп ⊲ удл, leng ⊳ = (Д вещ a) длин Д вещ: с длинное Д вещ значение, удлиненное из {2.1.3.1.e} значения 'a' с;
o) оп ⊲ укр, shorten ⊳ = (длин Д вещ a) Д вещ: с если
абс a ⩽ удл Д макс вещ, то Д вещ значение 'v' такое, что для любого Д вещ значения 'w' абс (удл v – a) ⩽ абс (удл w – a) с;
p) оп ⊲ окр, round ⊳ = (Д вещ a) Д цел: с Д цел значение, если оно существует, обобщаемое до {2.1.3.1.e} некоторого Д вещ значения, отличающегося не более чем на одну вторую от значения 'a' с;
q) оп ⊲ знак, sign ⊳ = (Д вещ a) цел:
(a > Д 0 | 1 | : a < Д 0 | – 1 | 0);
r) оп ⊲ целч, entier, ⌊ ⊳ = (Д вещ a) Д цел:
начало Д цел j := Д 0;
пока j < a цк j := j + Д 1 кц;
пока j > a цк j := j – Д 1 кц;
j
конец;
s) оп ⊲ ⊥, + ×, + *, i, им ⊳ = (Д вещ, a, b) Д компл: (a, b);

10.2.3.5. Операции над арифметическими операндами

a) оп P = (Д вещ a, Д цел b) Д вещ: a P Д вещ (b);
b) оп P = (Д цел a, Д вещ b) Д вещ: Д вещ (a) P b;
c) оп R = (Д вещ a, Д цел b) лог: a R Д вещ (b);
d) оп R = (Д цел a, Д вещ b) лог: Д вещ (a) R b;
e) оп ⊲ ⊥, + ×, + *, b ⊳ (Д вещ a, Д цел b) Д компл: (a, b);
f) оп ⊲ ⊥, + ×, + *, b ⊳ = (Д цел a, Д вещ b) Д компл: (a, b);
g) оп ⊲ ↑, * *, вверх, up ⊳ = (Д вещ a, цел b) Д вещ:
(Д вещ p := Д 1; до абс b цк p := p × a кц;
(b ⩾ 0 | p | Д 1 / p));

10.2.3.6. Операции над литерными операндами.

a) оп R = (лит a, b) лог: абс a R абс b; {10.2.1.n}
b) оп + = (лит a, b) строк: (a, b);

10.2.3.7. Операции над комплексными операндами.

a) оп ⊲ вч, re ⊳ = (Д компл a) Д вещ: re из a;
b) оп ⊲ мч, im ⊳ = (Д компл a) Д вещ: im из a;
c) оп ⊲ абс, abs ⊳ = (Д компл a) Д вещ:

```
    Д корень (вч a ↑ 2 + мч a ↑ 2);
d) оп ≮ арг, arg ≯ = (Д компл a) Д вещ:
    если Д вещ вч = вч a, мч = мч a;
    вч ≠ Д 0 ∨ мч ≠ Д 0
    то если абс вч > абс мч
    то Д арктанг (мч / вч) + Д пи / Д 2 ×
    (мч < Д 0 | знак вч – 1 | 1 – знак вч)
    иначе – Д арктанг (вч / мч) + Д пи / Д 2 × знак мч
    все
    все;
e) оп ≮ сопрж, conj ≯ = (Д компл a) Д компл: вч a ⊥ – мч a;
f) оп ≮ =, рв, eq ≯ = (Д компл a, b) лог:
    вч a = вч b ∧ мч a = мч b;
g) оп ≮ ≠,/ =, нр ne ≯ = (Д компл a, b), лог: ¬ (a = b):
h) оп – = (Д компл a, b) Д компл:
    (вч a – вч b) ⊥ (мч a – мч b);
i) оп – = (Д компл a) Д компл: – вч a ⊥ – мч a:
j) оп + = (Д компл a, b) Д компл:
    (вч a + вч b) ⊥ (мч a + мч b);
k) оп + = (Д компл a) Д компл : a;
l) оп ≮ ×, * ≯ = (Д компл a, b) Д компл:
    (вч a × вч b) – мч a × мч b) ⊥ (вч a × мч b + мч a × вч b);
m) оп / = (Д компл a, b) Д компл:
    (Д вещ d = вч (b × сопрж b); Д компл n = a × сопрж b;
    (вч n / d) ⊥ (мч n / d));
n) оп ≮ удл, leng ≯ = (Д компл a) long Д компл:
    удл вч a ⊥ удл мч a;
o) оп ≮ укр, shorten ≯ = (long Д компл a) Д компл:
    укр вч a ⊥ укр мч a;
p) оп P = (Д компл a, Д цел b) Д компл: a P Д компл (b);
q) оп P = (Д компл a, Д вещ b) Д компл: a P Д компл (b);
r) оп P = (Д цел a, Д компл b) Д компл: Д компл (a) Pb;
s) оп P = (Д вещ a, Д компл b) Д компл: Д компл (a) Pb;
t) оп ≮ ↑, * *, вверх, up ≯ = (Д компл a, цел b) Д компл:
    (Д компл p:= Д1; до абс b цк p := p × a кц;
    (b ⩾ 0 | p | Д1 / p));
u) оп E = (Д компл a, Д цел b) лог: a E Д компл (b);
v) оп E = (Д компл a, Д вещ b) лог: a E Д компл (b);
w) оп E = (Д цел a, Д компл b) лог: b E a;
x) оп E = (Д вещ a, Д компл b) лог: b E a;
y) оп ивч = (имя Д компл a) имя Д вещ: re из a;
z) оп имч = (имя Д компл a) имя Д вещ: im из a;
```

10.2.3.8. Битовые и связанные с ними операции.

```
a) оп ≮ =, рв, eg ≯ = (Д бит a, b) лог:
    начало лог c;
    для i до Д размер бит
    пока c := (Д F из a) [i] = (Д F из b) [i]
    цк пропуск кц;
    c
    конец;
b) оп ≮ ≠, / =, нр, ne ≯ = (Д бит a, b) лог: ¬ (a = b);
c) оп ≮ ∨, или, or ≯ = (Д бит a, b) Д бит:
    начало Д бит c:
    для i до Д размер бит
    цк (Д F из c) [i] := (Д F из a) [i] ∨ (Д F из b) [i] кц:
    c
    конец;
d) оп ≮ ∧, &, и, and ≮ = (Д бит a, b) Д бит:
    начало Д бит c;
    для i до Д размер бит
    цк (Д F из c) [i] := (Д F из a) [i] ∧ (Д F из b) [i] кц;
    c
    конец;
e) оп ≮ ⩽, <=, нб, le ≯ = (Д бит a, b) лог: (a ∨ b) = b;
f) оп ≮ ⩾, > =, нм, ge ≯ = (Д бит a, b) лог: b ⩽ a;
g) оп ≮ ↑, вверх, up, лев, shl ≯ = (Д бит a, цел b) Д бит:
    если абс b ⩽ Д размер бит
    то Д бит c := a;
    до абс b
    цк если b > 0 то
    для i от 2 до Д размер бит
    цк (Д F из c) [i – 1] := (Д F из c) [i] кц;
    (Д F из c) [Д размер бит] := ложь
    иначе
    для i от Д размер бит шаг – 1 до 2
    цк (Д F из c) [i] := (Д F из c) [i – 1] кц;
    (Д F из c) [1] := ложь
    все
    кц;
    c
    все;
h) оп ≮ ↓, вниз, down, прав, shr ≯ =
    (Д бит x, цел n) Д бит : x ↑ – n;
i) оп ≮ абс, abs ≯ = (Д бит a) Д цел:
    начало Д цел c := Д 0;
    для i до Д размер бит
    цк c := Д 2 × c + У абс (Д F из a) [i] кц;
```

```
    с
    конец;
j) оп ≮ бин, bin ≯ = (Д цел a) Д бит:
    если a ⩾ Д 0
    то Д цел b := a; Д бит c;
    для i от Д размер бит шаг – 1 до 1
    цк (Д F из c) [i] := нчт b; b:= b ÷ Д 2 кц;
    с
    все;
k) оп ≮ элем, elem, □ ≯ = (цел a, Д бит b) лог:
    (Д F из b) [a];
l) проц Д бит пак = ([ ] лог a) Д бит:
    если цел n =⌜a [c 1];
    n ⩽ Д размер бит
    то Д бит c;
    для i до Д размер бит
    цк (Д F из c) [i] :=
    (i ⩽ Д размер бит – n |ложь| a [c 1] [i – Д размер бит + n])
    кц
    с
    все;
    проц ([ ] лог) Д бит L bits pack = Д бит пак;
m) оп ≮¬, ~, не, not ≯ = (Д бит a) Д бит:
    начало Д бит c;
    для i до Д размер бит цк (Д F из c) [i] :=
                                ¬ (Д F из a) [i] кц;
    с
    конец;
n) оп ≮ удл, leng ≯ = (Д бит a) длин Д бит: длин Д бит пак (a);
o) оп ≮ укр, shorten ≯ = (длин Д бит a) Д бит: Д бит пак ([ ] лог (a)
    [длин Д размер бит – Д размер бит + 1 :]);
    10.2.3.9. Слоговые и связанные с ними операции.
a) оп R = (Д слог a, b) лог: строк (a) R строк (b);
b) оп ≮ элем, elem, □ ≯ = (цел a, Д слог b) лит: (Д F из b) [a];
c) проц Д слог пак = (строк a) Д слог:
    если цел n =⌜a [c 1]:
    n ⩽ Д размер слог
    то Д слог c;
    для i до Д размер слог
    цк (Д F из c) [i] := (i ⩽ n |a [c 1] [i]|заполнитель)
    кц;
    с
    все;
    проц (строк) Д слог L bytes pack = Д слог пак;
d) оп ≮ удл, leng ≯ = (Д слог a) длин Д слог: длин Д слог пак (a);
```

```
e) оп ≮ укр, shorten ≯ = (длин Д слог a) Д слог;
     Д слог пак (строк (a) [: Д размер слог]);
```

10.2.3.10. Строковые и связанные с ними операции.

```
a) оп ≮ <; мш, lt ≯ = (строк a, b) лог:
     начало цел m = ⌈ a [c 1], n = ⌈ · b [c 1]; цел c := 0;
     для i до (m < n | m | n)
     пока (c := абс a [c 1] [i] – абс b [c 1] [i]) = 0
     цк пропуск кц;
     (c = 0 | m < n ∧ n > 0 | c < 0)
     конец;
b) оп ≮ ⩽, <=, нб, le ≯ = (строк a, b) лог: ¬ (b < a);
c) оп ≮ =, рв, eq ≯ = (строк a, b) лог: a ⩽ b ∧ b ⩽ a;
d) оп ≮ ≠, / =, нр, ne ≯ = (строк a, b) лог: ¬ (a = b);
e) оп ≮ ⩾, > =, нм, ge ≯ = (строк a, b) лог: b ⩽ a;
f) оп ≮ >, бш, gt ≯ = (строк a, b) лог: b < a;
g) оп R = (строк a, лит b) лог: a R строк (b);
h) оп R = (лит a, строк b) лог: строк (a) R b;
i) оп + = (строк a, b) строк:
     (цел m = (int | a = ⌈ a [c 1]; |a < 0 | 0 | |a),
      n = (цел |b = | b [c 1]; |b < 0 | 0 | |b);
      [1 : m + n] лит c;
      (m > 0 | c [1 : m] := a [ c 1]);
      (n > 0 | c [m + 1 : m + n] := b [c 1]; c);
j) оп + = (строк a, лит b) строк : a + строк (b);
k) оп + = (лит a, строк b) строк : строк (a) + b;
l) оп ≮ ×, * ≯ = (строк a, цел b) строк:
     (строк c; до b цк c := c + a кц; c);
m) оп ≮ ×, * ≯ = (цел a, строк b) строк : b × a;
n) оп ≮ ×, * ≯ = (лит a, цел b) строк : строк (a) × b;
o) оп ≮ ×, * ≯ = (цел a, лит b) строк: b × a;
```

{ Из операций, определенных в a, g и h, следует, что если абс "a" < абс "b", то "<" "a"; "a" < "b"; "aa" < "ab"; "aa" < "ba"; "ab" < "b" и "ab" < < "ba". }

10.2.3.11. Операции, соединенные с присваиваниями.

```
a) оп ≮ минпр, minusab, –:= ≯ = (имя Д цел a, Д цел b)
     имя Д цел: a := a – b;
b) оп ≮ минпр, minusab, –:= ≯ = (имя Д вещ a, Д вещ b)
     имя Д вещ: a := a – b;
c) оп ≮ минпр, minusab, –:= ≯ = (имя Д компл a, Д компл b)
     имя Д компл: a := a – b;
d) оп ≮ плюспр, plusab, +:= ≯ = (имя Д цел a, Д цел b)
     имя Д цел: a := a + b;
e) оп ≮ плюспр, plusab, +:= ≯ = (имя Д вещ a, Д вещ b)
     имя Д вещ: a := a + b;
f) оп ≮ плюспр, plusab, +:= ≯ = (имя Д компл a, Д компл b)
```

имя Д компл: a := a + b;
g) оп ≺ умпр, timesab, × :=, * := ≻ = (имя Д цел a, Д цел b)
имя Д цел : a :=a × b;
h) оп ≺ умпр, timesab, × :=, * := ≻ = (имя Д вещ a, Д вещ b)
имя Д вещ; a := a × b;
i) оп ≺ умпр, timesab, × :=, * := ≻ = (имя Д компл a, Д компл b)
имя Д компл : a := a × b;
j) оп ≺ цедпр, overab, ÷:=, %:= ≻ = (имя Д цел a, Д цел b)
имя Д цел :a:= a ÷ b;
k) оп ≺ модпр, modab, ÷ × :=, ÷*:=, %×:=, %* := ≻
= (имя Д цел a, Д цел b) имя Д цел : a:= a ÷× b;
l) оп ≺ делпр, divab, /:= ≻ = (имя Д вещ a, Д вещ b)
имя Д вещ :a:= a / b;
m) оп ≺ делпр, /:= ≻ = (имя Д компл a, Д компл b)
имя Д компл: a:= a / b;
n) оп Q = (имя Д вещ a, Д цел b) имя Д вещ: a Q Д вещ (b);
o) оп Q = (имя Д компл a, Д цел b) имя Д компл: a Q Д компл (b);
p) оп Q = (имя Д компл a, Д вещ b) имя Д компл: a Q Д компл (b);
q) оп ≺ плюспр, plusab, +. = ≻ =
(имя строк a, строк b) имя строк: a:= a + b;
r) оп ≺ прип, plusto, + =: ≻ =
(строк a, имя строк b) имя строк: b:= a + b;
s) оп ≺ плюспр, plusab, +:= ≻ =
(имя строк a, лит b) имя строк: a +:= строк (b);
t) оп ≺ прип, plusto, +=: ≻ =
(лит a, имя строк b) имя строк: строк (a) +=: b;
u) оп ≺ умпр, timesab, ×:=, * := ≻ =
(имя строк a, цел b) имя строк: a:= a × b;
10.2.3.12. Стандартные математические константы и функции.
a) Д вещ Д пи = с Д вещ значение близкое к числу П;
Д вещ L pi = Д пи;
b) проц Д корень = (Д вещ x) Д вещ: с если x ≥ Д 0,
Д вещ значение, близкое к квадратному корню из 'x' с;
проц (Д вещ) Д вещ L sqrt = Д корень;
c) проц Д эксп = (Д вещ x) Д вещ: с Д вещ значение, если оно существует,
близкое к экспоненте от 'x' с;
проц (Д вещ) Д вещ L exp = Д эксп;
d) проц Д лг = (Д вещ x) Д вещ: с Д вещ значение, если оно существует,
близкое к натуральному логарифму от 'x' с;
проц (Д вещ) Д вещ L ln = Д лг;
e) проц Д кос = (Д вещ x) Д вещ: с Д вещ значение, близкое к косинусу
от 'x' с;
проц (Д вещ) Д вещ L cos = Д кос;
f) проц Д арккос (Д вещ x) Д вещ: с если абс x ≤ Д 1,

Д вещ значение, близкое к арккосинусу от 'x',
Д 0 ⩽ Д арккос (x) ⩽ Д пи с;
проц (Д вещ) Д вещ L arccos = Д арккос;

g) проц Д син = (Д вещ x) Д вещ: с Д вещ значение, близкое к синусу от 'x' с;
проц (Д вещ) Д вещ L sin = Д син;

h) проц Д арксин = (Д вещ x) Д вещ: с если абс x ⩽ Д 1,
Д вещ значение, близкое к арксинусу от 'x',
абс Д арксин (x) ⩽ Д пи / Д 2 с;
проц (Д вещ) Д вещ L arcsin = Д арксин;

i) проц Д танг = (Д вещ x) Д вещ: с Д вещ значение, близкое к тангенсу от 'x' с;
проц (Д вещ) Д вещ L tan = Д танг;

j) проц Д арктанг = (Д вещ x) Д вещ: с Д вещ значение, близкое к арктангенсу от 'x',
абс Д арктанг (x) ⩽ Д пи / Д 2 с;
проц (Д вещ) Д вещ L arctan = Д арктанг;

k) проц Д след псч = (имя Д цел a) Д вещ:
(a:= с следующее после 'a' псевдослучайное Д цел значение из некоторой однородно распределенной последовательности на отрезке [Д 0, Д макс цел] с;
с вещественное значение, соответствующее 'a' по некоторому отображению целых значений [Д 0, Д макс цел] в вещественные [Д 0, Д 1) {т.е. так, что 0 ⩽ x ⩽ 1}, такое, что порожденная при этом последовательность вещественных значений сохраняет свойства псевдослучайности и однородности распределения данной последовательности целых чисел с);
проц (имя Д цел) Д вещ L next random = Д след псч;

10.2.4. *Операции синхронизации*

Исполнение параллельного-предложения P {3.3.1.c} в некотором окружении E называется „параллельным действием". Исполнение составляющей основы этого P в E называется „процессом" этого параллельного действия.

Всякое исполнение A {в некотором окружении} одного из ЗАКРЫТЫХ-предложений, выделенных прагматами {9.2.1.b} прагм начало несовместимой части прагм и прагм конец несовместимой части прагм в формах 10.2.4.d и 10.2.4.e, несовместимо {2.1.4.2.e} ни с каким исполнением B другого из этих ЗАКРЫТЫХ-предложений, если A и B – наследные действия {2.1.4.2.b} разных процессов одного и того же параллельного действия.

a) вид сема = ст (имя цел F); вид sema = сема;

b) оп ⩽ уст, level ⩾ = (цел a) сема:
(сема s; .F из s := глоб цел := a; s);

c) оп ⩽ уст, level ⩾ = (сема a) цел : F из a;

d) оп ⩽ вниз, down ⩾ = (сема эдсгер) пуст;
начало имя цел дейкстра = F из эдсгер;

```
    пока
     прагм начало несовместимой части прагм
      если дейкстра ⩾ 1 то дейкстра –:= 1; ложь
      иначе
        с пусть Р будет таким процессом, что исполнение данного псевдо-
        примечания {10.1.3. Шаг 7} есть наследное действие этого Р, но
        никакого другого процесса, наследного для Р; данный процесс Р
        приостанавливается
        {2.1.4.3.f}
        с;
       истина
      все
     прагм конец несовместимой части прагм
    цк пропуск кц
   конец,
е) ол ≮ up, вверх ≯ = (сема эдсгер) пуст:
   прагм начало несовместимой части прагм
     если имя цел дейкстра = F из эдсгер;
          (дейкстра +:= 1)    ⩾ 1
     то
        с возобновляются {2.1.4.3.g} все процессы, приостановленные
        потому, что целое число, именуемое выдаваемым 'дейкстра' име-
        нем, было меньше единицы с
     все
    прагм конец несовместимой части прагм;
```

10.3. Описание обмена

{Стандартным-вступлением предусмотрено три способа „обмена" (т.е. ввода и вывода), а именно бесформатный обмен (10.3.3), форматный обмен (10.3.5) и двоичный обмен (10.3.6).}

10.3.1. *Книги, каналы и файлы*

{„Книги, „каналы" и „файлы" моделируют устройства обмена физической машины, используемой в реализации.}

10.3.1.1. Книги и связки.

{аа) Вся информация в системе содержится в ряде „книг". Книга (а) является структурой, содержащей поле текст вида, специфицированного описателем подвтекст (b), именующее информацию в форме литер. Этот текст имеет переменное число страниц, каждая из которых может содержать переменное число строчек с переменным числом литер в каждой строчке. Номер страницы, номер строчки и номер литеры определяют позицию в тексте. Книга включает также поле заполн, указывающее „логический конец" этой книги, т.е. позицию, до которой книга заполнена информацией, строку обозначение, идентифицирующую данную книгу и, возможно, содержащую некоторую другую (служебную) информацию, например о ее владельце, а также поля запись и пользователи, позволяющие открывать

(10.3.1.4.d) данную книгу одновременно для более чем одного файла только в том случае, если запись в нее невозможна ни для одного из них.

bb) Доступ к книгам в системе осуществляется через цепочку связок. Эта цепочка доступных для открытия (10.3.1.4.dd) книг именуется доступные книги. Данная книга может именоваться более чем одной связкой в этой цепочке, таким образом создается возможность одновременного доступа к отдельной книге из более чем одного процесса (10.2.4). Однако каждый такой доступ возможен только для чтения книги, так как только один процесс может иметь доступ к книге, чтобы писать в нее (aa). Цепочка книг, которые были сняты (10.3.1.4.o), именуется снятые книги.

cc) Одновременный доступ более чем одного процесса к цепочке связок предотвращается использованием семафора защита связей, обеспечивающем взаимное исключение между такими процессами.

dd) Книги могут создаваться (например, при вводе) или уничтожаться (например, после вывода) посредством задач (например, операционной системой) из списка-системных-задач (10.4.2); эти книги тогда добавляются к цепочке связок или удаляются из нее }.

```
a) вид ♀ книга =
       ст (подвтекст текст,
       позиция заполн ≠ логический конец книги ≠,
       строк обозначение ≠ для идентификации ≠,
      лог запись ≠ истина, если эта книга открыта для записи ≠
      цел пользователи ≠ сколько раз эта книга была открыта ≠ ;
 b) вид ♀ текст = имя [ ] [ ] [ ]      лит,
    вид ♀ подвтекст = имя подв [ ] подв [ ] подв [ ] лит;
c) вид ♀ позиция = ст (цел p, l, c);
d) прио ♀ вне = 5,
   оп вне = (позиция a, b) лог:
     если p из a < p из b то ложь
     инес p из a > p из b то истина
     инес l из a < l из b то ложь
     инес l из a > l из b то истина
     иначе c из a > c из b
     все;
e) вид ♀ связка = ст (имя книга книга,
                      имя связка след ≠ ующая ≠);
f) имя связка ♀ доступные книги := нил;
g) имя связка ♀ снятые книги := нил;
h) сема ♀ защита связей = (сема s; F из s := ПЕРВ цел := 1; s);
```

10.3.1.2. Каналы.

{aa) „Канал" соответствует одному или нескольким реальным устройствам (например, перфоратору или устройству построчной печати, или даже некоторой установке для ядерной физики, результаты работы которой со-

бираются вычислительной машиной) или же системному архиву данных, поддерживаемому операционной системой. Канал есть структура, поля которой являются процедурами, вырабатывающими истинностные значения, определяющие допустимые методы доступа к книгам, связанным (с файлами) через данный канал. Поскольку эти методы доступа к книге могут зависеть как от книги, так и от канала (например, некоторая книга может быть устроена так, что ее можно читать, но в нее нельзя записывать), большинство свойств этих методов доступа зависят и от канала, и от книги. Соответствующие свойства можно проверить с помощью запросов к обстановке, предусмотренных для файлов (10.3.1.3.ff). Для каналов предусмотрены два запроса к обстановке. Это:

- можно завести, вырабатывающий истина, если на данном канале можно „завести" (10.3.1.4.cc) другой файл;
- станд код, применяемый для получения „кодирующей таблицы" (bb), которая используется по умолчанию для данного канала.

bb) „Кодирующая таблица" есть значение вида, специфицируемого описателем код, используемое для перекодирования литер из значений, хранимых во „внутренней" форме в памяти машины, в значения, хранимые во „внешней" форме в некоторой книге, и обратно. Эта таблица является структурой, состоящей из вектора структур, каждая из которых содержит значение во внутренней форме и соответствующее ему внешнее значение. Дополнительные кодирующие таблицы могут предусматриваться реализациями в их библиотечных-вступлениях.

cc) Предусмотрены три стандартных канала, свойства которых определены ниже (e, f, g). Дополнительные каналы могут предусматриваться реализацией в ее библиотечном-вступлении. Поле номер канала предусмотрено для того, чтобы различать каналы, свойства которых в остальном совпадают. }

```
    a) вид канал = ст (проц (имя книга) лог ↑ уст нач,
        ↑ установка, ↑ ввод, ↑ вывод,
        ↑ двоичный, ↑ сжатие, ↑ переобозначение,
        проц лог ↑ заведение, проц позиция ↑ макс позиция,
        проц (имя книга) код ↑ станд код, цел ↑ номер канала);
    вид channel = канал;
b) вид ↑ код = ст ([1 : цел (пропуск)] ст (лит внутр, внешн) F);
c) проц можно завести = (канал кан) лог: заведение из кан;
    проц (канал) лог estab possible = можно завести;
d) проц станд код = (канал кан) проц (имя книга) код:
    станд код из кан;
    проц (канал) проц (имя книга) код stand conv = станд код;
e) канал станд канал ввода = с значение вида канал, поле которого, выбираемое по 'ввод', есть процедура, всегда вырабатывающая истина, а
    другие поля имеют какие-то подходящие значения с;
    канал stand in channel = станд канал ввода;
```

f) канал станд канал вывода = с значение вида канал, поле которого, выбираемое по 'вывод',есть процедура, всегда вырабатывающая истина, а другие поля имеют какие-то подходящие значения с,
канал stand out channel = станд канал вывода;

g) канал станд канал обмена = с значение вида канал, поля которого, выбираемые по 'установка', 'уст нач', 'ввод', 'вывод' и 'двоичный', являются процедурами, всегда вырабатывающими истина, а другие поля имеют какие-то подходящие значения с;
канал stand back channel = станд канал обмена;

10.3.1.3. Файлы.

aa) „Файлы" служат средством сообщения между собственно-программой и книгой, с которой этот файл был открыт на некотором канале. Файл есть структура, содержащая имя книги, с которой он был связан (10.3.1.4.bb) и отдельно имя текста этой книги. Файл содержат также информацию, необходимую для работы процедур обмена с этой книгой, включающую его текущую позицию тпоз в данном тексте, его текущее „состояние" (bb), его текущий „формат" (10.3.4) и канал, на котором он был открыт.

bb) „Состояние" файла определяется пятью полями:

- для чтения, которое есть истина, если данный файл предназначен для ввода;
- для записи, которое есть истина, если данный файл предназначен для вывода;
- для литер, которое есть истина, если данный файл предназначен для литерного обмена;
- для двоичн, которое есть истина, если данный файл предназначен для двоичного обмена;
- открыт, которое есть истина, если данный файл связан с какой-нибудь книгой.

cc) Файл включает некоторые „процедуры обработки события", вызываемые, когда во время обмена возникают определенные условия. По умолчанию предусматривается, что после открытия файла эти процедуры обработки события вырабатывают ложь, когда они вызываются, но программист может предусмотреть и другие процедуры обработки события. Поскольку соответствующие поля файла не доступны прямо для пользователя, процедуры обработки события можно изменять с помощью „процедур реакции" (l, m, n, o, p, q, r). Процедуры обработки события всегда задают в качестве параметра имя своего файла. Если исполнение процедуры обработки события прекращается, то вызвавшая ее процедура обмена не может действовать дальше; в противном случае, если она вырабатывает истина, предполагается, что данное условие было некоторым образом исправлено, и, если возможно, обмен продолжается, но, если она вырабатывает ложь, система продолжает работу, предпринимая действия по умолчанию. Процедуры реакции таковы:

- при конце лог файла. Соответствующая процедура обработки события

вызывается, когда в ходе ввода или в результате вызова установить достигается логический конец соответствующей книги (см. 10.3.1.6.dd).

Пример:

Программист хочет знать количество целых чисел на входной ленте. Файл лента ввода был открыт во внешнем блоке. Если он напишет

```
начало цел n := 0;
  при конце лог файла (лента ввода,
    (имя файл файл) лог: на f);
  цк ввод (лента ввода, лок цел); n +:= 1 кц;
  f : печ (n)
конец,
```

то такое присваивание соответствующему полю из файла лента ввода нарушит ограничения на области действия, поскольку область действия процедуры (имя файл файл) лог: на f меньше области действия файла лента ввода. Поэтому ему следует написать

```
начало цел n := 0; файл вспомог := лента ввода;
  при конце лог файла (вспомог,
    (имя файл файл) лог: на f);
  цк ввод (вспомог, лог цел); n +:= 1 кц;
  f: печ (n)
конец.
```

- при конце физ файла. Соответствующая процедура обработки события вызывается, когда текущий номер страницы данного файла превышает число страниц в соответствующей книге, а обмен пытаются продолжить (см. 10.3.1.6.dd).
- при конце страницы. Соответствующая процедура обработки события вызывается, когда текущий номер строчки данного файла превышает число строчек на текущей странице, а обмен пытаются продолжить (см. 10.3.1.6.dd).
- при конце строчки. Соответствующая процедура обработки события вызывается, когда текущий номер литеры данного файла превышает число литер в текущей строчке, а обмен пытаются продолжить (см. 10.3.1.6.dd).

Пример:

Программист хочет, чтобы на начале каждой страницы его файла f автоматически печатался заголовок:

```
при конце страницы (f, (имя файл файл) лог:
   (вывод (файл, (нов страница, "стр.", целое (i +:= 1,0).
      нов строчка)); истина)
      |# предполагается, что i было где-то описано #|).
```

- при ошибке литеры. Соответствующая процедура обработки события вызывается, когда перекодирование некоторой литеры не было успешным или когда в ходе ввода читается не „ожидаемая" (10.3.4.1.ll) литера. Эта процедура обработки события вызывается с именем литеры, предлагаемой в качестве замены. Процедура обработки события, задаваемая программи-

стом, может присваивать литеру, отличную от предлагаемой. Если данная процедура обработки события вырабатывает истина, то используется эта предлагаемая литера с возможным ее изменением.

Пример:

Программист хочет читать суммы денег, отперфорированные в форме „$123.45", „ –$23.45", „ – –$ 3.45" и т. д.:

при ошибке литеры /станд ввод,
(имя файл f, имя лит пред) лог:
если пред = "0"
то лит с; назад (f); ввод (f, с);
(с="$" | ввод (f, пред); истина | ложь)
иначе ложь
все);
цел центы; ф чит ((ф 3z". "dd ф, центы));

• при ошибке значения. Соответствующая процедура обработки события вызывается, когда:

(i) в ходе форматного обмена делается попытка обмена какого-нибудь значения под контролем некоторого „шаблона", с которым оно несовместимо, или когда число „рамок" недостаточно. Если эта процедура вырабатывает истина, то текущее значение и шаблон пропускаются, и обмен продолжается; если же процедура вырабатывает ложь, то вызывается не определено, перед которым при выводе выводится значение при помощи процедуры вывод;

(ii) в ходе ввода оказывается невозможным преобразовать строку в значение некоторого данного вида (это может случиться, например, при попытке прочитать целое число, большее, чем макс цел (10.2.1.с)).

• при конце формата. Соответствующая процедура обработки события вызывается, когда в ходе форматного обмена формат исчерпывается, в то время как еще остается некоторое значение, подлежащее обмену. Если данная процедура вырабатывает истина, то в случае когда она не обеспечила нового формата для этого файла, вызывается не определено, а иначе повторяется текущий формат.

dd) Поле код файла – это его текущая кодирующая таблица (10.3.1.2.bb). После открытия файла кодирующая таблица обеспечивается по умолчанию. Другие кодирующие таблицы могут применяться программистом при помощи вызова процедуры задать код (j). Такие таблицы должны предусматриваться в библиотечном-вступлении.

ее) Процедура задать стопстроку вызывается для присоединения к файлу некоторой строки. Эту строку используют во время ввода переменного числа литер, любая из ее литер служит для окончания ввода.

ff) Имеющиеся методы доступа к книге, открытой с каким-нибудь файлом, можно узнать посредством вызовов следующих процедур (выдача такого вызова может быть функцией и этой книги, и данного канала, и некоторых других факторов обстановки, не определяемых настоящим стандартом):

• возм ввод, которая вырабатывает истина, если данный файл можно использовать для ввода;
• возм вывод, которая вырабатывает истина, если данный файл можно использовать для вывода;
• возм двоичн, которая вырабатывает истина, если данный файл можно использовать для двоичного обмена;
• сжимаем, которая вырабатывает истина, если строчки и страницы будут сжимаемы (10.3.1.6.aa) в ходе вывода; в этом случае соответствующая книга тоже называется „сжимаемой";
• возм уст нач, которая вырабатывает истина, если для данного файла возможна установка на начало, т.е. его текущую позицию можно установить в (1,1,1);
• возм установка, которая вырабатывает истина, если данный файл можно установить, т.е. его текущей позиции можно придать заданное значение; соответствующая книга называется тогда книгой с „произвольным доступом", а иначе – книгой с „последовательным доступом";
• возм переобозначение, которая вырабатывает истина, если поле обозначение данной книги можно изменить;
• канал, которая вырабатывает канал, где был открыт данный файл (это можно использовать, например, при помощи процедуры, присваиваемой посредством при конце физ файла, для того чтобы открыть другой файл на том же канале).

gg) Если при книге с последовательным доступом двоичный и литерный обмен чередуются, то вызывается не определено (10.3.1.4.a), т.е. после открытия или установки на начало (10.3.1.6.j) возможен любой из этих обменов, но, если один из них имел место, другой может происходить только после очередной установки на начало.

hh) При книге с последовательным доступом вывод немедленно заставляет логический конец книги переместиться на текущую позицию (если они оба не находятся в одной и той же строчке); таким образом, за вводом не может следовать вывод без предшествующей установки на начало (10.3.1.6.j).

Пример:

```
начало файл f1, f2; [1 : 10 000] цел x; цел n := 0;
  открыть (f1, " ", канал 2);
  f2 := f1;
    # теперь f1 и f2 можно использовать один вместо
      другого #
задать код (f1, код связи);
задать код (f2, код телекса);
  # теперь f1 и f2 используют разные коды; код связи и
    код телекса определены библиотечным-вступлением
    данной реализации #
уст нач (f1); # в результате на f2 тоже произойдет
                установка на начало #
```

```
    при конце лог файла (f1, (имя файл f) лог: на сделано);
    для i цк ввод (f1,x [i]); n := i кц;
    # плохо, если при вводе окажется больше 10 000 целых
      чисел #
    сделано:
      уст нач (f1); для i до n цк вывод (f2, x [i]) кц;
          закрыть (f2) # f1 тоже закроется #
    конец }
a) вид файл = ст (имя книга ↑книга, об (подвтекст, текст)
     ↑текст, канал ↑кан, имя формат ↑формат,
     имя цел ук #азатель # фор # мата #,
     имя лог ↑для чтения, ↑для записи, ↑для литер, ↑для
       двоичн, ↑открыт,
     имя позиция ↑ т # екущая # поз # иция #,
     строк ↑ стопс # трока #,
     код ↑код # ирующая таблица для литер #,
     проц (имя файл) лог ↑испр # авлен # лог файл, ↑испр
       физ файл, ↑испр # авлена # страница, ↑испр строчка,
       ↑испр формат, ↑испр # авлена # ошибка значения,
     проц (имя файл, имя лит) лог ↑испр # авлена # ошибка
       литеры);
   вид file = файл;
b) проц возм # ожен # ввод = (имя файл f) лог:
     (открыт из f | (ввод из кан из f) (книга из f) |
     не определено; пропуск);
   проц (имя файл) лог get possible = возм ввод;
c) проц возм вывод = (имя файл f) лог:
     (открыть из f | (вывод из кан из f) (книга из f) |
     не определено; пропуск);
   проц (имя файл) лог put possible = возм вывод;
d) проц возм двоичн # ый обмен # = (имя файл f) лог:
     (открыт из f | (двоичный из кан из f) (книга из f) |
     не определено; пропуск);
   проц (имя файл) лог bin possible = возм двоичн;
e) проц сжимаем = (имя файл f) лог:
     (открыт из f | (сжатие из кан из f) (книга из f) |
     не определено; пропуск);
   проц (имя файл) лог compressible = сжимаем;
f) проц возм уст нач = (имя файл f) лог:
     (открыт из f | (уст нач из кан из f) (книга из f) |
     не определено; пропуск);
   проц (имя файл) лог reset possible = возм уст нач;
g) проц возм установка = (имя файл f) лог:
     (открыт из f | (установка из кан из f) (книга из f) |
     не определено; пропуск);
```

```
    проц (имя файл) лог set possible = возм установка:
h) проц возм переобозначение = (имя файл f) лог:
      (открыт из f | (переобозначение из кан из f)
                     (книга из f) | не определено; пропуск);
    проц (имя файл) лог reidf possible = возм переобозначение;
i) проц канал = (имя файл f) канал:
      (открыт из f | кан из f | не определено; пропуск);
    проц (имя файл) канал chan = канал;
j) проц задать код = (имя файл f, проц (имя книга) код c)
      пуст: (открыт из f | код из f := c (книга из f) |
            не определено);
    проц (имя файл, проц (имя книга) код) пуст
        make conv = задать код;
k) проц задать стопстроку = (имя файл f, строк t) пуст:
      стопс из f := t;
    проц (имя файл, строк) пуст
       make term = задать стопстроку;
l ) проц при конце лог файла = (имя файл f,
      проц (имя файл) лог p) пуст: испр лог файл из f := p;
    проц (имя файл, проц (имя файл) лог) пуст
      on logical file end = при конце лог файла;
m) проц при конце физ файла = (имя файл f,
      проц (имя файл) лог p) пуст: испр физ файл из f := p;
    проц (имя файл, проц (имя файл) лог) пуст
     on physical file end = при конце физ файла;
n) проц при конце страницы = (имя файл f,
      проц (имя файл) лог p) пуст: испр страница из f := p;
    проц (имя файл, проц (имя файл) лог) пуст
      on page end = при конце страницы;
o) проц при конце строчки = (имя файл f,
      проц (имя файл) лог p) пуст: испр строчка из f := p;
    проц (имя файл, проц (имя файл) лог) пуст
      on line end = при конце строчки;
p) проц при конце формата = (имя файл f,
      проц (имя файл) лог p) пуст: испр формат из f := p;
    проц (имя файл, проц (имя файл) лог) пуст
      on format end = при конце формата;
q) проц при ошибке значения = (имя файл f,
      проц (имя файл) лог p) пуст:
                              испр ошибка значения из f := p;
    проц (имя файл, проц (имя файл лог) пуст
      on value error = при ошибке значения;
r) проц при ошибке литеры = (имя файл f,
      проц (имя файл, имя лит) лог p) пуст: испр ошибка
      литеры из f := p;
```

```
проц (имя файл, проц (имя файл, имя лит) лог) пуст
  on char error = при ошибке литеры;
s) проц переобозначение = (имя файл f, строк обозн) пуст:
    если открыт из f ∧ возм переобозначение (f) ∧
     приемлемо обозн (обозн)
    то обозначение из книга из f := обозн
    все;
проц (имя файл, строк) пуст reidf = переобозначение;
```

10.3.1.4. Открытие и закрытие файлов.

{aa) Когда в ходе обмена происходит что-то, оставленное неопределенным, например, явно вызывается не определено (a), это вовсе не значит, что исполнение катастрофически и немедленно прерывается (2.1.4.3.h), а лишь что предпринимаются какие-то осмысленные действия, которые не описываются или их нельзя описать одним данным стандартом, то есть они зависят от реализации.

bb) Книга „связывается" с файлом при помощи процедур завести (b), создать (c) и открыть (d). Эта связь может прекратиться с помощью процедур закрыть (n), снять (o) и стереть (p).

cc) Когда файл „заводится" на каком-то канале, создается (5.2.3) книга с текстом заданного размера, с заданной идентифицирующей строкой, с полем запись, установленным в истина и с логическим концом книги в (1,1,1). Некоторые реализации могут требовать (g), чтобы образующие идентифицирующую строку литеры были взяты из ограниченного множества и чтобы эта строка была ограниченной длины. Может также потребоваться, чтобы никакие две книги не имели одинаковые идентифицирующие строки. Если заведение завершается успешно, то вырабатывается значение 0, а иначе вырабатывается ненулевое целое число (его значение может указывать, почему файл не был успешно заведен).

Когда файл „создается" на канале, заводится какой-то файл с книгой, размер текста которой принят по умолчанию для этого канала, а идентифицирующая строка не определена.

dd) Когда файл „открывается", в цепочке связок ищется первая книга, такая, что соответствует (h) вырабатывает истина. (Точной метод соответствия не определяется настоящим стандартом и, вообще говоря, зависит от реализации. Например, поставляемая в качестве параметра процедуры открыть строка может в некоторой форме включать в себя какой-нибудь пароль). Если достигается конец данной цепочки связок или книга уже была выбрана и ее поле запись вырабатывает истина, или запись в эту книгу возможна через данный канал, а книга уже открыта, то дальнейшее исполнение не определено. Если данный файл уже открыт, то операция вверх гномы обеспечивает возможность для подходящих системных действий над этой предварительно связанной книгой (в случае, если не осталось никакой другой копии данного файла, чтобы сохранить эту связь).

ee) Процедуру соединить можно использовать, чтобы соединить с файлом значение вида, специфицируемого одним из описателей имя [ ] лит,

имя [ ] [ ] лит или имя [ ] [ ] [ ] лит, таким образом, подобные переменные можно использовать в качестве книги некоторого файла.

ff) Когда файл „закрывается", его книга присоединяется к цепочке связок, именуемой доступные книги. Затем активизируется некоторая системная-задача при помощи вверх гномы. (Она может реорганизовать эту цепочку связок, удалить данную книгу или добавить ее другие копии. Она может также вызвать выдачу этой книги на внешнее устройство).

gg) Когда файл „снимается", его книга присоединяется к цепочке связок, именуемой снятые книги. Затем активизируется некоторая системная-задача при помощи вверх гномы. Снятую книгу нельзя снова открыть до тех пор, пока какая-нибудь последующая системная-задача не переместит эту книгу в цепочку связок, доступных для открытия.

hh) Когда файл „стирается", активизируется некоторая системная-задача при помощи вверх гномы. (Она может некоторым образом удалить книгу, связанную с этим файлом.)}

а) проц ?не определено = цел: с какое-либо осмысленное
действие системы, выдающее целое число, чтобы указать, что было сделано; предполагается, что это действие системы может зависеть от знания любого значения, к которому есть доступ {2.1.2.с} через участок любого окружения, старшего, чем то окружение, в котором исполняется данное псевдопримечание {, несмотря на то, что нельзя написать никакого конструкта Алгола 68, который может иметь доступ к этим значениям} с;

b) проц завести =
(имя файл файл, строк обозн, канал кан, цел *p, l, c*) цел:
начало
вниз защита связей;
ПЕРВ книга книга :=
(ПЕРВ подв [1 : p] подв [1 : l] подв [1 : c] лит,
(1,1,1), обозн, истина, 1):
если файл доступен (кан) ∧
(вывод из кан) (книга) ∧
заведение из кан ∧ ¬
(позиция (p, l, c) вне макс позиция из кан)
∧ ¬ (позиция (1,1,1) вне позиция (p, l, c)) ∧
обозн приемлемо (обозн)
то
(открыт из файл |
вверх гномы | вверх защита связей);
файл :=
(книга, текст из книга, кан, пропуск, пропуск,
состояние: # глоб лог := ложь, глоб лог := истина,
глоб лог := ложь, глоб лог := ложь, глоб лог := истина,
глоб позиция := (1,1,1), " ",
(станд код из кан) (книга),

```
        # процедуры обработки события: # ложь, ложь, ложь,
        ложь, ложь, ложь,
        (имя файл f, имя лит a) лог: ложь);
        (¬ возм двоичн (файл) |
          настроить на литерное (файл));
        0
        иначе вверх защита связей; не определено
        все
      конец;
      проц (имя файл, строк,
            канал, цел, цел, цел) цел establish = завести;
c)  проц создать = (имя файл, файл, канал кан) цел:
    начало позиция макс позиция = макс позиция из кан:
      завести (файл, пропуск, кан, р из макс позиция,
      l из макс позиция, с из макс позиция)
    конец;
    проц (имя файл, канал) цел create = создать;
d)  проц открыть = (имя файл файл,
                    строк обозн, канал кан) цел:
    начало
      вниз защита связей;
      если файл доступен (кан)
      то имя имени связка св := доступные книги;
      лог есть := ложь;
       пока (имя связка (св) :≠: нил) ∧ ¬ есть
       цк
         если соответствует (обозн, кан, книга из св)
         то есть := истина
         иначе св := след из св
         все
       кц;
       если ¬ есть
       то вверх защита связей; не определено
      иначе имя книга книга := книга из св;
        если запись из книга ∨ (вывод из кан) (книга) ∧
          пользователи из книга > 0
       то
         вверх защита связей; не определено
        # в этом случае открытию мешают другие
         пользователи — система может либо ждать, либо
         выдать ненулевое значение (указывающее
         на неуспех открытия) немедленно #
      иначе
       пользователи из книга +:= 1;
       (( вывод из кан) (книга) |
```

```
        запись из книга := истина);
      имя имени связка (св) := след из св;
      # удалить связку из цепочки доступных #
      (открыт из файл |
        вверх гномы | вверх защита связей);
      файл :=
        (книга, текст из книга, кан, пропуск, пропуск,
      # состояние: # глоб лог := ложь, глоб лог := ложь,
        глоб лог := ложь, глоб лог := ложь,
        глоб лог:= истина,
      глоб позиция := (1,1,1), "",
      (станд код из кан) (книга),
      # процедуры обработки события: #
      ложь, ложь, ложь, ложь, ложь, ложь,
      (имя файл f, имя лит а) лог: ложь);
     (¬ возм двоичн (файл) |
        настроить на литерное (файл));
     (¬ возм ввод (файл) |
        настроить на запись (файл));
     (¬ возм вывод (файл) |
        настроить на чтение (файл));
     0
    все
   все
  иначе вверх защита связей; не определено
  все
конец;
проц (имя файл, строк, кан) цел open = открыть;
е) проц соединить =
    (имя файл файл, имя [ ] [ ] [ ] лит том) пуст:
    если цел р = нигр том; цел l = нигр том [p];
    цел с = нигр том [p] [l]; p = 1 ∧ l = 1 ∧ c = 1
   то
    проц и = (имя книга а) лог: истина;
    проц л = (имя книга а) лог: ложь;
    канал кан = (и, и, и, и, л, л, л, лог: ложь,
      позиция: (макс цел, макс цел, макс цел),
                пропуск, пропуск);
     (открыт из файл |
      вниз защита связей; вверх гномы);
     файл :=
      (глоб книга := (пропуск, (вегр том + 1,1,1 ),
      пропуск, истина, 1), том, кан, пропуск, пропуск,
      # состояние: # глоб лог := ложь, глоб лог := ложь,
      глоб лог := истина, глоб лог := ложь, глоб лог := истина,
```

```
        глоб позиция := (1,1,1), " ", пропуск,
        # процедуры обработки события: #
        ложь, ложь, ложь, ложь, ложь, ложь,
        (имя файл f, имя лит a) лог: ложь)
      иначе не определено
       все;
   проц (имя файл,
         имя [ ] [ ] [ ] лит) пуст associate = соединить;
f) проц ?файл доступен = (канал кан) лог:
    с истина, если в данный момент
      на 'кан'але можно открыть
      некоторый другой файл, а иначе ложь с;
g) проц ?обозн приемлемо = (строк обозн) лог: с истина,
      если 'обозн'ачение премлемо для данной реализации
      в качестве обозначения, идентифицирующего новую книгу,
      а иначе ложь с;
h) проц ?соответствует =
       (строк обозн, канал кан, имя книга имя книги) лог:
      с истина, если книга, именуемая через 'имя книги',
       может идентифицироваться посредством этого
       'обозн'ачения, и если к данной книге может быть
       правильный доступ через 'кан'ал, а иначе ложь с;
i) проц ?ложь = (имя файл файл) лог: ложь
     с включена для сокращения текста процедур
     'завести', 'открыть' и 'соединить' с;
j) проц ?настроить на запись = (имя файл f) пуст:
     если ¬ возм вывод (f) ∨
          ¬ возм установка (f) ∧
            для двоичн из f ∧ для чтения из f
     то не определено
     иначе имя лог (для чтения из f) := ложь;
       имя лог (для записи из f) := истина
     все;
k) проц ?настроить на чтение = (имя файл f) пуст:
     если ¬ возм ввод (f) ∨
       ¬ возм установка (f) ∧
          для двоичн из f ∧ для записи из f
     то не определено
     иначе имя лог (для чтения из f) := истина;
       имя лог (для записи из f) := ложь
     все;
l) проц ?настроить на литерное = (имя файл f) пуст:
     если ¬ возм установка (f) ∧ для двоичн из f
     то не определено
     иначе имя лог (для литер из f) := истина;
```

```
        имя лог (для двоичн из f) := ложь
      все;
m)  проц настроить на двоичное = (имя файл f) пуст:
      если ¬ возм двоичн (f) ∨¬
            возм установка (f) ∧ для литер из f
      то не определено
      иначе имя лог (для литер из f) := ложь;
        имя лог (для двоичн из f) := истина
      все;

n)  проц закрыть = (имя файл файл) пуст:
      если открыт из файл
      то
        вниз защита связей;
          имя лог (открыт из файл) := ложь;
          имя книга книга = книга из файл;
          запись из книга := ложь;
          пользователи из книга −:= 1;
          (текст из файл | (подвтекст) : доступные книги:=
              ПЕРВ связка := (книга, доступные книги));
        вверх гномы
      все;
    проц (имя файл) пуст close = закрыть;
o)  проц снять = (имя файл файл) пуст:
      если открыт из файл
      то
       вниз защита связей;
         имя лог (открыт из файл) := ложь;
         имя книга книга = книга из файл;
         запись из книга := ложь;
         пользователи из книга −:= 1;
         (текст из файл | (подвтекст) : снятые книги :=
            ПЕРВ связка := (книга, снятые книги));
        вверх гномы
      все;
    проц (имя файл) пуст lock = снять;
p)  проц стереть = (имя файл файл) пуст:
      если открыт из файл
      то
       вниз защита связей;
          имя лог (открыт из файл) := ложь;
         запись из книга из файл := ложь;
         пользователи из книга из файл −:= 1;
        вверх гномы
      все;
    проц (имя файл) пуст scratch = стереть;
```

10.3.1.5. Запросы позиции.

{aa) „Текущей позицией" книги, открытой с данным файлом является значение, именуемое полем тпоз этого файла. Она продвигается каждой операцией обмена в соответствии с числом записанных или прочитанных литер.

Если с — номер текущей литеры и lb — длина текущей строчки, то всегда $1 \leqslant c \leqslant lb + 1$. Из $c = 1$ вытекает, что следующая операция обмена будет применена к первой литере этой строчки, а из $c = lb + 1$ вытекает, что данная строчка переполнена и следующая операция обмена будет вызывать какую-то процедуру обработки события. Если $lb = 0$, то данная строчка пуста и, значит, всегда переполнена. Соответствующие оговорки относятся и к номерам текущих строчки и страницы. Если данная страница переполнена, то текущая строчка пуста, а если переполнена данная книга, то пусты и текущая страница, и текущая строчка (e).

bb) Пользователь может определить текущую позицию при помощи процедур номер литеры, номер строчки и номер страницы (a, b, c).

cc) Если в текущей позиции переполнена строчка, страница или книга, то говорят, что она находится за пределами „физического файла" (f, g, h).

dd) Если при чтении текущая позиция находится на логическом конце (книги), то говорят, что она находится за пределами „логического файла" (i).}

{Каждая процедура в этом разделе вызывает не определено, если при входе в нее не открыт соответствующий файл.}

```
a) проц номер литеры = (имя файл f) цел:
     (открыт из f | c из тпоз из f | не определено);
   проц (имя файл) цел char number = номер литеры;
b) проц номер строчки = (имя файл f) цел:
     (открыт из f | 1 из тпоз из f | не определено);
   проц (имя файл) цел line number = номер строчки;
c) проц номер страницы = (имя файл f) цел:
     (открыт из f | p из тпоз из f | не определено);
   проц (имя файл) цел page number = номер страницы;
d) проц ↑текущая позиция = (имя файл f) позиция:
     (открыт из f | тпоз из f | не определено; пропуск);
e) проц ↑границы книги = (имя файл f) позиция:
   начало позиции тпоз = текущая позиция (f);
     цел p = p из тпоз, l = l из тпоз;
     выб текст из f в
       (текст t1):
          (цел pb = вегр t1;
          цел lb = (p ⩽ 0 ∨ p > pb |0| вегр t1 [p]);
          цел cb = (l ⩽ 0 ∨ l > lb |0| вегр t1 [p] [l]);
          (pb, lb, cb)),
        (подвтекст t2):
          (цел pb = вегр t2;
```

```
            цел lb = (p ⩽ 0 ∨ p > pb |0| вегр t2 [p]);
            цел cb = (l ⩽ 0 ∨ l > lb |0| вегр t2 [p] [l]);
            (pb, lb, cb))
        быв
      конец;
f)  проц ?строчка окончена = (имя файл f) лог:
      (цел c = c из текущая позиция (f);
      c > c из границы книги (f));
g)  проц ?страница окончена = (имя файл f) лог:
      (цел l = l из текущая позиция (f);
      l > l из границы книги (f));
h)  проц ?физ файл окончен = (имя файл f) лог:
      (цел p = p из текущая позиция (f);
      p  > p из границы книги (f));
i) проц ?лог файл окончен = (имя файл f) лог:
      ¬ (заполн из книга из f вне текущая позиция (f));
```

10.3.1.6. Процедуры расположения.

{aa) В книге, введенной с внешнего насителя некоторой системной-задачей, строчки и страницы могут быть разной длины. С другой стороны, все строчки и страницы заведенной (10.3.1.4.cc) книги первоначально имеют размер, указанный пользователем. Однако если в ходе вывода в сжимаемую книгу (10.3.1.3.ff) вызывается нов строчка (нов страница) с текущей позицией в той же строчке (странице), что и логический конец этой книги, то данная строчка (страница, содержащая эту строчку,) сокращается до номера литеры (номера строчки) этого логического конца. Таким образом, печ (("абвг", нов строчка)) может вызвать уменьшение текущей строчки до длины в 5 литер. Отметим, что то же вполне осмысленно и для строчки, не содержащей литер, и для страницы, не содержащей строчек.

Хотя действия канала, на котором книги как сжимаемы, так и позволяют произвольный доступ (10.3.1.3.ff) корректно определены, предполагают, что в фактических реализациях такая комбинация вряд ли встретится.

bb) Процедуры вперед (a), нов строчка (c) и нов страница (d) служат продвижению текущей позиции к следующей литере, строчке и странице соответственно. Однако они не изменяют (за исключением того, что предусмотрено в cc ниже) содержания пропускаемых позиций. Поэтому печ (("а", назад, вперед)) и печ (("а", назад, пробел)) дают разные результаты.

Текущую позицию можно также изменить вызовами назад (b), уст номер литеры (k) и на соответствующих каналах вызовами установить (i) и уст нач (j).

cc) Содержание вновь заведенной книги не определено, при этом ее текущая позиция и логический конец оба равны (1,1,1). В процессе вывода она заполняется литерами и в соответствии с этим ее логический конец продвигается вперед. Если в ходе литерного вывода вызывается вперед с текущей позицией на логическом конце, то записывается литера пробел

(аналогичное происходит в случае вызова нов строчка и нов страница, если данная книга несжимаема).

Вызов установить, пытающийся оставить текущую позицию за пределами логического конца, приводит к вызову не определено (некоторое осмысленное действие системы может тогда продвинуть логический конец к данной текущей позиции или даже к физическому концу книги). Следовательно, нет законного способа, которым можно было бы вывести текущую позицию за пределы логического конца или который оставлял бы любую литеру в пределах логического файла в ее начальном неопределенном состоянии.

dd) Операция чтения или записи или вызов вперед, нов строчка, нов страница, установить или уст номер литеры могут вывести текущую позицию за пределы физического или логического файла (10.3.1.5.cc, dd), но это не вызывает немедленных последствий. Однако перед любой дальнейшей попыткой обмена или следующим вызовом вперед, нов строчка или нов страница (но не установить или уст номер литеры) текущая позиция должна сделаться „хорошей". Файл будет „хорошим", если его текущая позиция не находится при записи (чтении) вне физического (логического) файла (10.5.1.1.cc, dd). Страница (строчка) будет „хорошей", если не переполнен номер строчки (номер литеры). Следовательно, в нужных случаях вызывается процедура обработки события (10.3.1.3.cc), соответствующая при конце лог файла, при конце физ файла, при конце страницы или при конце строчки. Действие по умолчанию, за исключением форматного обмена (использующего проверить позицию, 10.3.3.2.c), состоит в том, чтобы вызвать, если процедура обработки ситуации вырабатывает ложь, соответственно не определено, не определено, нов страница или нов строчка. После этого (или если вырабатывается истина), если позиция остается не хорошей, снова вызывается процедура обработки события (не обязательно та же самая).

ee) Состояние файла (10.3.1.3.bb) управляет некоторыми действиями процедур расположения. Если настройка на запись/чтение дает чтение, то действие вперед, нов строчка и нов страница при попытке пересечь логический конец состоит в вызове процедуры обработки события, соответствующей при конце лог файла, вместе с вызовом по умолчанию не определено; если же она дает запись, то результатом будет вывод пробелов (при настройке на двоичное запись неопределенной литеры) либо сжатие текущей строчки или страницы (см. cc). Если при входе в процедуру расположения настройка на запись/чтение не определена, то вызывается не определено. По выходе восстанавливается та настройка на запись/чтение, которая была при входе.}

```
а) проц вперед = (имя файл f) пуст:
    если ¬ открыт из f то не определено
    иначе
      лог чтение =
        (для чтения из f | истина | : для записи из f | ложь
```

```
      | не определено; пропуск);
     (¬строчка хороша (f, чтение) | не определено);
     имя позиция тпоз = тпоз из f;
      если чтение то с из тпоз +:= 1
      иначе
      если лог файл окончен (f) то
        если для двоичн из f то
              (текст из f | (подвтекст t2):
              t2 [р из тпоз] [l из тпоз]
              [с из тпоз] := пропуск);
           с из тпоз +:= 1; заполн из книга из f := тпоз
         иначе вывести литеру (f, "–")
         все
       иначе с из тпоз +:= 1
       все
      все
     все;
   проц (имя файл) пуст space = вперед;
b) проц назад = (имя файл f) пуст;
   если ¬ открыт из f то не определено
   иначе имя цел с = с из тпоз из f;
     (с > 1 | с –:= 1 | не определено)
    все;
   проц (имя файл) пуст back space = назад;
c) проц нов ≠ая ≠строчка = (имя файл f) пуст:
   если ¬ открыт из f то не определено
   иначе
    лог чтение =
     (для чтения из f | истина | : для записи из f | ложь
    | не определено; пропуск);
     (¬страница хороша (f, чтение) | не определено);
     имя позиция тпоз = тпоз из f,
      заполн = заполн из книга из f;
     если р из тпоз = р из заполн ∧ l из тпоз =l из заполн
     то с из тпоз := с из заполн;
      если чтение то нов строчка (f)
      иначе
      если сжимаем(f)
      то имя цел pl = р из заполн, ll = l из заполн;
        подвтекст текст = (текст из f |
        (подвтекст t2) : t2);
        текст [pl] [ll] := текст [pl] [ll]
        [: с из заполн–1]
      иначе пока ¬ строчка окончена (f) цк вперед (f) кц
      все;
```

```
        тпоз := заполн := (р из тпоз, l из тпоз + 1,1)
       все
      иначе тпоз := (р из тпоз, l из тпоз + 1,1)
       все
     все;
   проц (имя файл) пуст new line = нов строчка;
d) проц нов страница = (имя файл f) пуст:
    если ¬ открыт из f то не определено
    иначе
     лог чтение =
       (для чтения из f | истина |:для записи из f | ложь
       | не определено; пропуск);
      (¬ страница хороша (f, чтение) | не определено);
      имя позиция тпоз = тпоз из f,
      ппоз = заполн из книга из f;
      если р из тпоз = р из ппоз
      то тпоз := ппоз;
        если чтение то нов страница (f)
        иначе
         если сжимаем (f) ∧
          l из ппоз ⩽ l из границы книги (f)
          то имя цел pl = р из ппоз, ll = l из ппоз;
            подвтекст текст = (текст из f |
            (подвтекст t2) : t2);
            текст [pl] [ll] := текст [pl] [ll]
            [: с из ппоз – 1];
            текст [pl] := текст [pl]
            [: (с из ппоз > 1 | ll | ll – 1)]
          иначе пока ¬ страница окончена (f)
            цк нов строчка (f) кц
          все;
           тпоз := ппоз := (р из тпоз + 1,1,1)
         все
        иначе тпоз := (р из тпоз + 1,1,1)
        все
      все;
   проц имя файл пуст newpage = нов страница;
```

{Каждая из трех следующих процедур либо вырабатывает истина и тогда соответствующая строчка, страница или файл хороши (dd), либо вырабатывает ложь и тогда текущая позиция может оказаться за пределами логического файла или может переполниться номер страницы, либо же она циклится до тех пор, пока не устранится соответствующая причина, либо такая процедура прекращается в результате какого-то перехода. При выходе из такой процедуры настройка на запись/чтение определяется ее параметром чтение.}

```
e)  проц строчка хороша = (имя файл f, лог чтение) лог:
      начало лог не оконч;
       пока не оконч := страница хороша (f, чтение);
        строчка окончена (f) ∧ не оконч
       цк (¬ (испр строчка из f) (f) | настроить (f, чтение);
        нов строчка (f)) кц;
       не оконч
      конец;
f)  проц страница хороша = (имя файл f, лог чтение) лог:
      начало лог не оконч;
       пока не оконч := файл хорош (f, чтение);
         страница окончена (f) ∧ не оконч
       цк (¬ (испр страница из f) (f) | настроить (f, чтение);
        нов страница (f)) кц;
       не оконч
      конец;
g)  проц файл хорош = (имя файл f, лог чтение) лог:
      начало лог не оконч := истина;
       пока настроить (f, чтение);
        не оконч ∧
        (чтение | лог файл окончен | физ файл окончен) (f)
       цк не оконч := (чтение | испр лог файл из f
       | испр физ файл из f) (f) кц;
       не оконч
      конец;
h)  проц настроить = (имя файл f, лог чтение) пуст:
      (чтение | настроить на чтение (f) |
      настроить на запись (f));
i)  проц установить = (имя файл f, цел p, l, c) пуст:
      если ¬ открыт из f ∨
      ¬ возм установка (f) то не определено
      иначе лог чтение = (для чтения из f | истина
        | : для записи из f | ложь | не определено; пропуск);
       имя позиция тпоз = тпоз из f,
       ппоз = заполн из книга из f;
       позиция раб тпоз = тпоз;
       если (тпоз := (p, l, c)) вне ппоз
       если тпоз := ппоз;
         (¬ (испр лог файл из f) (f) | не определено);
        настроить (f, чтение)
       инес позиция границы = границы книги (f);
        p < 1 ∨ p > p из границы + 1 ∨
        l < 1 ∨ l > l из границы + 1 ∨
        c < 1 ∨ c > c из границы + 1
       то тпоз := раб тпоз; не определено
```

```
      все
     все;
    проц (имя файл, цел, цел, цел) пуст set = установить;
j)  проц уст нач = (имя файл f) пуст:
     если ¬ открыт из f ∨ ¬возм уст нач (f) то не определено
     иначе
       имя лог (для чтения из f) := ¬возм вывод (f);
       имя лог (для записи из f) := ¬возм ввод (f);
       имя лог (для литер из f) := ¬возм двоичн (f);
       имя лог (для двоичн из f) := ложь;
       имя позиция (тпоз из f) := (1,1,1)
     все;
    проц (имя файл) пуст reset = уст нач;
k)  проц уст номер литеры = (имя файл f, цел с) пуст:
     если ¬ открыт из f то не определено
     иначе имя имени позиция тпоз = тпоз из f;
      пока с из тпоз  ≠ с
      цк
       если с < 1 ∨ с > с из границы книги (f) + 1
       то не определено
       инес с > с из тпоз
       то вперед (f)
       иначе назад (f)
       все
      кц
     все;
    проц (имя файл, цел) пуст set char number =
     уст номер литеры;
```

10.3.2. *Значения для обмена*

10.3.2.1. Процедуры преобразования.

{Процедуры целое, фикс и плав предназначены для использования с процедурами бесформатного обмена вывод, печ и зап в тех случаях, когда требуется небольшое добавочное управление над порождаемым расположением. Каждая из этих процедур имеет параметр разрядность, абсолютная величина которого задает длину строки, порождаемой преобразованием предъявляемого арифметического значения V. И фикс, и плав имеют параметр после, задающий число цифр, требуемых после десятичной точки, а параметр порядок в плав задает разрядность, отводимую для порядка. Если V нельзя выразить строкой в пределах заданного разрядность, – даже за счет уменьшения значения после, если оно есть, – то вместо этого вырабатывается строка, заполненная литерами литера ошибки (10.2.1.t).

Знак включается обычным образом, и начальные нули заменяются пробелами. Однако пользователь может указать, задавая отрицательную разрядность, чтобы знак включался только для отрицательных значений. Если задана разрядность нуль, то вырабатывается кратчайшая возможная строка,

в которую, согласно остальным параметрам, можно преобразовать V. Следующие примеры иллюстрируют некоторые из этих возможностей:

```
    печ (целое (i, –4)),
      которое может печатать "÷÷÷0", "÷÷99", "÷–99", „9999"
      или, если i было больше, чем 9999, „****", где „*" –
      выдача литера ошибки;
    печ (целое (i, 4)),
      которое будет печатать "÷+99", а не "÷÷99";
    печ (целое (i, 0)),
      которое может печатать „0", „99", „–99", „9999" или
      „99999";
    печ (фикс (x, –6, 3)),
       которое может печатать „2.718", „27.183",
      или „271.83" (в последнем случае пожертвовано одним
      местом после десятичной точки, чтобы приспособиться
      к данному числу);
    печ (фикс (x, 0, 3)),
      которое может печатать „2.718", „27.183" или „271.828";
    печ (плав (x, 9, 3, 2)),
      которое может печатать „–2.718₁₀ + 0", „+2.718₁₀ –1"
      или „+2.72₁₀ +11" (в последнем случае пожертвовано
      одним местом после десятичной точки, чтобы освободить
      пространство для неожиданно большого порядка).}
a)  вид число = об (⊰ Д вещ ⊱, ⊰ Д цел ⊱);
b) проц целое = (число v, цел разрядность) строк:
     выб v в
     ⊰ (Д цел x):
       (цел длина := абс разрядность – (x < Д 0 ∨ разрядность > 0 | 1 | 0),
       Д цел n := абс x;
       если разрядность = 0 то
       Д цел m := n; длина := 0;
        пока m ÷:= Д 10; длина +:= 1; m ≠ Д 0
       цк пропуск кц
      все;
       строк s := предст целого (n, длина);
       если длина = 0 ∨
        литера в строке (литера ошибки, лок цел, s)
       то абс разрядность × литера ошибки
       иначе
        (x < Д 0 | "–" | : разрядность > 0 |"+"| " ")прип s;
       (разрядность ≠ 0|
        (абс разрядность – вегр s) × ". " прип s);
        s
       все) ⊱,
     ⊰ (Д вещ x) : фикс (x, разрядность, 0) ⊱
```

```
    быв;
    проц (число, цел) строк whole = целое;
c)  проц фикс = (число v, цел разрядность, после) строк:
     выб v в
      ≮ (Д вещ x):
        если цел длина := абс разрядность –
       (x < Д 0 ∨ разрядность > 0 | 1 | 0); после ⩾ 0 ∧
       (длина > после ∨ разрядность = 0)
      то Д вещ y = абс x;
       если разрядность = 0
       то длина := (после = 0 | 1 | 0);
         пока y + Д.5 × Д .1 ↑ после ⩾ Д 10 ↑ длина
         цк длина +:= 1кц;
         длина +:= (после = 0 | 0 | после + 1)
       все;
       строк s := предст рационального (y, длина, после);
       если ¬ литера в строке (литера ошибки, лок цел, s)
       то (длина > вегр s ∧ y < Д 1.0 | "0" прип s);
         (x < Д 0 | „–" | : разрядность > 0 | "+" | " ") прип s;
         (разрядность ≠ 0 |
         (абс разрядность – вегр s) × "." прип s);
         s
       инес после > 0
      то фикс (v, разрядность, после – 1)
        иначе абс разрядность × литера ошибки
        все
       иначе не определено; абс разрядность × литера ошибки
       все ≯,
      ≮ (Д цел x): фикс (Д вещ (x), разрядность, после) ≯
      быв;
    проц (число, цел, цел) строк fixed = фикс;
d)  проц плав = (число v, цел разрядность,
       после, порядок) строк:
     выб v в
     ≮ (Д вещ x):
     если цел прежде = абс разрядность – абс порядок –
       (после ≠ 0 | после + 1 | 0) – 2;
       знак прежде + знак после > 0
     то строк s, Д вещ y := абс x, цел p := 0;
       Д нормализовать (y, прежде, после, p);
       s :=
         фикс (знак x × y, знак разрядность ×
         (абс разрядность – абс порядок – 1), после) + „10" +
           целое (p, порядок);
      если порядок = 0 ∨ литера в строке (литера ошибки,
```

```
        лок цел, s)
       то
        плав (x, разрядность, (после ≠ 0 | после – 1 | 0),
             (порядок > 0 | порядок + 1 | порядок – 1))
       иначе s
       все
      иначе не определено; абс разрядность X литера ошибки
      все ⊱,
     ⊰ (Д цел x) : плав (Д вещ (x),
       разрядность, после, порядок) ⊱
     быв;
    проц (число, цел, цел, цел) строк float = плав;
e)  проц ^предст #авление #целого =
       (число v, цел разр) строк:
     #вырабатывает строку с максимальной длиной
     'разр'ядность, содержащую десятичное представление
     положительного целого числа 'v' #
     выб v в
     ⊰ (Д цел x) :
      начало строк s, Д цел n := x;
       пока цифру в литеру (С (n мод Д 10)) прип s;
       n ÷ := Д 10; n ≠ Д 0
      цк пропуск кц;
       (вегр s > разр | разр X литера ошибки | s)
     конец ⊱
     быв;
f)  проц ^предст #авление #рационального =
       (число v, цел разр, после) строк:
     #вырабатывает строку с максимальной длиной
      'разр'ядность, содержащую округленное десятичное
      представление положительного вещественного числа
      'v'; если 'после' больше нуля, эта строка содержит
      десятичную точку, за которой следует 'после'
      цифр #
     выб v в
     ⊰ (Д вещ x) :
      начало строк s, цел прежде := 0;
       Д вещ y := x + Д.5 X Д.1 ↑ после;
      проц подб #ор #циф #ры #= (имя Д вещ y) лит:
     цифру в литеру ((цел c := С целч (y X:= Д 10.0);
     (c > 9 | c := 9); y –:= У c; c));
    пока y ⩾ Д 10.0 ↑ прежде цк прежде +:= 1 кц;
    y /:= Д 10.0 ↑ прежде;
    до прежде цк s плюспр подб циф (y) кц;
    (после > 0 | s плюспр ".");
```

```
      до после цк s плюспр подб циф (у) кц;
      (вегр s > разр | разр × литера ошибки | s)
    конец ⊁
    быв;
g)  проц ⊀Д нормализовать = (имя Д вещ у, цел прежде, после,
            имя цел р)
        пуст: # приспосабливает значение 'у' к тому, чтобы
        оно могло подвергаться обмену согласно формату ф n
        (прежде) d . n (после) d ф;
        'р' устанавливается таким, чтобы у × 10 ↑ р равнялось
        первоначальному значению 'у' #
    начало
      Д вещ g = Д 10.0 ↑ прежде; Д вещ h = g × Д.1;
      пока у ⩾ g цк у ×:= Д.1; р +:= 1 кц;
      (у ≠ Д 0.0 | пока у < h цк у ×:= Д 10.0; р −:= 1 кц);
      (у + Д.5 × Д.1 ↑ после ⩾ g | у := h; р +:= 1)
    конец;
h)  проц ⊀цифру в литеру = (цел х) лит:
      „0123456789abcdef" [х + 1];
i)  проц ⊀строку в Д цел = (строк s, цел основание,
      имя Д цел i) лог:
      # вырабатывает истина, если абсолютное значение
        результата не больше Д макс цел #
    начало
      Д цел lr = У основание; лог безопасно := истина;
      Д цел n := Д 0, Д цел m = Д макс цел ÷ lr;
      Д цел m1 = Д макс цел − m × lr;
      для i от 2 до вегр s
      пока Д цел циф = Y литеру в цифру (s [i]);
        безопасно := n < m ∨ n = m ∧ циф ⩽ m1
      цк n := n × lr + циф кц;
      если безопасно то i := (s [1] ="+" |n| − n); истина
      иначе ложь все
    конец;
j)  проц ⊀строку в Д вещ = (строк s, имя Д вещ r) лог:
      # вырабатывает истина, если абсолютное значение
        результата не больше, чем Д макс вещ #
    начало
      цел e := вегр s + 1;
      литера в строке ("10", e, s);
      цел р := e; литера в строке (". " р, s);
      цел j := 1, длина := 0, Д вещ х := Д 0.0;
        # пропуск начальных нулей: #
        для i от 2 до e − 1
        пока s [i] ="0" ∨ s [i] = ". " ∨ s [i] ="-"
```

```
      цк j := i кц;
      для i от j + 1 до e – 1 пока длина <
       Д разрядность вещ
      цк
       если s [i] ≠ ". "
       то x := x × Д 10.0 + У литеру в цифру (s [j := i]);
         длина +:= 1
       все ♯ все значащие  цифры преобразованы ♯
      кц;
      ♯ предварительно установить порядок: ♯
      цел пор ♯ ядок ♯ := (p > j | p – j – 1 | p – j), показ := 0;
      ♯ преобразовать порядок: ♯
      лог безопасно :=
        если e < вегр s
        то строку в Д цел (s [e + 1 :], 10, показ)
       иначе истина
       все;
      ♯ подготовить представление Д макс вещ для
        сравнения с Д вещ значением, которое должно быть
        выработано: ♯
      Д вещ макс мант ♯ исса ♯ := Д макс вещ,
        цел макс пор ♯ ядок ♯ := 0;
      Д нормализовать (макс мант, длина, 0, макс пор);
        пор +:= показ;
      если ¬ безопасно ∨ (пор > макс пор ∨ пор = макс пор ∧
        x > макс мант)
        то ложь
        иначе r := (s [1] = "+" | x | – x) × Д 10.0 ↑ пор; истина
       все
     конец;
k)  проц ? литеру в цифру = (лит x) цел:
      (x = "." | 0 | цел i;
      литера в строке (x, i, " 0123456789   abcdef" ); i – 1);
l)  проц литера в строке = (лит c, имя цел i, строк s) лог:
      (лог есть := ложь;
      для k от нигр s до вегр s пока ¬ есть
      цк (c = s [k] | i := k; есть := истина) кц;
      есть);
      проц (лит, имя цел, строк) лог char in string = литера в
       строке;
m)  цел Д разрядность цел =
      ♯ наименьшее целое значение, такое, что 'Д макс цел'
        можно преобразовать без ошибки, используя трафарет
        n (Д разрядность цел) d ♯
      (цел c := 1;
```

```
    пока Д 10 ↑ (с – 1) < Д.1 × Д макс цел цк с +:= 1 кц;
    с);
  цел L int width = Д разрядность цел;
n) цел Д разрядность вещ =
    # наименьшее целое значение, такое, что при преобразовании
     '1.0' и '1.0 + Д точность вещ', с помощью трафарета
     d.n (Д разрядность вещ – 1) d порождаются
     разные строки #
   1 – С целч (Д лг (Д точность вещ) / Д лг (Д 10));
  цел L real width = Д разрядность вещ;
o) цел Д разрядность порядка =
    # наименьшее целое значение, такое, что 'Д макс вещ'
    можно преобразовать без ошибки, используя трафарет
    d.n (Д разрядность вещ – 1) den (Д разрядность
     порядка) d #
    1 + С целч (Д лг (Д лг (Д макс вещ) / Д лг (Д 10)) /
    Д лг (Д 10));
    цел L exp width = Д разрядность порядка;
```

10.3.2.2. Виды для обмена.

a) вид ♣провывод # простые для вывода # = об (⋖ Д цел ⋗,
⋖ Д вещ ⋗, ⋖ Д компл ⋗, лог, ⋖ Д бит ⋗, лит, [ ] лит);

b) вид ♣выводимое = с фактический-описатель, специфицирующий вид, объединенный из {2.1.3.6.a} достаточно множества видов, ни один из которых не является 'пустым значением', и не содержит 'подвижное', 'имя', 'процедура', 'объединение' с;

c) вид ♣проввод # простые для ввода # = об (⋖ имя Д цел ⋗,
⋖ имя Д вещ ⋗, ⋖ имя Д компл ⋗, имя лог,
⋖ имя Д бит ⋗, имя лит, имя [ ] лит, имя строк);

d) вид ♣вводимое = с фактический-описатель, специфицирующий вид, объединенный из {2.1.3.6.a} 'имени подвижного вектора из литерных' вместе с достаточным множеством видов, каждый из которых есть 'имя', за которым следует вид, не содержащий 'подвижное', 'имя', 'процедура', 'объединение' с;

{ См. замечания после 10.2.3.1, касающиеся термина „достаточное множество". }

10.3.2.3. Выстраивание.

a) оп ♣стройвывод = (выводимое х) [ ] провывод:
с результат „выстраивания" 'х' с;

b) оп ♣ стройввод = (вводимое х) [ ] проввод:
с результат выстраивания 'х' с;

c) Результатом „выстраивания" данного значения V является {одномерный} массив W, получаемый следующим образом:

- требуется, чтобы V {если оно есть имя,} не было псевдоименем;
- некоторый счетчик i устанавливается в 0;
- V „проходится" {d} с помощью i;

• W составляется из паспорта ((1, i)) и получаемых прохождением V элементов;

• если V не есть (есть) имя, то вид результата является видом, специфицируемым описателем [ ] провывод ([ ] проввод).

d) Значение V „проходится с помощью счетчика следующим образом:

Если V – значение (именует значение), из которого объединен вид, специфицируемый описателем провывод,

то

• i увеличивается на единицу;

• элементом, выбираемым по (i) в W, является V;

иначе

Случай A: V – {одномерный} массив (именует одномерный массив) с паспортом ((l, u));

• для j = l, ..., u элемент, выбираемый (подымя, выбираемое) по (j) в V, проходится с помощью i;

Случай B: V – {n-мерный, n ⩾ 2} массив (именует массив) с паспортом $((l_1, u_1), (l_2, u_2), ..., (l_n, u_n))$, где n ⩾ 2;

• для $j = l_1, ..., u_1$ массив, выбираемый {2.1.3.4.i} (имя, генерируемое {2.1.3.4.j} по отрезку $(j, (l_2, u_2, 0) ..., (l_n, u_n, 0))$, проходится с помощью i;

Случай C: V – структура (именует структуру) V 1:

• поля (подымена V, именующие поля) этой V1, взятые в их порядке, проходятся с помощью i.

10.3.3. *Бесформатный обмен*

{При бесформатном обмене элементы „списка данных" обмениваются один за другим через заданный файл. Каждый элемент этого списка данных является либо процедурой расположения, вид которой специфицируется проц (имя файл) пуст (10.3.1.6), либо значением вида, специфицируемого посредством выводимое (при выводе) или вводимое (при вводе). Когда процедура расположения встречается в списке данных, она вызывается с заданным файлом в качестве параметра. Другие значения в списке данных сначала выстраиваются (10.3.2.3), а затем результирующие значения одно за другим обмениваются через заданный файл.

Обычно обмен происходит на текущей позиции, но если (при выводе) недостаточно места в текущей строчке или (при вводе) на текущей позиции нет читаемого значения, то сначала вызывается процедура обработки события, соответствующая при конце строчки (или, где это целесообразно, при конце страницы, при конце физ файла или при конце лог файла), а затем, если она вырабатывает ложь, в данной книге ищется следующая „хорошая" позиция литеры, а именно первая позиция литеры в следующей непустой строчке.}

10.3.3.1. Бесформатный вывод.

{Для бесформатного вывода можно использовать вывод (а) и печ (или зап) (10.5.1.d). Каждое выстроенное значение V из списка данных выводится следующим образом:

aa) Если вид этого V специфицируется посредством Д цел, то сначала, если в остатке текущей строчки нет достаточно места для Д разрядность цел + 2 литер, отыскивается хорошая позиция на некоторой последующей строчке (см. 10.3.3), а затем, если это не начало строчки, дается один пробел и V выводится как бы под управлением шаблона

n (Д разрядность цел – 1) z + d.

bb) Если вид этого V специфицируется посредством Д вещ, то сначала, если в текущей строчке нет достаточного места для Д разрядность вещ + Д разрядность порядка + 5 литер, отыскивается хорошая позиция на некоторой последующей строчке, а затем, если это не начало строчки, дается один пробел и V выводится как бы под управлением шаблона

+ d.n (Д разрядность вещ – 1) den (Д разрядность порядка – 1) z + d.

cc) Если вид этого V специфицируется посредством Д компл, то сначала, если в текущей строчке нет достаточно места для 2 X (Д разрядность вещ + Д разрядность порядка) + 11 литер, отыскивается хорошая позиция на некоторой последующей строчке, а затем, если это не начало строчки, дается один пробел и V выводится как бы под управлением шаблона

+ d.n (Д разрядность вещ – 1) den (Д разрядность порядка – 1) z
+ d"–" i + d.n (Д разрядность вещ – 1) den (Д разрядность порядка – – 1) z + d.

dd) Если вид этого V специфицируется посредством лог, то сначала, если текущая строчка полна, отыскивается хорошая позиция в некоторой последующей строчке, а затем, если V есть истина (ложь), выводится литера, выдаваемая да (нет) (без прерывающего пробела).

ee) Если вид этого V специфицируется посредством Д бит, то элементы единственного поля этого V выводятся (как в dd) один за другим (без прерывающих пробелов, но с переходом на новую строчку, когда это требуется).

ff) Если вид этого V специфицируется посредством лит, то сначала, если текущая строчка полна, отыскивается хорошая позиция в некоторой последующей строчке, а затем выводится V (без прерывающего пробела).

gg) Если вид этого V специфицируется посредством [ ] лит, то элементы этого V выводятся (как в ff) один за другим (без прерывающих пробелов, но с переходом на новую строчку, когда это требуется). }

```
а) проц вывод = (имя файл f,
   [ ] об (выводимое, проц (имя файл) пуст) x) пуст:
   если открыт из f то
   для i до вегр x
   цк выб настроить на запись (f);
     настроить на литерное (f);
       x [i] в (проц (имя файл) пуст пиф) : пиф (f),
       (выводимое выв) :
       начало
         [ ] провывод y = стройвывод выв;
           ∢ проц Д преоб ∤ разование ∤ вещ =
```

```
        (Д вещ r) строк:
       плав (r, Д разрядность вещ + Д разрядность
        порядка + 4, Д разрядность вещ – 1,
        Д разрядность порядка + 1) ≯;
     для j до вегр y
     цк выб y [j] в
     (об (число, ≮ Д компл ≯) чиском):
      начало строк s :=
       выб чиском в
       ≮ (Д цел k): целое (k, Д разрядность цел + 1) ≯,
       ≮ (Д вещ r): Д преоб вещ (r) ≯,
       ≮ (Д компл w): Д преоб вещ (вч w) | "⊥"
        + Д преоб вещ (мч w) ≯
       быв;
       имя имени позиция тпоз = тпоз из f,
        цел n = вегр s;
      пока
       след позиция (f);
       (n > с из границы книги (f) | не определено);
       с из тпоз + (с из тпоз = 1 |n| n + 1) >
       с из границы книги (f) + 1
      цк ¬(испр строчка из f) (f) |
       вывод (f, нов строчка));
       настроить на запись (f)
      кц;
      (с из тпоз ≠ 1 | "⊥" прип s);
      для k до вегр s цк вывести литеру (f, s [k]) кц
     конец ⋕ вывода чисел ⋕
    (лог b): (след позиция (f);
     вывести литеру (f, (b |да| нет))),
    ≮ (Д бит дбит):
       для k до Д размер бит
       цк вывод (f, (Д F из дбит) [k]) кц ≯,
     (лит k): (след позиция (f); вывести литеру (f, k)),
     ([ ] лит стр):
       для k от нигр стр до вегр стр
       цк след позиция (f); вывести литеру (f, стр [k]) кц
     быв кц
    конец
   быв кц
  иначе не определено
  все;
 проц (имя файл, [ ] об (выводимое, проц (имя файл) пуст))
  пуст put = вывод;
b) проц ?вывести литеру = (имя файл f, лит лит) пуст:
```

```
    если открыт из f ∧ ¬ строчка окончена (f)
    то имя позиция тпоз = тпоз из f,
     ппоз = заполн из книга из f;
    настроить на литерное (f); настроить на запись (f);
    имя цел p = p из тпоз, l = l из тпоз, c = c из тпоз;
    лит k; лог есть := ложь;
    выб текст из f в
      (текст): (k := лит; есть := истина),
      (подвтекст):
      для i до вегр F из код из f пока ¬ есть
      цк ст (лит внутр, внешн) табл = (F из код из f) [i];
        (внутр из табл = лит | k := внешн из табл:
          есть := истина)
      кц
  быв;
  если есть то
     выб текст из f в
       (текст t1): t1 [p] [l] [c] := k
      (подвтекст t2): t2 [p] [l] [c] := k
     быв;
     c +:= 1;
     если тпоз вне ппоз то ппоз := тпоз
     инес ¬ возм установка (f) ∧
      позиция (p из ппоз, l из ппоз, 1) вне тпоз
    то ппоз := тпоз;
      (сжимаем (f) |
        с размер строчки и страницы, содержащей логический размер данной книги, и всех последующих строчек и страниц может увеличиться { например, до размеров, с которыми книга была заведена (10.3.1.4.cc) первоначально, или до размеров, предполагаемых из макс позиция из кан из f } с)
     все
   иначе k :="⊥";
     если ¬ (испр ошибка литеры из f) (f, k)
     то не определено; k :="⊥"
     все;
     проверить позицию (f); вывести литеру (f, k)
    все
  иначе не определено
  все # настройка на запись сохраняется #;
:) проц ↑ след позиция = (имя файл f) пуст:
  (¬ строчка хороша (f, для чтения из f) | не определено)
   # строчка теперь хороша { 10.3.1.6.dd } , а настройка на
     запись/чтение такая же, как и при входе #;
```

10.3.3.2. Бесформатный ввод.

{ Для бесформатного ввода можно использовать ввод (а) и чит

(10.5.1.e). Значения из данной книги присваиваются каждому выстроенному имени N из списка данных следующим образом:

aa) Если вид этого N специфицируется посредством имя Д цел, то сначала в данной книге ищется первая литера, не являющаяся пробелом (при необходимости отыскиваются хорошие позиции в последующих строчках); затем из данной книги читается наибольшая строка, которую можно „составить" (10.3.4.1.1.kk) под управлением некоторого шаблона, имеющего форму

+ n (k1) "÷" n (k2) dd или n (k2) dd

(где k1 и k2 выдают произвольные неотрицательные целые числа); эта строка преобразуется в целое число и присваивается N; если такое преобразование не было успешным, то вызывается процедура обработки события, соответствующая при ошибке значения.

bb) Если вид этого N специфицируется посредством имя Д вещ, то сначала в данной книге ищется первая литера, не являющаяся пробелом (при необходимости отыскиваются хорошие позиции в последующих строчках); затем из данной книги читается наибольшая строка, которую можно составить под управлением некоторого шаблона, имеющего форму

+ n (k1) "÷" n (k2) d или n (k2) d,

с идущими затем

n (k3) dd или ds.

и, возможно, далее

e n (k4) "÷" + n (k5) "÷" n (k6) dd или

e n (k5) "÷" n (k6) dd;

эта строка преобразуется в вещественное число и присваивается N; если такое преобразование не было успешным, то вызывается процедура обработки события, соответствующая при ошибке значения.

cc) Если вид этого N специфицируется посредством имя Д компл, то сначала вводится (как в bb) вещественное число и присваивается первому подымени этого N; затем в данной книге ищется первая литера, не являющаяся пробелом, а потом вводится некоторая литера и, если она не является ни "⊥", ни "i", ни "и", вызывается процедура обработки события, соответствующая при ошибке литеры (10.3.1.3.cc), с предлагаемой литерой "⊥"; наконец, вводится вещественное число и присваивается второму подымени данного N.

dd) Если вид этого N специфицируется посредством имя лог, то сначала в данной книге ищется первая литера, не являющаяся пробелом (при необходимости отыскиваются хорошие позиции в последующих строчках); затем читается некоторая литера; если эта литера та же, что и выдаваемая да (нет), то N присваивается истина (ложь), а в противном случае вызывается

процедура обработки события, соответствующая при ошибке литеры, с предлагаемой литерой нет.

ее) Если вид этого N специфицируется посредством имя Д бит, то производится ввод (как в dd) для подымен этого N, одного за другим (с переходом на новую строчку, когда это требуется).

ff) Если вид этого N специфицируется посредством имя лит, то сначала, если текущая строчка исчерпана, отыскивается хорошая позиция на некоторой последующей строчке, а затем читается и присваивается N некоторая литера.

gg) Если вид этого N специфицируется посредством имя [ ] лит, то производится ввод (как в ff) для подымен этого N, одного за другим (с переходом на новую строчку, когда это требуется).

hh) Если вид этого N специфицируется посредством имя строк, то литеры читаются до тех пор,

(i) пока не встретится литера, содержащаяся в строке, присоединенной к данному файлу вызовом процедуры задать стопстроку,

(ii) либо пока не исчерпается текущая строчка, вследствие чего вызывается процедура обработки события, соответствующая при конце строчки (или, где это целесообразно, при конце страницы, при конце физ файла или при конце лог файла); если данная процедура обработки события продвигает текущую позицию к хорошей позиции (см. 10.3.3), то ввод литер возобновляется.

Строка, состоящая из введенных литер, присваивается N (отметим, что если текущая строчка была исчерпана либо текущая позиция была на начале пустой строчки или вне логического файла, то этому N присваивается пустая строка).}

```
а) проц ввод = (имя файл f,
      [ ] об (вводимое, проц (имя файл) пуст) х) пуст:
     если открыт из f то
     для i до вегр х
     цк выб настроить на чтение (f);
       настроить на литерное (f);
       х [i] в
    (проц (имя файл) пуст пиф) : пиф (f),
    (вводимое вв) :
    начало
      [ ] проввод у = стройввод вв; лит k; лог k пусто;
      оп ? = (строк s) лог:
        # вырабатывает истина, если следующая литера,
          когда она есть, в текущей строчке содержится
          в 's' (эта литера присваивается 'k'),
          а иначе ложь #
        если k пусто ∧ (строчка окончена (f) ∨ лог файл
          окончен (f))
        то ложь
        иначе (k пусто | ввести литеру (f, k));
```

```
  k пусто := литера в строке (k, лок цел, s)
 все;
оп ? = (лит c) лог: ?строк (c):
прио ! = 8:
оп ! = (строк s, лит c) лит:
  ⋕ запрашивает литеру, содержащуюся в ,'s': если
    читается литера, не входящая в 's', вызывается
    процедура обработки события, соответствующая
    'при ошибке литеры', с предлагаемой литерой 'c' ⋕
 если (k пусто | проверить позицию (f);
   ввести литеру (f, k)):
   k пусто := истина;
   литера в строке (k, лок цел, s)
 то k
 иначе лит предл := c;
  если (испр ошибка литеры из f) (f, предл) то
    (литера в строке (предл, лок цел, s)
    | предл | не определено; c)
  иначе не определено; c
  все:
  настроить на чтение (f)
 все;
  оп ! = (лит s, c) лит: строк (s) ! c;
  проц проп ⋕ пуск ⋕ нач ⋕ альных ⋕ пробелов =
   пуст:
   пока (k пусто | след позиция (f));
   ? "–" цк пропуск кц;
 проц проп ⋕ уск ⋕ пробелов = пуст:
   пока ?"–" цк пропуск кц;
 проц чит ⋕ ать ⋕ циф ⋕ ры ⋕ = строк:
    (строк f :="0123456789" !"0":
   пока ? "0123456789" цк t плюспр k кц; t);
 проц чит знак = лит:
   (лит t = (проп пробелов; ? "+–" | k | "+");
   проп пробелов; t);
 проц чит чис ⋕ ло ⋕ = строк:
   (лит t = чит знак;  t + чит циф);
 проц чит вещ ⋕ естественное ⋕ = строк:
   (строк t := чит знак;
   (¬? " . " | t плюспр чит циф | k пусто := ложь);
   (  ? " . " t плюспр " . " + чит циф);
   (  ? "10\e" | t плюспр "10" + чит чис); t);
 для j до вегр y
 цк лог не конч ⋕ ено ⋕ := ложь;
```

```
  k пусто := истина;
  выб у [j] в
  ∢ (имя Д цел идц):
     (проп нач пробелов;
      не конч := ¬строку в Д цел (чит чис, 10,
     идц)) ∌,
  ∢ (имя Д вещ идв):
     (проп нач пробелов;
       не конч:=
     ¬ строку в Д вещ (чит вещ, идв)) ∌,
  ∢ (имя Д компл идк):
     (проп нач пробелов;
      не конч:=
     ¬ строку в Д вещ (чит вещ, ивч идк);
     проп пробелов; „i и ⊥ „!" ⊥";
     не конч := не конч ∨
     ¬ строку в Д вещ (чит вещ, имч идк)) ∌,
     (имя лог ил):
       (проп нач пробелов;
       ил := (да + нет) ! нет = да),
   ∢ (имя Д бит идб):
     для i до Д размер бит
     цк ввод (f, (Д F из идб) [i]) кц ∌,
   (имя лит ил): (след позиция (f):
     ввести литеру (f, ил)).
   (имя [ ] лит имл):
      для i от нигр имл до вегр имл
      цк след позиция (f);
      ввести литеру (f, имл [i]) кц;
   (имя строк ис):
     начало строк t:
     пока проверить позицию (f);
       если строчка окончена (f)
          ∨ лог файл окончен (f)
       то ложь
       иначе ввести литеру (f, k);
       k пусто := ¬литера в строке (k,
                     лок цел, стопс из f)
       все
     цк t плюспр k кц;
     ис := t
     конец
   быв;
(¬k пусто | назад (f));
если не конч
то (¬(испр ошибка значения из f) (f) |
```

```
        не определено);
        настроить на чтение (f)
         все
        кц
       конец
      быв кц
      иначе не определено
      все:
   проц (имя файл, [ ] об (вводимое, проц (имя файл пуст))
        пуст get = ввод:
b) проц ?ввести литеру = (имя файл f, имя лит лит) пуст:
      если открыт из f ∧ ¬строчка окончена (f) ∧
      ¬лог файл окончен (f)
      то имя позиция тпоз = тпоз из f:
       настроить на литерное (f);
       настроить на чтение (f);
       цел p = p из тпоз, l = l из тпоз, c = c из тпоз;
       c из тпоз +:= 1;
       лит := выб текст из f в
         (текст t1): t1 [p] [l] [c],
         (подвтекст t2):
          (лит k := t2 [p] [l] [c]:
          лог есть := ложь;
          для i до вегр F из код из f пока ¬ есть
          цк ст (лит внутр, внешн) табл =
            (F из код из f) [i];
            (внешн из табл = k | k := внутрь из табл:
            есть := истина)
          цк:
          если есть то k
          иначе k :="•";
             если (испр ошибка литеры из f) (f, k)
             то k
            иначе не определено: ". "
             все;
         настроить на чтение (f)
      все)
      быв
    иначе не определено
    все #настройка на чтение сохраняется#;
c) проц ?проверить позицию = (имя файл f) пуст:
      начало лог чтение = для чтения из f;
       лог не окончено := истина;
       пока не окончено := не окончено ∧
         страница хороша (f, чтение);
```

строчка окончена (f) ∧ не окончено
цк не окончено := (испр строчка из f) (f) кц
конец;

{ Процедура проверить позицию используется в форматном обмене перед каждым вызовом процедур вывести литеру и ввести литеру. Если позиция не хороша (10.3.1.6.dd), вызывается соответствующая процедура обработки события и, если вырабатывается истина, могут дальше вызываться процедуры обработки события. Если вырабатывается ложь, то в случае процедуры обработки события, соответствующей при конце страницы, вызывается нов страница, а для любой другой процедуры обработки события действия по умолчанию не предпринимаются и никакой процедуры обработки события больше не вызывается. По выходе сохраняется настройка на чтение/запись, которая была при входе, но текущая позиция может не быть хорошей. В этом случае не определено будет вызвано в следующей процедуре вывести литеру или ввести литеру. Однако проверить позицию вызывается также при вводе строк (hh), и в этом случае, если позиция не хороша, строка обрывается. }

10.3.4. *Тексты формата*

{ В форматном обмене каждое выстроенное значение из списка данных (ср. 10.3.3) сопоставляется с составляющим шаблоном некоторого текста-формата, обеспечиваемого пользователем. Шаблон задает, как преобразовать значение в или из последовательности литер, и предписывает расположение этих литер в книге. Возможности, которые можно задавать, включают: число цифр, позиции десятичной точки и знака, если они есть, подавление нулей и вставку произвольных строк. Например, с помощью шаблона

–d.3d "÷"3d"÷" e z + d

значение 1234.567 можно было бы обменивать, как строку

"÷ 1.234 ÷ 567 ÷ $_{10}$ ÷ + 3 ".

„Формат" – это структура (т.е. внутренний объект) вида 'ФОРМАТ', отражающая иерархическое строение текста-формата (являющегося внешним объектом). В данном разделе даны синтаксис текстов-формата и семантика для получения соответствующих им форматов. Фактический форматный обмен осуществляется посредством процедур, задаваемых в п. 10.3.5. Для удобства описание их операций дается здесь в связи с соответствующим синтаксисом.}

10.3.4.1. Наборы и шаблоны.

10.3.4.1.1. Синтаксис.

{ Следующие описания-вида (взятые из 10.3.5.а) отражены в приводимых ниже метаправилах от А до К.

A) вид формат = ст (подв [1 : 0] кадр F):

B) вид кадр = ст (цел утн, счет, оук, подв [1 : 0] набор н):

C) вид набор = об (шаблон, пакет):

D) вид пакет = ст (вставка в1, проц цел повт, цел удк,
вставка в2):

E) вид вставка = подв [1 : 0] ст (проц цел повт, об (строк, лит) стр):
F) вид шаблон = ст (об (трафарет, травыб, трафор, трабесф, пуст) траф, вставка в):
G) вид трафарет = ст (цел тип, подв [1 : 0] рамка рамки):
H) вид рамка = ст (вставка в, проц цел повт, лог подав, лит марк):
I) вид травыб = ст (вставка в, цел тип, подв [1 : 0] вставка стр):
J) вид трафор = ст (вставка в, проц формат прф):
K) вид трабесф = ст (вставка в, подв [1 : 0] проц цел спец):}

A) ФОРМАТ :: структура содержащая букву алеф для выборки вектора из КАДРОВ в себе.

B) КАДР : структура содержащая букву у букву т букву н для выборки целого букву с букву ч букву е букву т для выборки целого букву о букву у букву к для выборки целого букву н для выборки вектора из НАБОРОВ в себе.

C) НАБОР :: объединение ШАБЛОНА ПАКЕТА воедино.

D) ПАКЕТ :: структура содержащая букву в цифру один для выборки ВСТАВКИ букву п букву о букву в букву т для выборки процедуры вырабатывающей целое букву у букву д букву к для выборки целого букву в цифру два для выборки ВСТАВКИ в себе.

E) ВСТАВКА :: вектор из структур содержащих букву п букву о букву в букву т для выборки процедуры вырабатывающей целое букву с букву т букву р для выборки объединения вектора из литерных литерного воедино в себе.

F) ШАБЛОН :: структура содержащая букву т букву р букву а букву ф для выборки объединения ТРАФАРЕТА ТРАВЫБА ТРАФОРА ТРАБЕСФОРА пустого значения воедино букву в для выборки ВСТАВКИ в себе.

G) ТРАФАРЕТ :: структура содержащая букву т букву и букву п для выборки целого букву р букву а букву м букву к букву и для выборки вектора из РАМОК в себе.

H) РАМКА :: структура содержащая букву в для выборки ВСТАВКИ букву п букву о букву в букву т для выборки процедуры вырабатывающей целое букву п букву о букву д букву а букву в для выборки логического букву м букву а букву р букву к для выборки литерного в себе.

I) ТРАВЫБ :: структура содержащая букву в для выборки ВСТАВКИ букву т букву и букву п для выборки целого букву с букву т букву р для выборки вектора ВСТАВОК в себе.

J) ТРАФОР :: структура содержащая букву в для выборки ВСТАВКИ букву п букву р букву ф для выборки процедуры вырабатывающей ФОРПАТ в себе.

K) ТРАБЕСФ :: структура содержащая букву в для выборки ВСТАВКИ букву с букву п букву е букву ц для выборки вектора из процедур вырабатывающих целое в себе.

L) ФОРПАТ :: ци I определение структуры содержащей букву алеф

для выборки вектора из структур содержащих букву у
букву т букву н для выборки целого букву с букву ч букву е букву т
для выборки целого букву о букву у букву к для выборки
целого букву н для выборки вектора из объединения
структуры содержащей букву т букву р букву а букву ф
для выборки объединения ТРАФАРЕТА ТРАВЫБА
структуры содержащей букву в для выборки ВСТАВКИ
букву п букву р букву ф для выборки процедуры
вырабатывающей использование ци I в себе ТРАБЕСФОРА
пустого значения воедино букву в для выборки ВСТАВКИ
в себе ПАКЕТА
воедино
в себе
в себе.

{Вид 'ФОРПАТ' эквивалентен (2.1.1.2.a) виду 'ФОРМАТ'.}

M) ТОЧКА :: знак, точка; порядок; комплексное;
логическое.

N) ПУНКТ :: нуль; цифра; литера.

O) ?НЕПОДАВЛЯЕМОЕ :: неподавляемое; ПУСТО.

P) ТИП :: целое; вещественное; логическое; комплексное;
строковое; битовое; целого выбора; логического выбора;
форматное; бесформатное.

a) текст формата в СРЕДЕ выдающий ФОРМАТ (5D):
знак форматор ОФОРМЛЕННЫЙ {94f}.
список наборов в СРЕДЕ (b),
знак форматор ОФОРМЛЕННЫЙ {94f}.

b) набор в СРЕДЕ {a, b} :
возможная последовательность пояснений {92a},
шаблон в СРЕДЕ { c };
возможная последовательность пояснений {92a},
вставка в СРЕДЕ { d }, повторитель в СРЕДЕ { g },
упакованный кратким список наборов в СРЕДЕ { b },
возможная последовательность пояснений {92a},
вставка в СРЕДЕ { d }.

c) шаблон в СРЕДЕ { b }: возможный трафарет ТРАФАРЕТА в СРЕДЕ
{A342a, A343a, A344a, A345a, A346a, A347a, A348a, b,
A349a, A34Aa},
вставка в СРЕДЕ d .

d) вставка в СРЕДЕ { b, c, j, k, A347b, A348a, b, A349a, A34Aa}:
возможный литерал в СРЕДЕ {i},
возможная последовательность размещений в СРЕДЕ {e}.

e) размещение в СРЕДЕ {d} :
повторитель в СРЕДЕ { g }, код размещения {f},
возможный литерал в СРЕДЕ { i }.

f) код размещения {e}:

символ буква ка лат {94a} либо символ буква к {94a}:
символ буква икс лат {94a} либо символ буква х {94a}:
символ буква игрек лат {94a} либо символ буква у {94a}:
символ буква эль лат {94a} либо символ буква л {94a}:
символ буква пэ лат {94a} либо символ буква п {94a}:
символ буква ку лат {94a} либо символ буква ю {94a};

g) повторитель в СРЕДЕ {b, e, i, k}:
возможный неподавляемый повторитель в СРЕДЕ {h}.

h) неподавляемый повторитель в СРЕДЕ {g, i}:
натуральное число {811b}:
символ буква эн лат {94a} либо символ буква н {94a},
ЗАКРЫТОЕ предложение в СРЕДЕ раскрыто
выдающее целое {31a, 34a, -},
возможная последовательность пояснений {92a}.

i) ?НЕПОДАВЛЯЕМЫЙ литерал в СРЕДЕ {d, e, i, A348c}:
?НЕПОДАВЛЯЕМЫЙ повторитель в СРЕДЕ {g, h},
приведенное изображаемое в СРЕДЕ {80a} сильно
выдающее вектор из литерных {61a},
возможный неподавляемый литерал в СРЕДЕ {i}.

j) ?НЕПОДАВЛЯЕМАЯ рамка
ТОЧКИ в СРЕДЕ {A342c, A343b, c, A344a, A345a}:
вставка в СРЕДЕ {d},
?НЕПОДАВЛЯЕМОЕ подавление {1},
маркер ТОЧКИ {A342e, A343d, e, A344b, A345b}.

k) ?НЕПОДАВЛЯЕМАЯ рамка ПУНКТА в СРЕДЕ {A342b, c, A346a}:
вставка в СРЕДЕ {d}, повторитель в СРЕДЕ {g},
?НЕПОДАВЛЯЕМОЕ подавление {1},
маркер ПУНКТА {A342d, f, A346b}.

l) ?НЕПОДАВЛЯЕМОЕ подавление {j, k, A347b}:
если (?НЕПОДАВЛЯЕМОЕ) есть (неподавляемое), ПУСТО:
если (?НЕПОДАВЛЯЕМОЕ) есть (ПУСТО),
возможный символ буква эс лат {94a} либо возможный
символ буква ш {94a}.

m)* рамка: ?НЕПОДАВЛЯЕМАЯ рамка ТОЧКИ в СРЕДЕ {j}:
?НЕПОДАВЛЯЕМАЯ рамка ПУНКТА в СРЕДЕ {k}:
рамка ДВОИЧНОГО в СРЕДЕ {A347b}.

n)* маркер: маркер ТОЧКИ {A342e, A343d, e, A344b, A345b}:
маркер ПУНКТА {A342d, f, A346b}; маркер основания {A347c}.

o)* трафарет: трафарет ТИПА в СРЕДЕ {A342a, A343a, A344a, A345a, A346a, A347a, A348a, b, A349a, A34Aa}.
{Примеры:
a) Ф п „таблица" х 10а, л н (пред – 1) ("х = "12ж + д2х,
+ .12де + 2д3q" + j Х "3" ÷" si + .10де + 2дл) п ф
b) п „таблица" х10а · л н (пред – 1) ("х = "12ж + d2x,
+ .12де + 2д3q" + j Х "3" ÷" si + .10де + 2дл) п

c) l20кц ("север", "восточ", "юж", "запад") "ный"
d) п "таблица" х
e) п "таблица"
h) 10 · е (пред – 1)
i) "+ j X "3" "÷"
j) ши
k) "х = "12ж
l) ш }

{ Пояснения (9.2.1.a) могут встречаться в текстах-формата только в определенных позициях. В общем случае пояснения (как и всюду в языке) не могут встретиться между двумя символами-БУКВА или -ЦИФРА. }

aa) Для форматного вывода можно использовать процедуры ф вывод (10.3.5.1.a) и ф печ (или ф зап) (10.5.1.f), а для форматного ввода процедуры ф ввод (10.3.5.2.a) и ф чит (10.5.1.g). Каждый элемент в списке данных (ср. 10.3.3) является либо форматом, который должен присоединяться к данному файлу, либо значением, подлежащим обмену (таким образом, формат можно включить в список данных непосредственно перед значением, обмен которого использует этот формат).

bb) При вызове процедур ф вывод и ф ввод обмен происходит следующим образом:

Для каждого рассматриваемого поочередно элемента списка данных,
если это формат,
он становится текущим форматом файла посредством
процедуры присоединить формат (10.3.5.k),
а иначе этот элемент выстраивается (10.3.2.3.c) и
каждый элемент полученного массива выводится (hh) или вводится
(ii) с помощью очередного „шаблона" (cc, gg) из текущего формата.

cc) „Шаблон" есть выдача некоторого шаблона. Он состоит из „трафарета" какого-то конкретного 'ТИПА' (в соответствии с синтаксисом трафарета-ТИПА этого шаблона), за которым идет „вставка" (ee). За исключением трафаретов 'выбора', 'форматного' и 'бесформатного', трафареты состоят из „рамок", возможно „подавляемых", каждая из которых имеет вставку, „повторитель" (dd) и маркер, указывающий, что это "d" ("д"), "z" ("ж"); "i" ("и") и т.п. рамка. Рамки каждого трафарета могут группироваться в „образцы знака", „образцы целого" и т.п. в соответствии с синтаксисом соответствующего трафарета.

dd) „Повторитель" есть процедура, вырабатывающая целое число и конструируемая из повторителя (10.3.4.1.2.c). Например, повторитель 10 создает процедуру, состоящую из цел: 10; более того, n (предел – 1) является „динамическим" повторителем и создает процедуру цел: (предел – 1). Отметим, что область действия повторителя ограничивает область действия любого содержащего его формата, и потому может оказаться необходимым взять локальную копию файла, прежде чем присоединять к нему формат. Повторитель, вырабатывающий отрицательное значение, рассматривается (за исключением "k" ("к") размещения) как вырабатывающий значение нуль.

Когда шаблон „подготавливается", все его повторители и прочие процедуры (включая и те, что содержатся в его вставках) вызываются совместно. Можно сказать, что подготовленный трафарет „управляет" строкой, поскольку существует соответствие между рамками этого трафарета и литерами этой строки. Каждая рамка управляет n последовательными литерами данной строки, где n для "r" ("я") рамки равно 0, а иначе n – целое число, вырабатываемое повторителем рамки (которое всегда равно 1 для "+", "–", " . ", "e", "i" ("и") или "b" ("б") рамки). Каждая из управляемых литер должна принадлежать определяемому соответствующей рамкой ограниченному множеству.

ee) „Вставка", являющаяся выдачей вставки (10.3.4.1.2.d), есть последовательность повторяемых „размещений" и строк; вставка, не содержащая размещений, называется „литералом". Вставка „осуществляется" посредством осуществления ее размещений (ff) и при выводе (вводе) записью („ожиданием") (ll)) каждой литеры из ее повторяемых строк (строка повторяется посредством воспроизведения ее столько раз, каково вырабатываемое ее повторителем число).

ff) „Размещение" есть литера, выдаваемая кодом-размещения (10.3.4.1.2.d). Повторяемое n раз размещение осуществляется следующим образом:

- "k" ("к") заставляет вызваться уст номер литеры с n в качестве ее второго параметра;
- "x" ("х") заставляет n раз вызваться вперед;
- "y" ("у") заставляет n раз вызваться назад;
- "l" ("л") заставляет n раз вызваться нов строчка;
- "p" ("п") заставляет n раз вызваться нов страница;
- "q" ("ю") при выводе (вводе) заставляет n раз записать (ожидать) литеру пробел.

gg) Формат может состоять из последовательности шаблонов, каждый из которых выбирается по очереди посредством взять след шаблон (10.3.5.b). В дополнение к этому некоторое множество шаблонов можно сгруппировать вместе и образовать повторяемый „набор" (который сам может содержать подобные наборы). Когда выбирается последний шаблон набора, снова выбирается его первый шаблон и т.д., пока весь этот набор не повторится n раз, где n – целое число, вырабатываемое его повторителем. Набор можно снабдить двумя вставками, первая из которых осуществляется перед набором, а вторая после него.

Формат может также вызывать другие форматы с помощью трафаретов 'форматного ' (10.3.4.9.1).

Когда формат исчерпывается, вызывается процедура обработки события, соответствующая при конце формата; если она вырабатывает ложь, то данный формат повторяется, а иначе, если только эта процедура обработки события не может обеспечить новый формат, вызывается не определено.

hh) Значение V выводится с помощью шаблона P следующим образом:

Если трафарет Q этого P есть трафарет 'выбора' или 'бесформатного',

то V выводится с помощью P (см. 10.3.4.8.1.aa, dd, 10.3.4.1.aa),
а иначе V выводится так:
• P подготавливается.
Если вид этого V „совместим по выводу" с Q,
то
• V преобразуется в строку, управляемую (dd) этим Q.
Если данный вид не совместим по выводу или если это преобразование не было успешным,
то
• вызывается процедура обработки события, соответствующая при ошибке значения:
• если она вырабатывает ложь, V выводится с помощью процедуры вывод и вызывается не определено;
а иначе данная строка „редактируется" (jj) с помощью трафарета Q;
• осуществляется вставка из P.

ii) Значение вводится в имя N с помощью шаблона P следующим образом:
Если трафарет Q этого P есть трафарет 'выбора' или 'бесформатного',
то значение вводится в N с помощью P (см. 10.3.4.8.1.bb, ee, 10.3.4.10.1.bb);
а иначе
• P подготавливается:
• „составляется" строка, управляемая Q (kk).
Если вид этого N „совместим по вводу" с Q,
то
• данная строка преобразуется с помощью Q в подходящее для N значение;
• если это преобразование было успешным, данное значение присваивается N.
Если данный вид не совместим по вводу или если это преобразование не было успешным,
то
• вызывается процедура обработки события, соответствующая при ошибке значения;
• если она вырабатывает ложь, вызывается не определено;
• осуществляется вставка из P.

jj) Строка „редактируется" с помощью трафарета P следующим образом:
В каждой части строки, управляемой образцом знака,
• если (указывающая знак) первая литера этой строки есть "+", а данный образец знака содержит рамку "–", то эта литера заменяется на "÷"
• первая литера (т.е. знак) сдвигается вправо через все начальные нули в данной части этой строки и эти нули заменяются пробелами (например, с помощью образца знака 4z + (4ж +) строка

"+0003" становится строкой "␣␣␣+3").

В каждой части этой строки, управляемой образцом целого,

- управляемые "z" ("ж") рамками нули заменяются пробелами, если они расположены
  - между началом данной строки и первой ненулевой цифрой;
  - между каждой "d" ("д") "e" ("е") или "i" ("и") рамкой и следующей ненулевой цифрой;

(например, с помощью трафарета zdzd2d (ждждд) строка "180168" становится "18␣168").

Для каждой рамки F из P

- осуществляется вставка из F;
- если рамка F не подавляема, записываются управляемые ею литеры;

(например, редактирование с помощью трафарета 4ж + ш. " , " д строки "+0003.5" дает "␣␣␣+3,5", а редактирование строки "180168" с помощью трафарета жд"–"жд"–19"2д дает "18–␣1–1968").

kk) Строка „составляется" с помощью трафарета P следующим образом:

Для каждой рамки F из P

- осуществляется вставка из F.

Для каждого элемента этой строки, управляемого F, следующим образом получается некоторая литера:

Если F содержится в образце знака,

то

- если знак уже был, ожидается цифра с предлагаемым "0";
- а иначе ожидается "+" или "–" с предлагаемым "+" и, кроме того, если данный образец знака содержит "–" рамку, в качестве знака будет приемлем пробел; предшествующий первой цифре (заменяемый на "+");

а иначе если рамка F содержится в образце целого, то

если она подавляема,

то подается "0";

а иначе:

Случай A: F – "d" ("д") рамка:

- ожидается некоторая цифра с предлагаемым "0";

Случай B: F – "z" ("ж") рамка:

- ожидается цифра или пробел с предлагаемым "0", причем пробел приемлем только в следующих случаях:
- между началом данной строки и первой ненулевой цифрой;
- между каждой "d" ("д"), "e" ("е") или "i" ("и") рамкой и следующей ненулевой цифрой;
- такие пробелы заменяются нулями;

а иначе если F – "a" ("а") рамка,

то, если она неподавляема, читается и подается некоторая литера, а иначе подается "␣";

иначе, если F неподавляема,

то, если F – " . " ("е ("е"), "i" ("и"), "b" ("б")) рамка, ожидается " . " ("$_{10}$" или "\" или "е", "⊥" или "i", да или нет) с предлагаемым " . " ("$_{10}$", "⊥", нет):

а иначе, если F – подавляема " . " ("е" ("е"), "i" ("и")) рамка, подается литера " . " ("$_{10}$", " ⊥").

ll) Элемент множества литер S „ожидается" с предлагаемой литерой С следующим образом:

- читается некоторая литера:

если это одна из ожидаемых литер (т.е. принадлежит множеству s),

то она подается,

а иначе вызывается процедура обработки события, соответствующая при ошибке литеры, с предлагаемой литерой С: если эта процедура вырабатывает истина, а (возможно, измененная) литера С есть одна из ожидаемых литер, то подается С, а иначе вызывается не определено. }

10.3.4.1.2. Семантика.

{ Форматы вводятся в действие посредством текстов-формата. Формат лучше всего рассматривать как дерево с некоторым набором в каждом узле и шаблоном на каждом из его концов. Чтобы не нарушались ограничения на области действия, каждый узел в этом дереве запакован – в настоящем стандарте – в значение вида 'КАДР'. Формат составлен вектором таких кадров, и они содержат указатели друг на друга в форме индексов, выбирающих (их) из этого вектора. Без сомнения, в реализациях это дерево будет храниться более эффективным способом. Это возможно, так как указатель-поля любого формата скрыт от пользователя, чтобы тот не мог проникнуть в это поле.

Хотя текст-формата может содержать ЗАКРЫТЫЕ-предложения (в повторителях и трафаретах-форматного) или основы (в трафаретах-бесформатного), эти ЗАКРЫТЫЕ-предложения и основы не исполняются при исполнении текста-формата, но превращаются в процедуры, вызываемые впоследствии, когда до них дойдет дело в ходе форматного обмена. В действительности исполнение текста-формата не приводит ни к каким действиям, имеющим какой-нибудь смысл для пользователя. }

а) Выдача текста-формата F в окружении Е есть структура, единственным полем которой является массив W вида 'вектор из КАДРОВ'. Этот массив составлен из паспорта ((1, n)) и n элементов и определяется следующим образом:

- счетчик i устанавливается в 1;
- W получается „трансформацией" {b} F в окружении Е с помощью i.

b) Массив W, вид которого есть 'вектор из КАДРОВ', получается „трансформацией" текста-формата или упакованного-списка-наборов С в окружении Е с помощью счетчика i следующим образом:

- элемент этого W, выбираемый по (i), является структурой, вид которой есть 'КАДР' и поля которой, взятые в их порядке, таковы
  - { утн } не определено;

• {счет} не определено;
• {оук} не определено;
• {н} массив V, вид которого есть 'вектор из НАБОРОВ' с паспортом ((1, m)), где m – число составляющих наборов из C, и элементами, определяемыми так:

Для j = 1, ..., m пусть C_j будет j-м составляющим набором из C:

Случай A: Прямой наследник этого C_j включает шаблон P:

• составляющий трафарет T, если он есть, и вставка I этого P исполняются совместно;
• j-м элементом V является структура, вид которой есть 'ШАБЛОН' и поля которой, взятые в их порядке, таковы:
• {траф} выдача этого T, если она есть, {e, 10.3.4.8.2, 10.3.4.9.2, 10.3.4.10.2}, а иначе пусто;
• в выдача этой I {d};

Случай B: Прямой наследник этого C_j включает первую вставку I1, повторитель REP, упакованный-список-наборов P и вторую вставку I2:

• i увеличивается на 1;
• I1, REP и I2 исполняются совместно;
• j-м элементом V является структура, вид которой есть 'ПАКЕТ' и поля которой, взятые в их порядке, таковы
• {в1} выдача I1 {d};
• {повт} выдача REP {c};
• {удк} i;
• {в2}выдача I2;
• W получается трансформацией P в окружении E с помощью i.

c) Выдача в некотором окружении E ?НЕПОДАВЛЯЕМОГО-повторителя-в-СРЕДЕ R {10.3.4.1.1.g, h} – это процедура вида 'процедура вырабатывающая целое', составленная из текста-процедуры-в-СРЕДЕ-выдающего-процедуру-вырабатывающую-целое, основа которого есть U вместе с окружением, необходимым {7.2.2.c} для U в E, причем U определяется следующим образом:

Случай A: R содержит ЗАКРЫТОЕ-*предложение-раскрыто-выдающее-целое* C:

• U – новая основа, подобная {1.1.3.2.k} C;

Случай B: R содержит натуральное-число D, а не ЗАКРЫТОЕ-предложение:

• U – новая основа, подобная D;

Случай C: R невидимо:

• U – новая основа, подобная натуральному-числу, имеющему естественное {8.1.1.2} значение 1.

d) Выдача вставки I {10.3.4.1.1.d} – это массив W вида 'ВСТАВКА', определяемый следующим образом:

• пусть $U_1, ..., U_n$ – составляющие ?НЕПОДАВЛЯЕМЫЕ-повторители вставки I и A_i для i = 1, ..., n есть приведенное-изображаемое или код-раз-

мещения, непосредственно следующий за U_i;

· пусть R_1, ..., R_n и D_1, ..., D_n – совместные выдачи U_1, ..., U_n и A_1, ..., A_n, причем выдача кода-размещения символ-буква-ка-лат или символ-буква-к-(символ-буква-икс-лат или символ-буква-х, символ-буква игрек-лат или символ-буква-у, символ-буква-эль-лат или символ-буква-л, символ-буква-пэ-лат или символ-буква-п, символ-буква-ку-лат или символ-буква-ю) есть {литера, которая есть} естественное значение {8.1.4.2b} символа-буква-ка-лат (символа-буква-икс-лат, символа-буква-игрек-лат, символа-буква-эль-лат, символа-буква-пэ-лат, символа-буква-ку-лат);

· паспортом этого W является ((1, n)):

· элемент этого W, выбираемый по (i), i = 1, ..., n, является структурой, вид которой специфицируется посредством ст (проц цел повт, об (строк, лит) стр) и поля которой, взятые в их порядке, таковы:

· {повт} R_i;

· {стр} D_i.

е) Выдача трафарета-целого, -вещественного, -логического, комплексного, -строкового или -битового P {10.3.4.2.1.a,} 10.3.4.3.1.a, ..., 10.3.4.7.1.a} – это структура W вида 'ТРАФАРЕТ', определяемая следующим образом:

· пусть V_1, ..., V_n – совместные выдачи составляющих рамок из P {f}:

· поля структуры W, взятые в их порядке таковы:

· {тип} i (2, 3, 4, 5), если P – трафарет-целого (-вещественного, -логического, -комплексного, -строкового), и 6 (8, 12, 20), если P – трафарет-битового, составляющее ДВОИЧНОЕ-основание которого есть двоичное-(четверичное-, восьмеричное-, шестнадцатеричное-) основание;

· {рамки} массив, вид которого есть 'вектор из РАМОК', имеющий паспорт ((1, n) и n элементов, причем выбираемым по (i) элементом будет V_i.

f) Выдача всякой рамки F {10.3.4.1.1.m} – это структура W вида 'РАМКА' определяемая следующим образом:

· вставка и повторитель, если они есть, из F исполняются совместно;

· поля структуры W, взятые в их порядке, таковы:

· {в} выдача ее вставки;

· {повт} выдача ее повторителя {с}, если он есть, а иначе, выдача невидимого повторителя;

· {подав} истина, если ее ?НЕПОДАВЛЯЕМОЕ-подавление содержит символ-букву-эс-лат (символ-букву-ш), и ложь в противном случае;

· {марк}{литера, которая есть} естественное значение {8.1.4.2.b} символа S, определяемого следующим образом:

Случай А: F – составляющая неподавляемая-рамка-нуля некоторого образца-знака {, такого, как 3ж+}, составляющий маркер-знака которого содержит символ-плюс:

• S есть символ-буква-у-лат;

Случай B: F – составляющая неподавляемая-рамка-нуля некоторого образца-знака { , такого, как 3ж– }, составляющий маркер-знака которого содержит символ-минус:

• S есть символ-буква-ве-лат;

Прочие случаи: составляющий символ маркера из F есть символ-буква-а-лат или символ-буква-а (символ-буква-бе-лат или символ-буква-б, символ буква-де лат или символ-буква-д, символ-буква-е-лат или символ-буква-е, символ-буква-и-лат или символ-буква-и, символ-буква-эр-лат или символ-буква-я, символ-буква-зэт-лат или символ-буква-ж):

• S есть символ-буква-а-лат (символ-буква-бе-лат, символ-буква-де-лат, символ-буква-е-лат, символ-буква-и-лат, символ-буква-эр-лат, символ-буква-зэт-лат).

{Таким образом, маркер-нуля ж можно передать как литеру "u", "v" или "z" в зависимости от того, образует он часть образца-знака (с наследным символом-плюс или символом-минус) или же часть образца-целого. Кроме того, русские символы маркера заменяются их латинскими эквивалентами.

10.3.4.2. Трафареты целого.

10.3.4.2.1. Синтаксис.

a) трафарет целого в СРЕДЕ {A341c, A343c}:
возможный образец знака в СРЕДЕ {c},
образец целого в СРЕДЕ {b}.

b) образец целого в СРЕДЕ {a, A343b, c, A347a};
последовательность рамок цифры в СРЕДЕ {A341k}.

c) образец знака в СРЕДЕ {a, A343a}:
возможная последовательность неподавляемых
рамок нуля в СРЕДЕ {A341k},
неподавляемая рамка знака в СРЕДЕ {A341j}.

d) маркер нуля {f, A341k}: символ буква зэт лат 94a
либо символ буква ж {94a}.

e) маркер знака {A341j}:
символ плюс {94c}; символ минус {94c}.

f) маркер цифры {A341k}:
символ буква де лат {94a} либо символ буква д {94a};
маркер нуля {d}.

{Примеры:
a) "x = "12ж + д b) ∂
c) "x = "12ж + }

{О семантике трафаретов-целого см. 10.3.4.1.2.e.}

{aa) Совместные по выводу (вводу) с трафаретами целого виды специфицируются посредством Д цел (имя Д цел).

bb) Значение V преобразуется в строку S с помощью трафарета 'целого' P следующим образом:

• если P содержит образец знака, то первой литерой S будет знак этого V, а иначе, если $V < 0$, преобразование не будет успешным;

оставшаяся часть S заполняется десятичным представлением этого V, получаемым следующим образом:

элементами строки S, управляемыми "d" ("д") и "z" ("ж") рамками, будут соответствующие цифры (таким образом, шаблон определяет число используемых цифр);

• если V нельзя представить такой строкой, то данное преобразование не будет успешным.

(Например, используя трафарет zzd (жжд), значение 99 можно преобразовать в строку, а значения 9999 и – 99 нельзя).

cc) Строка S преобразуется в целое число, подходящее для имени N, с помощью трафарета 'целого' следующим образом:

• рассматривается целое число l, десятичное представление (8.1.1.2) которого содержится в S;

• если l превышает наибольшее значение, которое может именовать N, то преобразование не будет успешным; иначе l – требуемое целое число (например, если вид этого N специфицируется посредством имя кор цел, а значением кор макс цел служит 65535, то никакую строку, содержащую десятичное представление значения, превышающего 65535, преобразовать нельзя).}

10.3.4.3. Трафареты вещественного.

10.3.4.3.1. Синтаксис.

a) трафарет вещественного в СРЕДЕ {A341c, A345a}:
возможный образец знака в СРЕДЕ {A342c},
образец рационального в СРЕДЕ {b} либо
образец действительного в СРЕДЕ {c}.

b) образец рационального в СРЕДЕ {a, c}:
образец целого в СРЕДЕ {A342b},
рамка точки в СРЕДЕ {A341j},
возможный образец целого в СРЕДЕ {A342b};
рамка точки в СРЕДЕ {A341j},
образец целого в СРЕДЕ {A342b}.

c) образец действительного в СРЕДЕ {a}:
образец рационального в СРЕДЕ {b} либо
образец целого в СРЕДЕ {A342b},
рамка порядка в СРЕДЕ {A341j},
образец целого в СРЕДЕ {A342a}.

d) маркер точки {A341j}: символ точка {94b}.

e) маркер порядка {A341j}:
символ буква e лат {94a} либо символ буква е {94a}.

{Примеры:

a) + жд.11д • +.12де + 2д b) жд.11д • .12д

c) .12де + 2д}

{О семантике трафаретов-вещественного см. 10.3.4.1.2.e.}

{aa) Виды, совместимые по выводу (вводу) с трафаретами 'вещественного', специфицируются посредством Д вещ и Д цел (имя Д вещ).

bb) Значение V преобразуется в строку S с помощью трафарета 'вещественного' P следующим образом:

- если P содержит образец знака, то первой литерой строки S будет знак этого V, а иначе, если $V < 0$, преобразование не будет успешным;
- оставшаяся часть S заполняется десятичным представлением этого V, определенным следующим образом:
  - если необходимо, V обобщается до вещественного числа;
  - элементом строки S, управляемым " . " ("e" или "е") рамкой из P, если он есть, будет " . " ("$_{10}$");

    Если P содержит "e" ("е") рамку,

    то
    - пусть W – последовательность рамок, предшествующих этой "e" ("е") рамке, а lP – следующий за ней трафарет 'целого';
    - порядок E вычисляется посредством нормализации этого V до наибольшего значения, преобразуемого с помощью W (см. ниже);
    - управляемая lP часть строки S получается преобразованием порядка E с помощью lP (см. 10.3.4.2.1.bb);

    а иначе
    - P целиком становится последовательностью W;
- элементами строки S, управляемыми "d" ("д") и "z" ("ж") рамками из W, являются соответствующие цифры (таким образом, данный шаблон определяет число используемых цифр, а также число цифр, помещаемых после десятичной точки, если она есть);
- если V нельзя представить такой строкой, то данное преобразование не будет успешным.

cc) Строка S преобразуется в вещественное число, подходящее для имени N, с помощью трафарета 'вещественного' следующим образом:

- рассматривается вещественное число R, десятичное представление которого содержится в S;
- если R превышает наибольшее значение, которое может именовать N, то преобразование не будет успешным; иначе R – требуемое вещественное число.}

10.3.4.4. Трафареты логического.

10.3.4.4.1. Синтаксис.

a) трафарет логического в СРЕДЕ {A341c}:

неподавляемая рамка логического в СРЕДЕ {A341j}.

b) маркер логического {A341j, A348в}:

символ буква бе лат {94a} либо символ буква б {94a}.

{Пример:

а) 14хб

{О семантике трафаретов-логического см. 10.3.4.1.2.e.}

{aa) Вид, совместимый по выводу (вводу) с трафаретами 'логического', специфицируется посредством лог (имя лог).

bb) Значение V преобразуется в строку с помощью трафарета 'логического' следующим образом:

- если V – истина (ложь), то это строка, выдаваемая литерой да (нет).

cc) Строка S преобразуется в логическое значение с помощью трафарета 'логического' следующим образом:

- если S совпадает со строкой, выдаваемой да (нет),

то требуемым значением будет истина (ложь).}

10.3.4.5. Трафареты комплексного.

10.3.4.5.1. Синтаксис.

a) трафарет комплексного в СРЕДЕ {A341c}:
трафарет вещественного в СРЕДЕ {A343a},
рамка комплексного в СРЕДЕ {A341j},
трафарет вещественного в СРЕДЕ {A343a}.

b) маркер комплексного {A341j}:
символ буква и лат {94a} либо символ буква и {94a}.

{Пример:

а) +.12де + 2д3ю "+ j X" 3"-" ши + .10де + 2д}

{О семантике трафаретов-комплексного см. 10.3.4.1.2.e.}

{aa) Виды, совместимые по выводу (вводу) с трафаретами 'комплексного', специфицируются посредством Д компл, Д вещ и Д цел (имя Д компл).

bb) Значение V преобразуется в строку S с помощью трафарета 'комплексного' P следующим образом:

- если необходимо, V обобщается до комплексного числа;
- элементом строки S, управляемым " i" ("и"), рамкой из P, является "⊥";
- управляемая первым (вторым) трафаретом 'вещественного' этого P часть строки S получается преобразованием первого (второго) поля из V в строку с помощью первого (второго) трафарета 'вещественного' этого P (10.3.4.1.bb);
- если какое-то из этих преобразований не было успешным, то и преобразование этого V не будет успешным.

cc) Строка преобразуется в комплексное значение C, подходящее для имени N, с помощью трафарета 'комплексного' P следующим образом:

- часть строки, управляемая первым (вторым) трафаретом 'вещественного' этого P, преобразуется в подходящее вещественное число (10.3.4.1.cc), давая первое (второе) поле этого C;
- если какое-то из этих преобразований не было успешным, то и преобразование этого C не будет успешным.}

10.3.4.6. Трафареты строкового.

10.3.4.6.1. Синтаксис.

a) трафарет строкового в СРЕДЕ {A341c}:
последовательность рамок литеры в СРЕДЕ {A341k}.

b) маркер литеры {A341k}: символ буква а лат {94a}
либо символ буква а {94a}.

{Пример:

а) n "таблица" x10a}

{О семантике трафаретов-строкового см. 10.3.4.1.2.e.}

{aa) Виды, совместимые по выводу (вводу) с трафаретами 'строкового', специфицируются посредством лит и [ ] лит (имя лит, имя [ ] лит и имя строк).

bb) Значение V преобразуется в строку с помощью трафарета 'строкового' P следующим образом:

- если необходимо, V векторизуется в строку;
- если длина этой строки V равна длине строки, управляемой P, то подается V, а иначе преобразование не будет успешным.

cc) Строка S преобразуется в литеру или строку, подходящую для имени N, с помощью трафарета 'строкового' следующим образом:

Случай A: Вид этого N специфицируется посредством имя лит:

- если S не состоит из одной литеры, то преобразование не будет успешным; иначе подается эта литера;

Случай B: Вид этого N специфицируется посредством имя [ ] лит:

- если длина строки S не равна числу литер, именуемых именем N, то преобразование не будет успешным; а иначе подается S;

Случай C: Вид этого N специфицируется посредством имя строк:

- подается S.}

10.3.4.7. Трафареты битового.

10.3.4.7.1. Синтаксис.

a) трафарет битового в СРЕДЕ {A341c}:
рамка ДВОИЧНОГО основания в СРЕДЕ {b},
образец целого в СРЕДЕ {A342b}.

b) рамка ДВОИЧНОГО основания в СРЕДЕ {a}:
вставка в СРЕДЕ {A341d},
ДВОИЧНОЕ основание {82d, e, f, g},
неподавляемое подавление {A3411},
маркер основания {c}:

c) маркер основания {b}:
символ буква эр лат {94a} либо символ буква я {94a}.

{Примеры:

a) 2r6d26sd · 2я6д26сд b) 2r · 8я}

{О семантике трафаретов-битового см. 10.3.4.1.2.e.}

{aa) Виды, совместимые по выводу (вводу) с трафаретами 'битового', специфицируются посредством Д бит (имя Д бит).

bb) Значение V преобразуется в строку с помощью трафарета 'битового' P следующим образом:

- с помощью обозначения-операции абс (10.2.3.8.i) определяется целое число l, соответствующее V;

Если "r" ("я") рамка из P выдана рамкой-двоичного- (-четверичного-, -восьмеричного-, -шестнадцатеричного-) -основания, то l преобразуется

в управляемую образцом целого из P строку, содержащую двоичное (четверичное, восьмеричное, шестнадцатеричное) представление этого l (ср. 10.3.4.2.1.bb),

• если l нельзя представить такой строкой, то преобразование не будет успешным.

cc) Строка S преобразуется в битовое значение, подходящее для имени N, с помощью трафарета 'битового' P следующим образом:

• если "г" ("я") рамка из P выдана рамкой-двоичного-(-четверичного-, восьмеричного-, -шестнадцатеричного-) -основания, то определяется целое число l, для которого S содержит двоичное (четверичное, восьмеричное, шестнадцатеричное) представление;

• с помощью обозначения-операции бин (10.2.3.8.j) определяется битовое значение B, соответствующее этому l;

• если размер этого B больше размера значения, именуемого N, то данное преобразование не будет успешным.}

10.3.4.8. Трафареты выбора.

10.3.4.8.1. Синтаксис.

a) трафарет целого выбора в СРЕДЕ {A341c}:
вставка в СРЕДЕ {A341d}, символ буква цэ лат {94a} либо
символ буква ц {94a},
упакованный кратким список
поясняемых литералов в СРЕДЕ {c},
возможная последовательность пояснений {92a}.

b) трафарет логического выбора в СРЕДЕ {A341c}:
вставка в СРЕДЕ {A341d}, маркер логического {A344b},
знак начало краткий {94f},
поясняемый литерал в СРЕДЕ {c},
знак а также {94f},
поясняемый литерал в СРЕДЕ {c},
знак конец краткий {94f},
возможная последовательность пояснений {92a}.

c) поясняемый литерал в СРЕДЕ {a, b}:
возможная последовательность пояснений {92a},
литерал в СРЕДЕ {A341i}.

{Примеры:
a) л 20k ц (""севeр", ""восточ", ""юж", ""запад")
b) б ("", " ошибка")
c) ""север" }

{aa) Значение V выводится с помощью шаблона P, трафарет которого Q был выдан трафаретом-целого-выбора C, следующим образом:

• подготавливается (10.3.4.1.1.dd) и осуществляется (10.3.4.1.1.ee) вставка из Q;

Если вид значения V специфицируется посредством цел, а $V > 0$,
и если число составляющих литералов в упакованном-списке-поясняемых-литералов этого C не меньше V,

то

- подготавливается и осуществляется литерал, выдаваемый V-м из этих литералов;

а иначе

- вызывается процедура обработки события, соответствующая при ошибке значения;
- если она вырабатывает ложь, V выводится с помощью вывод и вызывается не определено;
- подготавливается и осуществляется вставка из P.

bb) Значение вводится в имя N с помощью шаблона P, трафарет которого Q был выдан трафаретом-целого-выбора C, следующим образом:

- подготавливается и осуществляется вставка из Q;
- по очереди подготавливается и „отыскивается" (cc) каждый из литералов, выдаваемых составляющими литералами упакованного-списка-поясняемых-литералов этого C;

Если вид этого N специфицируется посредством имя цел и i-й литерал окажется первым из искомых,
то i присваивается N;
а иначе

- вызывается процедура обработки события, соответствующая при ошибке значения;
- если она вырабатывает ложь, вызывается не определено;
- подготавливается и осуществляется вставка из P.

cc) Литерал „отыскивается" путем чтения литер и сопоставления их с последовательными литерами этого литерала. Если будет достигнут конец текущей строчки либо логического файла или для некоторой литеры не найдется соответствия, то этот поиск не будет успешным, а текущая позиция возвращается на то место, откуда поиск начался.

dd) Значение V выводится с помощью шаблона P, трафарет Q которого был выдан трафаретом-логического-выбора C, следующим образом:

- подготавливается и осуществляется вставка из Q;

Если вид этого V специфицируется посредством лог,
то

- если V – истина (ложь), подготавливается и осуществляется литерал, выдаваемый первым (вторым) составляющим литералом из C;

а иначе

- вызывается процедура обработки события, соответствующая при ошибке значения;
- если она вырабатывает ложь, V выводится с помощью вывод и вызывается не определено;
- подготавливается и осуществляется вставка из P.

ee) Значение вводится в имя N с помощью шаблона P, трафарет Q которого был выдан трафаретом-логического-выбора C, следующим образом:

- подготавливается и осуществляется вставка из Q;
- по очереди подготавливается и отыскивается каждый из литералов, вы-

даваемых составляющими литералами из С;

Если вид этого N специфицируется посредством имя лог и первый (второй) литерал окажется искомым,

то этому N присваивается истина (ложь);

а иначе

- вызывается процедура обработки события, соответствующая при ошибке значения;
- если она вырабатывает ложь, вызывается не определено;
- подготавливается и осуществляется вставка из Р. }

10.3.4.8.2. Семантика.

Выдача трафарета-выбора Р является структурой W вида 'ТРАВЫБ', определяемой следующим образом:

- пусть n – число составляющих литералов-в-СРЕДЕ упакованного-списка-поясняемых-литералов из Р;
- пусть S_i, i = 1, ..., n, есть вставка-в-СРЕДЕ, подобная {1.1.3.2.k} i-му из этих составляющих литералов-в-СРЕДЕ;
- вставка l из Р и все $S_1, S_2, ..., S_n$ исполняются совместно;
- поля структуры W, взятые в их порядке, таковы:
  - { в } выдача вставки l;
  - { тип } 1 (2), если Р – трафарет-логического-(-целого-)-выбора;
  - { стр } массив вида 'вектор из ВСТАВОК', имеющий паспорт ((1, n)) и n элементов, причем выбираемый по (i), i = 1, ..., n, элемент – это выдача вставки S_i.

10.3.4.9. Трафареты форматного.

10.3.4.9.1. Синтаксис.

а) трафарет форматного в СРЕДЕ {А341с}:
вставка в СРЕДЕ {А341d},
символ буква эф лат {94а} либо символ буква ф {94а},
ЗАКРЫТОЕ предложение в СРЕДЕ
раскрыто выдающее ФОРМАТ {31а, 34а},
возможная последовательность пояснений {92а}.
{Пример:
а) ф (uir | (цел) : ф 5дф, (вещ) : фд.3дф)}

{Трафарет-форматного можно использовать, чтобы динамически создавать форматы для применения в обмене. Когда в течение вызова процедуры взять след шаблон дело доходит до трафарета 'форматного', он подготавливается, а затем осуществляется его вставка. Первый шаблон формата, вырабатываемого процедурой этого трафарета, подается в качестве очередного шаблона, а дальше шаблоны берутся из этого формата до тех пор, пока он не исчерпается.}

10.3.4.9.2. Семантика.

Выдача в окружении Е некоторого трафарета-форматного Р – это структура, вид которой есть 'ТРАФОР', и поля которой, взятые в их порядке, таковы:

- { в } выдача его вставки;
- { прф } процедура вида 'процедура вырабатывающая ФОРМАТ', со-

ставленная из текста-процедуры-в-СРЕДЕ-выдающего-процедуру-вырабатывающую-ФОРМАТ, основа U которого является новой основой, подобной {1.1.3.2.k} ЗАКРЫТОМУ-предложению-раскрыто-выдающему-ФОРМАТ этого P, вместе с окружением, необходимым для U в E.

10.3.4.10. Трафареты бесформатного.

10.3.4.10.1. Синтаксис.

a) трафарет бесформатного в СРЕДЕ {A341c}:
вставка в СРЕДЕ {A341d},
символ буква ге лат {94a} *либо символ буква г* {94a},
возможное задание разрядности в СРЕДЕ {b}.

b) задание разрядности в СРЕДЕ {a}:
знак начало краткий {94f},
основа в СРЕДЕ раскрыто выдающая целое {32d}.
возможное задание после {c},
знак конец краткий {94f},
возможная последовательность пояснений {92a}.

c) задание после в СРЕДЕ {b}:
знак а также {94f},
основа в СРЕДЕ раскрыто выдающая целое {32d},
возможное задание порядка в СРЕДЕ {d}.

d) задание порядка в СРЕДЕ {c}:
знак а также {94f},
основа в СРЕДЕ раскрыто выдающая целое {32d}.

{Примеры:

a) г • г (–18, 12, –3) b) –18, 12, –3

c) , 12, –3 d) , –3}

{aa) Значение V выводится с помощью шаблона P, трафарет Q которого был выдан трафаретом-бесформатного G, следующим образом:

- P подготавливается;
- осуществляется вставка из Q;

Если Q не параметризован (т.е. не содержит задания-разрядности), то V выводится с помощью вывод;

а иначе, если вид этого V специфицируется посредством Д цел или Д вещ,

то

- если Q содержит один (два, три) параметр (а, ов), V преобразуется в строку с помощью целое (фикс, плав);
- эта строка записывается с помощью вывод;

а иначе

- вызывается процедура обработки события, соответствующая при ошибке значения;
- если она вырабатывает ложь, V выводится с помощью вывод и вызывается не определено;
- осуществляется вставка из P.

bb) Значение вводится в имя N с помощью шаблона P, трафарет кото-

рого есть трафарет 'бесформатного', следующим образом:

- P подготавливается;
- осуществляется вставка из Q;
- (игнорируются любые параметры и) данное значение вводится в N с помощью ввод;
- осуществляется вставка из P.}

10.3.4.10.2. Семантика.

Выдача в окружении E некоторого трафарета-бесформатного-в-СРЕДЕ P – это структура вида 'ТРАБЕСФ', поля которой, взятые в их порядке, таковы

- {в} выдача вставки из P;
- {спец} массив W вида 'вектор из процедур вырабатывающих целое', имеющий паспорт ((1, n)), где n – число составляющих основ-раскрыто-выдающих-целое возможного-задания-разрядности из P, и n элементов, определяемых следующим образом:

Для i = 1, ..., n

- i-м элементом массива W служит процедура, вид которой есть 'процедура вырабатывающая целое', составленная из текста-процедуры-в-СРЕДЕ-выдающего-процедуру-вырабатывающую-целое, основа U которой является новой основой, подобной {1.1.3.2.k} i-й из этих основ-раскрыто-выдающих-целое, вместе с окружением, необходимым для U в E.

10.3.5. *Форматный обмен*

а)
```
вид формат = ст (подв [1 : 0 ] кадр F);
вид format = формат;
вид ?кадр =
  ст (цел у#казатель #т #екущего #н #абора #,
      счет #чик числа раз,
      сколько должен быть повторен кадр #,
      о #братный #ук #азатель #, подв [1 : 0] набор н);
вид ?набор = об (шаблон, пакет);
вид ?пакет =
  ст (вставка в1, проц цел повт #оритель #,
      цел у #казатель #д #ругого #к #адра #,
      вставка в2);
вид ?вставка = подв [1 : 0] ст (проц цел повт #оритель #,
  об (строк, лит) стр #ока текста #);
вид ?шаблон = ст (об (трафарет, травыб, трафор,
                      трабесф, пуст)
                  траф, вставка в);
вид ?трафарет = ст (цел тип #трафарета #, подв [1 : 0]
                                          рамка рамки);
вид ?рамка = ст (вставка в, проц цел повт #оритель #,
  лог подав #ление #, лит марк #ер #);
вид ?травыб #трафарет выбора #=
  ст (вставка в, цел тип
```

```
        #логического или целого #,
        подв [1 : 0] вставка стр #оки текста #);
    вид #трафор #трафарет форматного # =
      ст (вставка в, проц формат прф);
    вид #трабесф #трафарет бесформатного # = ст (вставка в,
                              подв [1 : 0] проц цел спец);
b)  проц #взять след #ующий #шаблон = (имя файл f,
      лог чит #ать #, имя шаблон шаблон) пуст:
    начало
      лог есть шаблон := ложь, формат окончен := ложь;
      пока ¬ есть шаблон
      цк если  ук #азатель #фор #мата #из f = 0 то
        если формат окончен
        то не определено
        инес ¬ (испр формат из f) (f)
        то имя цел (укфор из f) := 1;
          утн из (F из формат из f) [1] := 1;
          счет из (F из формат из f) [1] := 1;
        иначе формат окончен := истина
        все
      иначе
        имя цел укфор = укфор из f;
        имя подв [ ] кадр алеф = F из формат из f:
        выб (н из алеф [укфор]) [утн из алеф [укфор]]в
        (пакет пак):
        ([1 :вегр (в1 из пак)] подвставка пв;
        оук из алеф [ удк из пак] := укфор; укфор := пропуск;
        (подготовить вставку (в1 из пак, пв),
          счет из алеф [удк из пак] := повт из пак);
        (алеф :≠: F из формат из f | не определено);
        (чит | ввести вставку (f, пв)
             | вывести вставку (f, пв));
        утн из алеф [удк из пак] :=
          (счет из алеф [удк из пак] > 0|0
        | есть шаблон := истина; шаблон := (пустое, ( ));
        вегр н из алеф [удк из пак]);
        укфор := удк из пак),
    (шаблон шабл) : (есть шаблон := истина; шаблон := шабл)
быв;
пока
        (укфор ≠ 0
        | утн из алеф [укфор] = вегр н из алеф [укфор]
        | ложь)
цк если (счет из алеф [укфор] –:= 1) ⩽ 0
то
    если (укфор := оук из алеф [укфор]) ≠ 0
```

```
    то
     вставка добав =
       выб (н из алеф [укфор]) [утн из алеф [укфор]] в
       (пакет пак):
          (оук из алеф [удк из пак] := 0; в2 из пак),
        (шаблон шабл):
        выб траф из шабл в
         (трафор траф):
          (цел k := укфор;
          пока оук из алеф [k] ≠ укфор цк k +:= 1 кц;
               алеф := алеф [:k – 1 ];
           в из шабл)
        быв
       быв;
     цел m = вегр в из шаблон, n = вегр добав;
      [1 : m + n] ст (проц цел повт, об (строк, лит) стр) с;
     с [1 : m] := в из шаблон; с [m + 1 : m + n] := добав;
    в из шаблон := с
      все
     иначе утн из алеф [укфор] := 0
     все кц;
     (укфор ≠ 0 | утн из алеф [укфор] +:= 1)
    все кц
    конец;
c)  вид ♣подвставка = ст (цел повт, об (строк, лит) стр);
d)  проц ♣подготовить вставку =
       (вставка вст, имя [ ] подвставка подвст) пуст:
      ♯ совместно вызывает все повторители во 'вст' авке ♯
    если вегр вст = 1
    то
      повт из подвст [1] := повт из вст [1];
      стр из подвст [1] := стр из вст [1]
    инес вегр вст > 1
    то (подготовить вставку (вст [1], подвст [1]),
          подготовить вставку (вст [2 :], подвст [2 :]))
    все;
e)  вид ♣подрамка =
       ст (подв [1 : 0] подвставка пв, цел повт,
         лог подав, лит марк);
f)  проц ♣подготовить рамки =
       ([ ] рамка рамки, имя [ ] подрамка подрамки) пуст:
      ♯ совместно вызывает
        все повторители в рамках 'рамки' ♯
    если вегр рамки = 1
    то
```

```
        [1 : вегр (в из рамки [1])] подвставка пв;
        (подготовить вставку (в из рамки [1], пв),
        повт из подрамки [1] := повт из рамки [1]);
       пв из подрамки [1] := пв;
       подав из подрамки [1] := подав из рамки [1];
       марк из подрамки [1] := марк из рамки [1]
    инес вегр рамки > 1
    то (подготовить рамки (рамки [1], подрамки [1]),
          подготовить рамки (рамки [2:], подрамки [2:]))
    все;
g)  проц ?вывести вставку =
                (имя файл f, [ ] подвставка пв) пуст:
    начало настроить на запись (f);
     для k до вегр пв
     цк
      выб стр из пв [k] в
       (лит a): разместить (f, повт из пв [k], a, ложь),
       (строк s):
        до повт из пв [k]
        цк
         для i до вегр s
         цк проверить позицию (f);
            вывести литеру (f, s [i]) кц
        кц
      быв
    кц
      конец;
h)  проц ?ввести вставку =
                    (имя файл f, [ ] подвставка пв) пуст:
     начало настроить на чтение (f);
       для k до вегр пв
       цк
        выб стр из пв [k] в
          (лит a): разместить (f, повт из пв [k], a, истина),

         (строк s):
           (лит c;
           до повт из пв [k]
           цк
            для i до вегр s
            цк проверить позицию (f);
               ввести литеру (f, c);
             (c ≠ s [i]
           | (¬ (испр ошибка литеры из f) (f, c := s [i])
           | не определено);
```

```
                настроить на чтение (f))
            кц
          кц)
        быв
      кц
    конец;
i)  проц ?разместить =
          (имя файл f, цел повт, лит а, лог чит) пуст:
    если а = "х" то до повт вперед (f) кц
    инес а = "у" то до повт цк назад (f) кц
    инес а = "l" то до повт цк нов строчка (f) кц
    инес а = "р" то до повт цк нов страница (f) кц
    инес а = "k" то уст номер литеры (f, повт)
    инес а = "q"
    то до повт
     цк
       если чит
       то лит с; проверить позицию (f);
                 ввести литеру (f, с);
         (с ≠ пробел |
         (¬ (испр ошибка литеры из f) (f, с := пробел)
          | не определено); настроить на чтение (f))
       иначе проверить позицию (f);
             вывести литеру (f, пробел)
       все;
     кц
    все;
j)  проц ?выполнить трафор = (имя файл f,
        трафор трафор, лог чит) пуст:
    начало формат прф;
      [1 : вегр (в из трафор)] подвставка пв;
      (подготовить вставку (в из трафор, пв),
       прф := прф из трафор);
      (чит | ввести вставку (f, пв)
           | вывести вставку (f, пв));
      имя цел укфор = укфор из f;
      имя подв [ ] кадр алеф = f из формата из f;
      цел m = вегр алеф, n = вегр (F из прф);
      [1 : m + n] кадр с; с [1 : m] := алеф;
      с [m + 1 : m + n] := F из прф;
      алеф := с; оук из алеф [m + 1] := укфор;
      укфор := m + 1; утн из алеф [укфор] := 1;
      счет из алеф [укфор] := 1;
      для i от m +1 до m + n
      цк
```

```
        для j до вегр н из алеф [i]
        цк
         выб (н из алеф [i]) [j] в
          (пакет пак):
            (н из алеф [i] [j] := пакет (в 1 из пак, повт
             из пак, удк из пак + m,в 2 из пак)
         быв
        кц
       кц
       конец;
k)  проц присоединить формат =
                          (имя файл f, формат формат) пуст:
       начало
        формат из f :=
         с вновь созданное имя, которое начинает именовать
          выдачу фактического-описателя-формата и область
          действия которого равна области действия
          значения, выдаваемого посредством 'формат' с
         := формат;
         укфор из f := глоб цел := 1;
         утн из (F из формат из f) [1] := 1;
         счет из (F из формат из f) [1] := 1;
         оук из (F из формат из f) [1] := 0
        конец;
```

10.3.5.1. Форматный вывод.

```
a)  проц ф вывод =
      (имя файл f, [ ] об (выводимое, формат) x) пуст:
      если открыт из f то
     для k до вегр x
      цк выб настроить на запись (f);
            настроить на литерное (f);
      x [k] в
      (формат формат) : присоединить формат (f, формат),
      (выводимое выв) :
      начало цел j := 0
        шаблон шаблон, [ ] провывод у = стройвывод выв;
        пока (j +:= 1) ≤ вегр у
        цк лог не конч := ложь;
         взять след шаблон (f, ложь, шаблон);
         настроить на запись (f);
          [1 : вегр (в из шаблон)] подвставка подвст;
         выб траф из шаблон в
        (трафарет трафарет):
      начало цел повт, ук#азатель #рам #ок # := 1;
       [1 : вегр (рамки из трафарет)] подрамка подрамки;
```

```
  (подготовить рамки (рамки из трафарет, подрамки),
   подготовить вставку (в из шаблон, подвст));
 строк s;
 оп ? = (строк s) лог:
 #истина, если следующий маркер есть один из
   элементов 's', а иначе ложь #
   если украм > верг подрамки
   то ложь
   иначе подрамка пр = подрамки [украм];
    повт := повт из пр;
    если литера в строке
                        (марк из пр, лог цел, s)
    то украм +:= 1; истина
    иначе ложь
    все
 все;
 оп ? = (лит с) лог: ?строк (с);
 проц цел трафарет =
                (имя лог образец знака) цел:
  (цел l := 0;
  пока ? "zuv" цк (повт ⩾ 0 | l +:= повт) кц;
  образец знака := ? "+ –";
  пока ? "zd"
  цк (повт ⩾ 0 | l +:= повт) кц; l);
≮ проц ред #актировать #Д цел = (Д цел i) пуст:
  (лог образец знака;
  цел l := цел трафарет (образец знака);
  строк t = предст целого (абс i, l);
  если литера в строке (литера ошибки лок цел, t) ∨
    l = 0 ∨ ¬ образец знака ∧ i < Д 0
  то не конч := истина
  иначе t прип s;
     (l – верг t) × "0" прип s;
     (образец знака | (i < Д 0 | "–" | "+") прип s)
  все) ≯;
≮ проп ред Д вещ = (Д вещ в) пуст:
  (цел b := 0, a := 0,
      пор := 0, Д вещ у := абс в,
  лог знак 1, строк точка = "";
  b := цел трафарет (знак 1);
  (? ". " | a := цел трафарет (лок лог);
       точка := ".");
  если ? "e"
    то Д нормализовать (у, b, a, пор);
     ред цел (пор);
```

```
          "10" прип s
        все;
        строк t = предст рационального
                              (y, b + a + (a ≠ 0 | 1 | 0), a);
        если литера в строке
                              (литера ошибки, лок цел, t) ∨
        a + b = 0 ∨ ¬ знак 1 ∧ r < Д 0
        то не конч := истина
        иначе t [:b] + точка + t [b + 2:] прип s;
          (b + a + (a ≠ 0 | 1 | 0) – вегр t) × "0" прип s;
          (знак 1 | (в < Д 0 | "–" | "+") прип s)
        все) ⊱;
      ⊰ проц ред Д компл = (Д компл дк) пуст:
        пока ?" i" цк украм +:= 1 кц;
        ред Д вещ (мч дк);
        "⊥" прип s; украм := 1; ред Д вещ (вч дк)) ⊱;
      ⊰ проц ред Д бит =
                        (Д бит дб, цел основание) пуст:
        (Д цел n := абс дб; ? "r";
          цел l := цел трафарет (лок лог);
пока литеру в цифру (С (n мод У основание)) прип s;
      n ÷:= У основание; n ≠ Д 0
цк пропуск кц;
если вегр s ⩽ l
то (l – вегр  s) × "0" прип s
иначе не конч := истина
все) ⊱,
проц счет #чик #лит #ер # = цел:
    (цел l := 0;
    пока ? "a" цк (повт ⩾ 0  | l +:= повт) кц; l);
    выб тип из трафарет в
#целое #
    (y [j] |
    ⊰ (Д цел i) : ред Д цел (i) ⊱
    | не конч := истина),
#вещественное #
    (y [j] |
    ⊰ (Д вещ в) : ред Д вещ (в) ⊱,
    ⊰ (Д цел ц) : ред Д вещ (ц) ⊱,
    | не конч := истина),
#логическое #
    (y [j] |
    (лог b) : s := (b | да | нет)
    | не конч := истина),
# комплексное #
```

```
      (у [j] |
      ≮ (Д компл дк) : ред Д компл (дк) ≯,
      ≮ (Д вещ дв) : ред Д вещ (вд) ≯,
      ≮ (Д цел дц) : ред Д вещ (дц) ≯
      | не конч := истина),
  ♯ строковое ♯
      (у [j] |
      (лит с) : (счет лит = 1 | s := с|
       не конч := истина),
      ([ ] лит t) :
        (счет лит = вегр t – нигр t + 1
        |s := t [с1]
        | не конч := истина)
      | не конч := истина)
      либо
  ♯ битовое ♯
      (у [j] |
        ≮ (Д бит дб) : ред Д бит (дб, тип из трафарет – 4) ≯
      | не конч := истина)
      быв;
       если ¬ не конч
       то ред строку (f, s, подрамки)
       все
  конец,
      (травыб выбор) :
      начало
        [1 : вегр (в из выбор)] подвставка пв;
       подготовить вставку (в из выбор, пв);
       вывести вставку (f, пв);
       цел l =
         выб тип из выбор в
       ♯ логическое ♯
         (у [j] |
         (лог b) : (b | 1 | 2)
         | не конч := истина; пропуск),
       ♯ целое ♯
         (у [j] |
         (цел i) : i
         | не конч := истина; пропуск)
       быв;
      если ¬ не конч
      то
        если l > вегр (стр из выбор) ∨ l ≤ 0
        то не конч := истина
        иначе
```

```
      [1 : вегр ((стр из выбор) [l])] подвставка пвс;
      подготовить вставку ((стр из выбор) [l], пвс);
      вывести вставку (f, пвс)
     все
    все;
    подготовить вставку (в из шаблон, подвст)
  конец,
  (трафор трафор):
  начало
    выполнить трафор (f ,трафор, ложь);
    для i до вегр подвст цк подвст      [i] := (0, '' '') кц;
    j –:= 1
  конец,
  (трабесф трабесф):
  начало
    [1 : вегр (в из трабесф)] подвставка пв;
     [ ] проц цел спец = спец из трабесф;
   цел n = вегр спец; [1 : n] цел s;
   (подготовить вставку (в из трабесф, пв),
      (подготовить вставку (в из шаблон, подвст),
        s := (n | спец [1], (спец [1], спец [2]),
             (спец [1], спец [2], спец [3]) | ( ) ));
  вывести вставку (f, пв);
  если n = 0 то вывод (f, у [i])
  иначе
      число yj =
        (у [j] | ≮ (Д цел i): i ≯, ≮ (Д вещ в): в ≯
        | не конч := истина; пропуск);
    если ¬ не конч
    то выб n в
       вывод (f, целое (yj, s [1])),
       вывод (f, фикс (yj, s [1], s[2])),
       вывод (f, плав (yj, s[1], s[2], s[3]))
       быв
      все
    все
   конец,
  (пуст):
     (j –:= 1;
    подготовить вставку (в из шаблон, подвст))
   быв;
   если не конч
   то настроить на запись (f);
      (¬ (испр ошибка значения из f) (f) | вывод (f, у [j]);
        не определено)
```

```
    все;
    вывести вставку (f, подвст)
   кц
  конец
  быв кц
  иначе не определено
  все;
  проц (имя файл,
        [ ] об (выводимое, формат)) пуст put f = ф вывод:
b) проц ?ред ф актировать ф строку =
   (имя файл f, строк s, [ ] подрамка пр) пуст:
  начало лог подав, п ф одавл ф н ф ули ф := истина,
   выв ф еден ф зн
   ф ак ф := ложь, еще раз, цел j := 0, знак:
   проц копия = (лит c) пуст:
   (¬ подав | проверить позицию (f); вывести литеру (f, c));
  для k до вегр пр)
  цк подрамка прк = пр [k]: подав := подав из прк:
   вывести вставку (f, пв из прк);
   до повт из прк
   цк еще раз := истина:
    пока еще раз
    цк j +:= 1; еще раз := ложь:
     лит sj = s [j], маркер = марк из прк:
     если маркер = "d"
     то копия ( sj); пн := истина
     инес маркер = "z" то
      (sj = "0" | копия ((пн | "." | sj))
      | : sj = "+" | еще раз := истина
     | пн := ложь; копия (sj))
    инес маркер = "u" ∨ маркер = "v" то
       (sj = "+" | знак := 1; еще раз := истина
       | : sj = "–" | знак := 2; еще раз := истина
       | :sj = "0" | копия ((пн | "." | sj))
       | (¬ выв зн
         копия ((знак | (маркер = "u" | "+"| "."), "–"));
          выв зн := истина);
        копия (sj): пн := ложь)
    инес маркер = "+" то
        (sj = "+" ∨ sj = "–" | копия (sj)
        | (¬ выв зн | копия ((знак | "+", "–"))));
        j –:= 1)
    инес маркер = "–" то
        (sj = "+" | копия (".")
        | : sj = "–" | копия (sj)
```

```
        | (¬ выв зн | копия ((знак | "+" "–")));
            j –:= 1)
        инес маркер = ". " то
            копия (". ")
        инес маркер = "e" ∨ маркер = "i"
          ∨ маркер = "a" ∨ маркер = "b"
        то копия (sj); пн := истина; выв зн := ложь
        инес маркер = "r"
        то j –:= 1
       все
      кц
     кц
    кц
  конец;
```

10.3.5.2. Форматный ввод

```
а)  проц ф ввод = (имя файл, f,
                   [ ] об (вводимое, формат) x) пуст:
      если открыт из f то
      для k до вегр x
      цк выб настроить на чтение (f);
        настроить на литерное (f); x [k] в
        (формат формат) : присоединить формат (f, формат),
        (вводимое вв) :
        начало цел j := 0;
          шаблон шаблон, [ ] проввод у = стройввод вв;
          пока (j +:= 1) ⩽ вегр у
          цк лог не конч := ложь;
            взять след шаблон (f, истина, шаблон);
            настроить на чтение (f);
            [1 : вегр (в из шаблон)] подвставка подвст;
            выб траф из шаблон в
            (трафарет трафарет) :
          начало
            [1 : вегр (рамки из трафарет)] подрамка подрамки;
            (подготовить рамки (рамки из трафарет, подрамки),
            подготовить вставку (в из шаблон, подвставка));
          строк s;
          цел основание =
            (тип из трафарет ⩾ 6 | тип из трафарет – 4 | 10);
          сост строку (f, s, подрамки, основание);
          выб тип из трафарет в
        #| целое |#
            (у [j] |
            ≮ (имя Д цел идц) :
            не конч := ¬ строку в Д цел (s, 10, идц) ≯
```

```
      | не конч := истина),
    ⫽ вещественное ⫽
      (у [j] |
      ≮ (имя Д вещ идв):
        не конч := ¬ строку в Д вещ (s, идв) ≯
      | не конч := истина),
    ⫽ логическое ⫽
      (у [j] | (имя лог ил): ил := s = да
      | не конч := истина),
    ⫽ комплексное ⫽
      (у [j] |
      ≮ (имя Д компл идк):
        (цел i, лог b1, b2; литера в строке ("⊥", i, s);
          b1 := строку в Д вещ (s [: −1], ивч из идк);
          b2 := строку в Д вещ (s [i + 1·], имч из идк);
          не конч := ¬ (b1 ∧ b2)) ≯
              | не конч := истина),
    ⫽ строковое ⫽
      (у [j] |
      (имя лит сс):
        (вегр s = 1 | сс := s [1] | не конч := истина),
      (имя [ ] лит имл):
        (вегр имл − нигр имл + 1 = вегр s | имл [с 1] := s
        | не конч := истина),
        (имя строк ис) : ис:= s
        | не конч := истина),
        либо
    ⫽ битовые ⫽
      (у [j] |
      ≮ (имя Д бит идб):
        если Д цел i: строку в Д цел (s, основание, i)
        то идб := бин i
        иначе не конч := истина
        все ≯
      | не конч := истина)
    быв
конец,
(травыб выбор):
начало
    [1 : вегр (в из выбор)] подвставка пв;
    подготовить вставку (в из выбор, пв);
    ввести вставку (f, пв);
    цел с = с из тпоз f, лит kk;
    цел k := 0, лог есть := ложь;
    пока k < вегр (стр из выбор) ∧ ¬ есть
```

```
    цк k +:= 1;
        [1 : вегр ((стр из выбор) [k]) подвставка пв;
      лог лог := истина;
      подготовить вставку ((стр из выбор) [k], пв);
      строк s;
      для i до вегр пв
      цк s плюспр
        (стр из пв [i] | (строк ss) : ss) × повт из пв [i]
      цк;
      для jj до вегр s
      пока лог := лог ∧ ¬ строчка окончена (f)
     ∧ ¬ лог файл окончен (f)
      цк ввести литеру (f, kk); лог := kk = s [jj] кц;
       ( ¬ (есть := лог) | уст номер литеры (f, c))
кц;
если ¬ есть то не конч := истина
иначе
      выб тип из выбор в
#логическое#
      ( у [j] |
      (имя лог b) : b := k = 1
     | не конч := истина),
#целое#
      (у [j] |
      (имя цел i) : i := k
        | не конч := истина)
      быв
      все;
      подготовить вставку (в из шаблон, подвст)
конец,
      (трафор трафор) :
      начало выполнить трафор (f, трафор, истина);
  для i до вегр подвст цк подвст [i] := (0, " ") кц;
          j := 1
       конец,
       (трабесф трабесф) :
          ([1 : вегр (в из трабесф)] подвставка пв;
          (подготовить вставку (в из трабесф, пв),
          подготовить вставку (в из шаблон, подвст));
          ввести вставку (f, пв);
          ввод (f, у [j])),
     (пуст) :
          (j ¬ := 1;
          подготовить вставку (в из шаблон, подвст))
     быв;
```

```
      если не конч
      то настроить на чтение (f);
        (¬ (испр ошибка значения из f) (f) | не определено)
      все:
      ввести вставку (f, подвст)
     кц
     конец
    быв кц
    иначе не определено
    все;
   проц (имя файл,
         [ ] об (вводимое, формат)) пуст get f = ф ввод;
b) проц ?сост #авить #строку = (имя файл f,
    имя строк s, [ ] подрамка пр, цел основание) пуст:
   начало
    лог подав, н #одавляемые #н #ули #:= истина,
    есть знак := ложь,
    есть пробел := ложь, нет знака := ложь,
    цел ук #азатель #зн #ака #:= 1, повт;
   прио ! = 8;
   оп ! = (строк s, лит c) лит:
    #запрашивает некоторую литеру, содержащуюся в 's';
    если читаемая литера не входит в 's', то вызывается
    процедура обработки события, соответствующая 'при
    ошибке литеры', с предлагаемым 'c' #
    если лит k; проверить позицию (f);
                ввести литеру (f, k);
     литера в строке (k, лок цел, s)
    то k
    иначе лит предл := c;
    если (испр ошибка литеры из f) (f, предл) то
      (литера в строке (предл, лок цел, s) | предл
      | не определено; c)
    иначе не определено; c
    все;
    настроить на чтение (f)
    все;
   оп ! = (лит s, c) лит: строк (s) ! c;
   [ ] лит хор цифры = "0123456789abcdef" [:основание];
   s :="+";
   для k до вегр пр
   цк подрамка прк = пр[k]; подав := подав из прк;
     ввести вставку (f, пв из прк);
    до повт из прк
    цк лит маркер = марк из прк;
```

```
 если маркер = "d" то s плюспр
       (подав |"0"| хор цифры ! "0"); пн := истина
инес маркер = "z" то s плюспр
  (подав | "0"
   | лит с = ((пн | "-" | "  ") + хор цифры) ! "0";
     (с ≠ "-" (пн := ложь); с)
инес маркер = "u" ∨ маркер = "+" то
  если есть знак
  то пн := ложь; s плюспр ("0123456789" ! "0")
  иначе
   лит с = ("+ –" + (маркер = "u" | "-"| "  ")) ! "+";
   (с = "+" ∨ с = "–"
    | есть знак := истина; s [укзн] := с)
 все
инес маркер = "v" ∨ маркер = "–" то
  если есть знак
  то пн := ложь; s плюспр ("0123456789" ! "0")
  инес лит с; есть пробел
 то с := "+ – ".0123456789" ! "+";
  (с = "+" ∨ с "–"
   | есть знак := истина;
   s [укзн] := с
   | : с ≠ "-"| пн := ложь; есть знак := истина;
  s плюспр с)
 иначе с:=" +" –." ! "+";
   (с = "+" ∨ с = "–"
    | есть знак := истина; s [укзн] := с
    | есть пробел := истина)
 все
инес маркер = " . " то
   s плюспр (подав | " . " | " . " ! " . ")
  инес маркер = „e" то s плюспр
   (подав | "10" | "10\e" ! "10"; "10");
   есть знак := ложь; пн := истина;
   s плюспр "+"; укзн := вегр s
инес маркер = "i" то
   s плюспр (подав | "⊥" | "i ⊥" ! "⊥", "⊥");
   есть знак := ложь; пн := истина;
   s плюспр "+"; укзн := вегр s
инес маркер = "h" то
   s плюспр (да + нет) ) нет;
    нет знака := истина
инес маркер = "a" то s плюспр
   (подав |"-"| лит с; проверить позицию (f);
   вывести литеру (f, с);
```

```
        с);
        нет знака := истина
       инес маркер = "г"
       то пропуск
       все
      кц
     кц;
    если нет знака то s := s [2:] все
конец;
```

10.3.6. *Двоичный обмен*

{В двоичном обмене значения, полученные выстраиванием элементов списка данных (п. 10.3.3.), обмениваются одно за другим через данный файл. Способ, которым такие значения хранятся в книге, определен только до такой степени, чтобы значение вида М, выведенное на данную позицию, можно было впоследствии ввести обратно с той же позиции в имя вида 'имя М' (М – один из тех видов, из которых объединен вид провывод). Отметим, что в ходе ввода в имя, именующее массив, число прочитанных элементов будет равно существующему числу элементов, именуемых этим именем.

После каждого значения текущая позиция продвигается на соответствующую величину и на конце каждой строчки или страницы вызывается подходящая процедура обработки события; если она вырабатывает ложь, ищется следующая хорошая позиция литеры в данной книге (п. 10.3.3).

Для двоичного вывода можно использовать процедуры дв вывод (10.3.6.1.a) и дв зап (10.5.1.h), для двоичного ввода процедуры дв ввод (10.3.6.2.a) и дв чит (10.5.1.i)}.

```
а) проц в двоичное = (имя файл f, провывод x) [ ] лит:
    { значение вида 'вектор из литер', нижняя граница
      которого равна единице, а верхняя граница зависит
      от значения 'книга из f' и от вида и значения
      данного 'x', кроме того,
      x = из двоичного (f; x, в двоичное (f, x)) с;
b) проц из двоичного= (имя файл f,
                       провывод y, [ ]лит c) провывод:
    с значение, если оно есть, вида значения,
      выдаваемого 'y', такое, что
      c = в двоичное (f, из двоичного (f, y, c)) с;
```

10.3.6.1. Двоичный вывод.

```
а) проц дв вывод = (имя файл f, [ ] выводимое выв) пуст:
    если открыт из f то
     настроить на двоичное (f); настроить на запись (f);
     для k до вегр выв
     цк [ ] провывод y = стройвывод выв [k];
       для j до вегр y
       цк [ ] лит двоичн = в двоичное (f, y [j]);
```

```
            для i до вегр двоичн
          цк след позиция (f);
           настроить на двоичное (f);
           имя позиция тпоз = тпоз из f,
             ппоз = заполн из книга из f;
           выб текст из f в
           (подвтекст t2):
           t2 [р из тпоз]
              [l из тпоз]
              [с из тпоз] := двоичн [i]
          быв;
          с из тпоз +:= 1;
           если тпоз вне ппоз то ппоз := тпоз
           инес ¬ возм установка (f)
           ∧ позиция (р из ппоз, l из ппоз, 1) вне тпоз
           то ппоз := тпоз;
             (сжимаем (f) |
              с размеры строчки и страницы, содержащих
               логическии конец книги,
              и всех последующих строчек
              и страниц могут увеличиться с)
          все
        кц
      кц
    кц
    иначе не определено
    все;
    проц (имя файл, выводимое) пуст put bin = дв вывод;
    10.3.6.2. Двоичный ввод.
а)  проц дв.ввод = (имя файл f, [ ] вводимое вв) пуст:
     если открыт из f то
    настроить на двоичное (f); настроить на чтение (f);
    для k до вегр вв
    цк [ ] проввод y = стройввод вв [k];
      для j до вегр y
      цк
       провывод yj = выб y [j] в
       ≮ (имя Д цел i) : i ≯, ≮ (имя Д бит r) : r ≯,
       ≮ (имя Д компл идк) : идк ≯,
        (имя лог b) : b, ≮ (имя Д бит идб) : идб ≯,
        (имя лит c) : c, (имя [ ] лит s) : s,
        (имя строк ис) : ис быв;
    [1 : вегр (в двоичное (f, yj)) ] лит двоичн;
    для i до вегр двоичн
    цк след позиция (f); настроить на двоичное (f);
```

```
    имя позиции тпоз = тпоз из f;
    двоичн [j] :=
      выб текст из f в
      (подвтекст t2):
       t2 [р из тпоз] [l из тпоз] [с из тпоз]
      быв;
    с из тпоз +:= 1
  кц;
    выб у [j] в
     ≮ (имя Д цел идц) : (из двоичного (f, идц, двоичн) |
        (Д цел дц) : идц := дц) ≯,
     ≮ (имя Д вещ идв) : (из двоичного (f: идв, двоичн) |
       (Д вещ дв) : идв := дв) ≯,
     ≮ (имя Д компл идк) :
       (из двоичного (f, идк, двоичн) |
       (Д компл дк) : идк := дк) ≯,
     (имя лог ил) : (из двоичного (f, ил, двоичн) |
       (лог л) : ил := л),
     ≮ (имя Д бит идб) : (из двоичного (f, идб, двоичн) |
       (Д бит дб) : идб := дб) ≯,
     (имя лит ил) : (из двоичного (f, ил, двоичн) |
       (лит л) : ил := л),
     (имя [ ] лит имл) : (из двоичного (f, имл, двоичн) |
       (лит мл) : имл := мл),
     (имя строк ис) : (из двоичного (f, ис, двоичн) |
       ([ ] лит с) : ис := с)
     быв
    кц
   кц
   иначе не определено
  все;
 проц (имя файл, [ ] вводимое) пуст get bin = дв ввод;
```

10.4. Системное вступление и список задач

10.4.1. *Системное вступление*

Представление системного-вступления получается из нижеследующей формы, к которой могут добавиться дальнейшие формы, не определяемые в настоящем стандарте. { Синтаксис текстов-программы обеспечивает, что никакое описание, содержащееся в этом системном вступлении, не может противоречить никакому описанию в стандартном-вступлении. Предполагается, что эти дальнейшие формы могли бы содержать описания, необходимые для правильной работы любых системных-задач, которые могут добавиться (реализациями, как это предусмотрено в 10.1.2.d).}

```
а) сема ?гномы = (сема s; F из s := ПЕРВ цел:= 0; s);
```

10.4.2. *Список системных задач*

Представление { первой } составляющей системной-задачи данного

списка-системных-задач получается из нижеследующей формы. Другие системные-задачи, если они будут, не определяются настоящим стандартом {, но их можно определить в реализации, чтобы учесть конкретные свойства ее операционной обстановки, особенно в связи с тем, что она взаимодействует с выполнением собственно-программ (см., например, 10.3.1.1.dd).}

а) цк вниз гномы; не определено; вверх защита связей кц

{Предлагается, что вызываемая здесь процедура не определено, которая запускается посредством вверх гномы всякий раз, когда закрывается какая-нибудь книга, может реорганизовать как цепочку связок доступные книги, так и цепочку снятые книги, путем устранения книги, если она не должна быть доступна для последующего открытия, или путем вставки ее в цепочку связок еще несколько раз, если для нескольких собственно-программ требуется позволить читать эту книгу одновременно. Всякий раз, когда дается вверх гномы, семафор защита связей уже внизу и остается в этом положении, пока осуществляется рассматриваемая реорганизация.}

10.5. Собственные вступления и заключения

10.5.1. *Собственные вступления*

Представление собственного-вступления каждой задачи-пользователя получается из нижеследующих форм, к которым можно добавить другие такие формы – они могут понадобиться для правильной работы средств, определяемых составляющим библиотечным-вступлением данного текста-программы {, например, описания и вызовы открыть для дополнительных стандартных файлов}. Однако для каждого определяющего-ОБОЗНАЧЕНИЕ-индикатора-в-новом-с-новыми-!ПАРАМИ-с-СЛОЕМ2-выдающего-ПРИЗНАК, содержащегося в каждой такой дополнительной форме, должен выполняться предикат 'если ОБОЗНАЧЕНИЕ для ПРИЗНАКА не зависит от !ПАР1' {7.1.1.a, c}{, т.е. не может возникнуть противоречия ни с каким описанием, содержащимся в стандартном-вступлении}.

```
a) Д цел Д пред псч # псевдослучайное число # :=
     окр (Д макс цел/Д 2);
   Д цел L last random = Д пред псч;
b) проц Д псч = Д вещ;
     Д след псч (Д пред псч);
   проц Д вещ L random = Д псч;
c) файл станд ввод, станд вывод, станд обмен:
   имя файл stand in = станд ввод;
   имя файл stand out = станд вывод;
   имя файл stand back = станд обмен;
     открыть (станд ввод, „ ", станд канал ввода);
     открыть (станд вывод, „ ", станд канал вывода);
     открыть (станд обмен, „ ", станд канал обмена);
d) проц печ = ([ ] об (выводимое,
                       проц (имя файл) пуст) х) пуст:
```

```
    вывод (станд вывод, х);
  проц (об (выводимое,
    проц (имя файл) пуст)) пуст print = печ,
    зап = печ, write = печ;
e) проц чит = ([ ] об (вводимое,
    проц (имя файл) пуст) х) пуст:
    ввод (станд ввод, х);
  проц (об (вводимое, проц (имя файл) пуст)) пуст read = чит;
f) проц ф печ = ([ ] об (выводимое, формат ) х) пуст:
    ф вывод (станд вывод, х),
  проц ф зап = ([ ] об (выводимое, формат) х) пуст:
    ф вывод (станд вывод, х);
  проц ([ ] об (выводимое, формат)) пуст printf = ф печ,
    writef = ф зап;
g) проц ф чит ([ ] об (вводимое, формат) х) пуст:
    ф ввод (станд, ввод, х);
  проц ([ ] об (вводимое, формат)) пуст readf = ф чит;
h) проц дв зап = ([ ] выводимое х) пуст:
    дв вывод (станд обмен, х);
  проц ([ ] вводимое) пуст write bin = дв зап;
i) проц дв чит = ([ ] вводимое х) пуст:
    дв ввод (станд обмен, х);
  проц ([ ] вводимое) пуст read bin = дв чит;
```

10.5.. *Собственные заключения*

Представление собственного-заключения каждой задачи-пользователя получается из нижеследующей формы, к которой можно добавить другие такие формы – они могут понадобиться для правильного функционирования возможностей, определенных в составляющем библиотечном-вступлении данного текста-программы { , например, вызовы процедуры снять для дополнительных стандартных файлов } .

```
a) stop: стоп: снять (станд ввод); снять (станд вывод); снять (станд
     обмен)
```

ПРИЛОЖЕНИЕ 1
Справочное

ИСТОРИЧЕСКАЯ СПРАВКА

Язык Алгол 68 был разработан в рабочей группе 2.1 (РГ 2.1) Международной Федерации по обработке информации (ИФИП).

В период с 1963 по 1968 гг. на заседаниях РГ 2.1 обсуждались подходы к созданию нового языка и различные проекты языка.

В 1968 г. РГ 2.1 после обсуждения ряда последовательных версий приняла описание языка, разработанное в Амстердаме А. ван Вейнгаарденом, Б. Майу, Дж. Пеком и К. Костером. Этот язык, получивший по году принятия название Алгол 68, был одоб-

рен для публикации Генеральной ассамблеей ИФИП. Сообщение об Алголе 68 было опубликовано на английском языке (A. van Wijngaarden (Editor), B.J. Mailloux, J.E.L. Peck and C.H.A Koster. Report on the Algorithmic Language ALGOL 68, Numerische Mathematik, Vol. 14, pp. 19–218, 1969) а также переведено на ряд других языков, включая русский (Сообщение об алгоритмическом языке Алгол 68: Пер. с англ. А. А. Берса, А. П. Ершова, Л. Л. Змиевской, А. Ф. Рара // Кибернетика. – Киев. – 1969. – №6, – –1970. – №1.

В последующие годы Алгол 68 обсуждался и испытывался как членами РГ 2.1, так и вне ее, что привело к разработке нового варианта языка, содержащего целый ряд изменений, особенно в способе описания. Новый вариант в 1974 г. был принят РГ 2.1, затем утвержден техническим комитетом 2 (ТК 2) по программированию ИФИП и одобрен для публикации Генеральной Ассамблеей ИФИП. Он был опубликован в 1975 г. (Revised Report on the Algorithmic Language ALGOL 68. Edited by A.van Wijngaarden, B.J.Mailloux, J.E.L.Peck, C.H.A.Koster, M.Sintzoff, C.H.Lindsey, L.G.L.T. Meertens and R.G. Fisker. Acta Informatica.,Vol. 5. Fasc. 1–3, 1975).

Эта публикация была принята в качестве окончательного определяющего документа для Алгола 68, не подлежащего каким бы то ни было изменениям. Различные дополнительные возможности, предлагавшиеся в дальнейшем реализаторами языка, а также самой РГ 2.1, официально не включены в Алгол 68 и не отражены в настоящем стандарте.

Алгол 68 разрабатывался как международный язык программирования, поэтому определяющий документ, хотя и написан на английском языке и использует служебные слова и обозначения на основе английского языка, предусматривает возможность создания в соответствии с некоторыми правилами версий Алгола 68 для других национальных языков и переводов определяющего документа на другие языки. Перевод на русский язык одновременно содержит и русскую версию языка Алгол 68, позволяющую наряду с английскими служебными словами и обозначениями использовать служебные слова и обозначения на основе русского языка. Данный перевод был рассмотрен и принят временной научно-технической комиссией (ВНТК), специально созданной для этой цели при Государственном Комитете Совета Министров СССР по науке и технике (ГКНТ) в 1976 г. и утвержден ГКНТ в качестве определяющего документа для русской версии Алгола 68. Этот документ опубликован в 1979 г. параллельно с английским текстом (Пересмотренное сообщение об Алголе 68; Ред. А. Ван Вейнгаарден, Б. Майу, Дж. Пек, К. Костер, М. Синцов, Ч. Линдси, Л. Меертенс, Р. Фискер: Пер. с англ. А. А. Берса, под ред. А. П. Ершова – М.: Мир, 1979).

Настоящий стандарт воспроизводит часть публикации 1979 г. с минимальными изменениями технического характера.

Требования к машинному представлению программы, содержащиеся в приложении 2, также основаны на документе ИФИП (Wilfred J. Hansen, Hendrik Boom. The Report on the standard hardware representation for ALGOL 68. ALGOL Bulletin, 40, pp. 25–43), принятом РГ 2.1, утвержденном ТК 2 и одобренном для публикации Генеральной Ассамблеей ИФИП. В буквальном виде этот документ не может быть принят для русской версии из-за необходимости одновременного использования русского и латинского алфавитов и особенностей символьных наборов отечественных устройств подготовки и отображения информации. Правила машинного представления для русской версии Алгола 68 неоднократно обсуждались ВНТК и рабочей группой по алгоритмическому языку Алгол 68 (РГ А68) комиссии по языкам и системам программирования при ГКНТ. ВНТК приняла вариант машинного представления, предложенный В. В. Бролем и впоследствии реализованный В. Б. Яковлевым в трансляторе для МВК „Эльбрус".

При подготовке настоящего стандарта В. В. Бролем и В. Б. Яковлевым с учетом опыта использования реализации на МВК „Эльбрус" разработан новый вариант машинного представления программ на Алголе 68.

ПРИЛОЖЕНИЕ 2
Обязательное

ТРЕБОВАНИЯ К МАШИННОМУ ПРЕДСТАВЛЕНИЮ ПРОГРАММЫ

1. ОПРЕДЕЛЕНИЯ

1.1. Абстрактная литера – это одна из следующих 152 литер:
буквы латинского алфавита:
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z,
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
буквы русского алфавита:
А, Б, В, Г, Д, Е, Ё, Ж, З, И, Й, К, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я,
а, б, в, г, д, е, ё, ж, з, и, й, к, л, м, н, о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю, я
цифры:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9
прочие литеры:
пробел " # $ % ' ( ) * + , – . / : ; < = > @ [ ] _ |

Представление программы на Алголе 68 определяют как разделенную на строчки последовательность абстрактных литер.

1.2. Конкретная литера – это некоторая литера, имеющаяся на устройстве ввода-вывода. Каждая такая литера составлена из множества знаков и кодов в соответствии с местными соглашениями.

1.3. Разделитель – это особенность типографского набора (п. 9.4.d), начало или конец текста программы или любая абстрактная литера, отличная от буквы, цифры или знака подчеркивания. Слова и выделенные слова ограничиваются разделителями.

1.4. Две строки литер соприкасаются, если между ними нет литер или особенностей типографского набора. Если одна из строк литер следует за или предшествует другой, то они также соприкасаются.

1.5. Выделенное слово – это:
любое представление, составленное из выделенных букв или цифр в эталонном языке (подраздел 9.4) (т.е. символы-выделенное-СЛОВО и представления, указанные в п.9.4.1 как выделенные);
символ, представленный выделенным словом;
литеры, записывающие выделенное слово способом, специфицированным в подразделе 3.4 настоящего приложения.

1.6. Слово – это символ-СЛОВО (подпункт 9.4.2.2.а), например, – „конец файла" – это слово.

1.7. Слог – непустая последовательность букв и цифр (слово „конец файла", использованное так, как в п.3.4.1 настоящего приложения, состоит из двух слогов).

2. ПРЕДСТАВЛЕНИЕ КОНСТРУКТОВ АЛГОЛА 68

2.1. Для каждой абстрактной литеры реализация должна предусматривать одну или несколько конкретных литер, отличающихся от конкретных литер для других абстрактных литер.

Если предусмотрено несколько конкретных литер (например, для „|" и „l" и „!") то они должны эквивалентно обрабатываться всюду, кроме как в строках и при распечатке программ, где каждая представляет саму себя.

2.2. В каждом алфавите соответствующие друг другу прописные и строчные буквы эквивалентны, за исключением ситуаций, предусмотренных в подразделе 3.1 и п.3.5.2 настоящего приложения.

2.3. Конструкт в языке представления получается заменой символов на их представления. Представление для каждого символа дается в терминах абстрактных литер. Кодирование конструктов в языке представления для машинной обработки осуществляется заменой каждой абстрактной литеры на соответствующую ей конкретную литеру и вставкой особенностей типографского набора (там, где это разрешено).

В некоторых реализациях отдельные конкретные литеры могут не иметь представлений на устройствах ввода-вывода. В таких случаях реализация должна обеспечить дополнительное представление соответствующих им абстрактных литер (например, в виде комбинаций конкретных литер, доступных на этих устройствах). При этом, в частности, допускается расширение списка зарезервированных слов из подпункта 3.4.1.3 настоящего приложения, а также введение альтернативных обозначений из раздела 10 и расширение списка представлений символов из п.9.4.1 настоящего стандарта альтернативными представлениями, использующими только доступные конкретные литеры.

Чтобы перенос программ сводился только к простой транслитерации, реализация должна предусматривать способ представления программ в виде, не использующем подобных местных соглашений, и преобразование программ к такому виду.

3. ОТДЕЛЬНЫЕ ПРЕДСТАВЛЕНИЯ

3.1. Элементы строки

3.1.1. Множество элементов-строки (подпункт 8.1.4.1.b) – это множество абстрактных литер без кавычки и апострофа, но с символом-образ-кавычки и символом-образ-апострофа. Значение каждой абстрактной литеры есть сама литера. Соответствующие друг другу прописные и строчные буквы имеют различные естественные значения. Симол-образ-кавычки записывают двумя соприкасающимися кавычками и его естественным значением является кавычка. Символ-образ-апострофа записывают двумя соприкасающимися апострофами и его естественным значением является апостроф (один апостроф может использоваться в реализациях как регистровая литера).

Последовательность управляющих литер, отсутствующих на устройствах вводавывода, или одна такая литера в изображении-строки может быть представлена следующим образом:

символ-апостроф,
символ-открыть,
символ-образов-управляющих-литер,
символ-закрыть.

Образы литер в списке могут задаваться целыми десятичными числами, а также их обозначениями в русской или латинской нотации по ГОСТ 27465 и должны разделяться запятыми.

3.1.2. Дополнительная особенность типографского набора – „разрыв строки" – предусмотрена для использования только внутри изображений-строк и изображений-литерных и записывается как кавычка с последующими одной или более особенностями типографского набора, отличными от разрыва строки, с последующей еще одной кавычкой.

Когда изображение-строки должно быть размещено в исходном тексте программы на нескольких строчках, разрыв строки позволяет указывать количество пробелов в конце одной строчки и однозначно определяет положение продолжения изображения строки на следующей строчке.

3.2. Элементы прагматов

3.2.1. Последовательностью-элементов-ПОЯСНЕНИЙ-ОФОРМЛЕННЫХ (п.9.2.1.c) может служить любая последовательность литер (не обязательно абстрактных), в которую не входит последовательность (вместе с разделителями), представляющая сам символ-ПОЯСНЕНИЕ-ОФОРМЛЕННЫЙ (т.к. последним завершается прагмат). В реализации возможны, однако, дальнейшие ограничения на последовательность литер, допустимых в прагматах (но только не в примечаниях).

3.2.2. Предусмотрены шесть стандартных элементов-прагматов: СТРАНИЦА (PAGE), ТЧК (POINT), ВР (UPPER), РЕЗ (RES), ЗАПОМНИТЬ (PUSH), ВОССТАНОВИТЬ (POP) (В скобках английские эквиваленты элементов-прагматов). Эти элементы должны распознаваться хотя бы в их минимальной форме:

Символ-прагмат-ОФОРМЛЕННЫЙ,
элемент,
Символ-прагмат-ОФОРМЛЕННЫЙ.

Каждый из перечисленных элементов-прагматов записывается как последовательность букв, которым могут предшествовать или за которыми могут следовать особенности типографского набора. (Во всех выделяющих режимах символ-прагмат может быть записан как „ПРАГМ" с последующим разделителем).

3.2.3. *Новая страница*

3.2.3.1. При распечатке некоторого конструкта в конкретных литерах с помощью процессора Алгол 68 прагмат, содержащий элемент прагмата СТРАНИЦА (PAGE), указывает, что строчку, следующую за строчкой, содержащей замыкающий символ-прагмат, печатают с начала новой страницы. (Прагмат СТРАНИЦА не является, однако, особенностью типографского набора).

3.2.4. *Запоминание режима выделения*

3.2.4.1. Прагмат, содержащий элемент-прагмата ЗАПОМНИТЬ (PUSH), указывает, что значение действующего в данном месте прагмата, определяющего режим выделения (подраздел 3.4. настоящего приложения), запоминается для последующего восстановления.

3.2.4.2. Прагмат, содержащий элемент-прагмата ВОССТАНОВИТЬ (POP), связывается с последним из прагматов, содержащих элемент-прагмата ЗАПОМНИТЬ, еще не связанным с другим прагматом, содержащим элемент-прагмата ВОССТАНОВИТЬ (если такой есть), и восстанавливает действие того из прагматов, задающих режим выделения, который действовал перед этим прагматом.

3.3. Особенности типографского набора

3.3.1. Особенностями типографского набора являются пробел, новая строчка и разрыв строки. Новая строчка может быть одной конкретной литерой или физическим явлением, подобным концу записи. Разрыв строки используют только в изображениях-строки.

3.4. Слова и выделенные слова

3.4.1. Представление слов и выделенных слов определяют „режим выделения", Существует три режима выделения: выделение точкой, выделение прописными буквами, резервирование слов.

Новый режим вводится прагматом, содержащим один из элементов-прагмата ТЧК (POINT), ВР (UPPER), РЕЗ (RES) и начинает действовать сразу же после замыкающего символа-прагмат. Режим не действует на „оформление" представления (так, в режимах ВР и РЕЗ „ПРАГМ" соответствует „ПРАГМ"). Приводимые ниже правила требуют наличия разделителя в некоторой позиции. В качестве разделителей можно использовать особенности типографского набора. Слова различаются только тогда, когда различны конкатенации их подслов, например, „конец файла" может быть записан также как „коне цфайла".

3.4.1.1. В режиме выделения точкой (ТЧК) выделенные слова представляют следующим образом:

выделенные слова начинаются с точки (.), за которой следуют слоги, содержащие по порядку абстрактные буквы и цифры, соответствующие выделенным буквам и цифрам слова;

слоги должны разделяться литерами подчеркивания, но не особенностями типографского набора;

за выделенным словом должен следовать разделитель.

В режиме выделения точкой слова представляют следующим образом:

слово составляется из последовательности одного или более слогов, разделенных нулем или более особенностей типографского набора;

слог составляется из соответствующих, расположенных по порядку, абстрактных букв и цифр и может завершаться литерой подчеркивания;

если слог не завершается литерой подчеркивания, то после него должен следовать разделитель.

3.4.1.2. В режиме выделения прописными буквами (ВР) слова и выделенные слова представляют как в режиме ТЧК, но только с использованием дополнительных правил:

в выделенных словах не должно быть смешения прописных и строчных букв;

точка может быть опущена перед выделенным словом из прописных букв, если ему предшествует разделитель, отличный от точки, строчная буква или цифра, не являющаяся „прописной цифрой". „Прописная цифра" – это цифра, которой предшествует прописная буква или прописная цифра;

за выделенным словом из прописных букв не обязательно ставить разделитель, если за ним следует строчная буква;

прописные буквы могут быть использованы только в выделенных словах и в качестве элементов-основного-набора (подпункт 8.1.4.1.c).

3.4.1.3. В режиме резервирования слов (РЕЗ) слова и выделенные слова представляют как в режиме ТЧК, но с использованием дополнительных правил:

точка может опускаться перед зарезервированными словами, заданными в п. 9.4.1 как представления для символа языка, составленными из букв русского алфавита:

БИТ БЫВ В ВЕЩ ВИД ВСЕ ВЫБ ВЫХОД ГЛОБ ДЛИН ДЛЯ ДО
ЕСЛИ ЕСТЬ ИЗ ИМЕНИ ИМЯ ИНАЧЕ ИНЕС ИСТИНА КАНАЛ КОМПЛ КОН
КОНЕЦ КОР КЦ ЛИБО ЛИВЫБ ЛИТ ЛОГ ЛОЖЬ ЛОК НА НАЧ НАЧАЛО
НЕСТЬ НИЛ ОБ ОП ОТ ПАР ПОДВ ПОКА ПРАГМ ПРИМ ПРИО
ПРОПУСК ПРОЦ ПУСТ ПУСТОЕ С СЕМА СКИП СЛОГ СТ СТРУКТ
ТО Ф ФАЙЛ ФОРМАТ ЦЕЛ ЦК ЧЕРЕЗ ШАГ

и букв латинского алфавита:

AT BEGIN BITS BOOL BY BYTES CASE CHANNEL CHAR CO
COMMENT COMPL DO ELIF ELSE EMPTY END ESAC EXIT
FALSE FI FILE FLEX FOR FORMAT FROM GO GOTO HEAP
IF IN INT IS ISNT LOC LONG MODE NIL OD OF OP OUSE
OUT PAR PR PRAGMAT PRIO PROC REAL REF SEMA SHORP
SKIP STRING STRUCT THEN TO TRUE UNION VOID WHILE

если им предшествует разделитель, отличный от точки;

со слогом должен соприкасаться знак подчеркивания, если буквы и цифры этого слова соответствуют, в том же порядке, буквам и цифрам некоторого зарезервированного слова.

3.5. Составные представления

3.5.1. Некоторые представления, приведенные в п. 9.4.1, составленные из последовательности двух или более литер, не являющихся буквами (" ", =:, :=, |:, :=:, :/=:), являются последовательностями абстрактных литер, соответствующих этим символам.

3.5.2. Представление любого символа-ПОНЯТИЕ1-перед-ПОНЯТИЕМ2 является представлением для символа-ПОНЯТИЕ 1, за которым следует представление для символа-ПОНЯТИЕ 2, (символы-ПОНЯТИЕ 1-перед-ПОНЯТИЕМ 2 являются составными обозначениями-операций, упомянутыми в подпунктах 9.4.2.2. d, e).

3.6. Другие представления

3.6.1. Любой символ, представление которого в подразделе 9.4 соответствует абстрактной литере, представляется этой литерой. Символ-на-десять-в-степени, символ-плюс-и-на и символ-краткое-примечание не имеют представлений.

4. Обмен

4.1. Представление объектов для обмена должно использовать только абстрактные литеры (с тем, чтобы ввод можно было подготавливать, а вывод интерпретиро-

вать без ссылки на конкретную реализацию). Запросы к обстановке зависят от абстрактных литер следующим образом:

да	„T" латинская	или „Д" русская";
нет	„F" латинская	или „Н" русская;
литера ошибки	„*";	
пробел	„␣".	

4.2. Вместо абстрактных литер для символа-на-десять-в-степени и для символа-плюс-и-на необходимо использовать литеры „E" латинская или „Е" русская и „I" латинская или „И" русская.

Соответствующие друг другу прописные и строчные буквы эквивалентны, когда они входят при обмене в представление любого значения, отличного от значений видов „литерный" и „вектор из литерных".

4.3. Строковые значения, полученные в результате обмена и операции ПРЕД (REPR), могут содержать литеры, которые не соответствуют абстрактным литерам.

ПРИЛОЖЕНИЕ 3
Справочное

УКАЗАТЕЛЬ ПРИМЕНЯЕМЫХ В СТАНДАРТЕ ПОНЯТИЙ

1. Технические термины

Ниже приводятся указания на применяемые вхождения ряда слов, имеющих в настоящем стандарте специфическое техническое значение. Слова, встречающиеся в разных грамматических формах, даются только один раз, обычно в инфинитиве. Термины, используемые только в прагматических замечаниях, заключены в фигурные скобки.

абстракция (для протопонятий)	1.1.4.2.b	abstraction (protonotion of a protonotion)
активное (действие)	2.1.4.3.a	active (action)
альтернатива	1.1.3.2.c	alternative
апостроф	1.1.3.1.a	apostrophe
арифметическое значение	2.1.3.1.a	arithmetic value
большой синтаксический знак	1.1.3.1.a	large syntactic mark
вариант (значения)	4.4.2.c	variant (of a value)
вариант языка ALGOL 68	1.1.5.b	variant of ALGOL 68
{векторизация}	6	{rowing}
версия (обозначения-операции)	10.1.3.Шаг 3	version (of an operator)
верхняя граница	2.1.3.4.b	upper bound
вещественное число	2.1.3.1.a	real number
вид	2.1.1.2.b 2.1.5.f	mode
видимое	1.1.3.2.h	visible
в (конструкт в окружении)	2.1.5.b	in (a construct in an environ)
вместо (кортежа)	3.2.2.a 5.4.4.2	in place of
{внешний объект}	2.1.1	{external object}
вновь созданное (имя)	2.1.3.2.a	newly created (name)
{внутренний объект}	2.1.1	{internal object}

возобновить (действие)	2.1.4.3.g	resume (an action)
{временное имя}	2.1.3.6.c	{transient name}
в смысле численного анализа	2.1.3.1.e	sense of numerical analysis
{вставка}	10.3.4.1.1.ee	{insertion}
выбирать (массив по отрезку)	2.1.3.4.i	select (a trim selecting a multiple value)
выбирать (массив по 'СЛОВУ')	2.1.3.4.k	select (a 'TAG' selecting a multiple value)
выбирать (подымя по индексу)	2.1.3.4.g	select (an index selecting a subname)
выбирать (подымя по 'СЛОВУ')	2.1.3.3.e	select (a 'TAG' selecting a subname)
выбирать (поле по 'СЛОВУ')	2.1.3.3.a	select (a 'TAG' selecting a field)
выбирать (поле по указателю-поля)	2.1.5.g	select (a field-selector selecting a field)
выбирать (сцену из состава-ВЫБИРАЮЩЕГО-предложения)	3.4.2.b	chosen (scene of a chooser-CHOICE-clause)
выбирать (элемент по индексу)	2.1.3.4.a	select (an index selecting an element)
{выборка из массива}	5.3.1	{multiple selection}
выдача (сцены)	2.1.2.b 2.1.4.1.b 2.1.5.c, d	yield (of a scene)
вызвать (процедуру)	5.4.3.2.b	calling (of a routine)
{выписывать (вид)}	2.1.1.2	{spelling (of a mode)}
выполняться (для предикатов)	1.3.2.	hold (of a predicate)
вырожденный паспорт	2.1.3.4.c	flat descriptor
выстраивание	10.3.2.3.c	straightening
генерировать (имя по отрезку)	2.1.3.4.j	generate (a trim generating a name)
генерировать (имя по 'СЛОВУ')	2.1.3.4.l	generate (a 'TAG' generating a name)
гиперальтернатива	1.1.3.4.c	hyperalternative
гиперпонятие	1.1.3.1.e	hypernotion
гиперправило	1.1.3.4.b	hyper-rule
граница	2.1.3.4.b	bound
граничная пара	2.1.3.4.b	bound pair
двоеточие	1.1.3.1.a	colon
действие	2.1.4.1.a	action
деленное на (для арифметических значений)	2.1.3.1.e	divided by (of arithmetic values)
дерево порождения	1.1.3.2.f	production tree
дефис	1.1.3.1.a	hyphen
{динамический (повторитель)}	10.3.4.1.1.dd	{dynamic (replicator)}
естественное значение	8.1.1.2 8.1.2.2.a, b 8.1.4.2.b 8.2.2.b, c	intrinsic value
есть (является) (для гиперпонятий)	2.1.5.e	is (of hypernotions)
завершаться (о действии)	2.1.4.3.c, d	complete (an action)
{завести (файл на канале)}	10.3.1.4.cc	{establish (a file on a channel)}
{закрыть (файл)}	10.3.1.4.ff	{close (a file)}
заложение (гиперпонятие заложено в протопонятие)	1.1.4.1.c	envelop (a protonotion enveloping a hypernotion)
{запрос к обстановке}	10.2	{environment enquiry}
запускать (действие)	2.1.4.3.b, c	initiate (an action)
запятая	1.1.3.1.a	comma

звездочка	1.1.3.1.a	asterisk
звено	1.1.3.2.d	member
значение	2.1.1.1.a	value
{зона}	3.0.2	{reach}
идентифицировать (для индикаторов)	7.2.2.b	identify (an indicator identifying an indicator)
из (конструкт из конструкта)	2.1.5.a	of (construct of a construct)
из (конструкт из сцены)	2.1.1.1.d	of (construct of a scene)
из (окружение из сцены)	2.1.1.1.d	of (environ of a scene)
из (среда из конструкта)	3.0.2	of (nest of a construct)
именовать	2.1.2.e 2.1.3.2.a	refer to
иметь доступ (внутри участка)	2.1.2.c	access (inside a locale)
имя	2.1.3.2.a	name
имя подвижного (именующее массив)	2.1.3.4.f	flexible name (referring to a multiple value)
имя фиксированного (именующее массив)	2.1.3.4.f	fixed name (referring to a multiple value)
индекс (для выбора элемента)	2.1.3.4.a	index (to select an element)
исполнение	2.1.4.1.a	elaboration
исполнять совместно	2.1.4.2.f	elaborate collaterally
истинностное значение	2.1.3.1.f	truth value
{канал}	10.3.1.2	{channel}
{книга}	10.3.1.1	{book}
{кодирующая таблица}	10.3.1.2	{conversion key}
конструкт	1.1.3.2.e	construct
конструкт в языке представления	9.3.b	construct in a representation language
{крепкая (позиция)}	6.1.1	{firm (position)}
{крепко связаны}	7.1.1	{firmly related}
{куча}	5.2.3	{heap}
{ликвидировать (файл)}	10.3.1.4.hh	{scratch (a file)}
{литера}	2.1.3.1.g	character
{литерал}	10.3.4.1.1.ee	{literal}
{логический конец}	10.3.1.1.aa	{logical end}
{логический файл}	10.3.1.5.dd	{logical file}
локализующее окружение	5.2.3.2.b	local environ
малый синтаксический знак	1.1.3.1.a	small syntactic mark
{маркер}	10.3.4.1.1.cc	{marker}
массив	2.1.3.4.a	multiple value
меньше (наследник меньше, чем дерево порождения)	1.1.3.2.g	smaller (descendent smaller than a production tree)
меньше чем (для арифметических значений)	2.1.2.d 2.1.3.1.e	smaller than (of arithmetic values)
метапонятие	1.1.3.1.d	metanotion
метаправило	1.1.3.3.b	metaproduction rule
минус (для арифметических значений)	2.1.3.1.e	minus (of arithmetic values)
младше чем (область действия)	2.1.2.f	newer (of scopes)
{мягкая (позиция)}	6.1.1	{soft (position)}
{набор}	10.3.4.1.1.gg	{collection}
наддействие	2.1.4.2.c	direct parent
надъязык	2.2.2.c	superlanguage
наибольший целочисленный эквивалент (литеры)	2.1.3.1.g	largest integral equivalent (of a character)

наследник	1.1.3.2.g	descendent
наследное действие	2.1.4.2.b	descendent action
начать именовать (для имен)	2.1.3.2.a	make to refer to (of a name)
неактивное действие	2.1.4.3.a	inactive (action)
невидимое	1.1.3.2.h	invisible
неделимое действие	2.1.4.2.a	inseparable action
{независимость (свойств)}	7.1.1	{independence (of properties)}
незанятый участок	2.1.1.1.b	vacant locale
нелокализующее (окружение)	3.2.2.b	nonlocal
необходимое (окружение для сцены)	7.2.2.c	necessary for (an environ for a scene)
не определено	1.1.4.3.a	undefined
несовместимые действия	2.1.4.2.e	incompatible actions
нижняя граница	2.1.3.4.b	lower bound
область действия (значения)	2.1.1.3.a	scope (of a value)
область действия (окружения)	2.1.1.3.b	scope (of an environ)
{обмен}	10.3	{transput}
обобщать (целое число до вещественного)	2.1.2.d 2.1.3.1.e	widenable to (an integer to areal number)
{обобщение}	6	{widening}
обозначать (гиперпонятие обозначает протопонятие)	1.1.4.1.a	designate (a hypernotion designating a protonotion)
обозначать (парапонятие обозначает конструкт)	1.1.4.2.a	designate (a paranotion designating a construct)
{объединение}	6	{uniting}
объединять из (видов)	2.1.3.6.a	united from (of modes)
объект	2.1.1	object
{ожидать}	10.3.4.1.1.ll	{expect}
окружение	2.1.1.1.c	environ
{операция синхронизации}	10.2	{synchronization operation}
{*описание обмена*}	10.2	{*transput declaration*}
определяющий блок (для индикатора)	7.2.2.a	defining range (of an endicator)
опускаемые гиперпонятия	1.1.4.2.c	elidible hypernotion
{опустошение}	6	{voiding}
осмысленная программа	1.1.4.3.c	meaningful program
особенности типографского набора	9.4.d	typograohical display feature
{осуществить вставку}	10.3.4.1.1.ee	{perform an insertion}
{осуществить (размещение)}	10.3.4.1.1.ff	{perform (an alignment)}
{открыть (файл)}	10.3.1.4.dd	{open (a file)}
отрезок	2.1.3.4.h	trim
параллельное действие	10.2.4.	parallel action
парапонятие	1.1.4.2.a	paranotion
паспорт	2.1.3.4.b	descriptor
первичное окружение	2.2.2.a	primal environ
{перегруженное (обозначение-операции)}	4.5	{overload}
{перекрестные ссылки (в синтаксисе)}	1.1.3.4.f	{cross-reference (in the syntax)}
{переполнение}	2.1.4.3.h	{overflow}
{повторитель}	10.3.4.1.1.dd	{replicator}
{подавляемая рамка}	10.3.4.1.1.cc	{suppressed frame}
{подготовить (шаблон)}	10.3.4.1.1.dd	{staticize (a picture)}
{поддействие}	2.1.4.2.a	direct action
подобно (для деревьев порождения)	1.1.3.2.k	akin (a production tree to a production tree)

подъязык	2.2.2.c	sublanguage
подымя	2.1.2.g	subname
поле	2.1.3.3.a	field
получаться простой подстановкой	1.1.3.3.d	substitute simply
получаться согласованной подстановкой	1.1.3.4.e	substitute consistently
получать доступ (к значению внутри участка)	2.1.2.c	make to access (a value inside a locale)
понятие	1.1.3.1.c	notion
порождать	1.1.3.2.f	produce
порождающее правило	1.1.3.2.b	production rule
после (в текстуальном порядке)	1.1.3.2.i	after (in the textual order)
последовательное действие	2.1.4.2.a	serial action
{последовательный доступ}	10.3.1.3.ff	{sequential access}
постоянное соотношение	2.1.2.a	permanent relationship
построить (имя из имени)	6.6.2.c	built (the name built from a name)
построить (массив из значения)	6.6.2.b	built (the multiple value built from a value)
{правильность построения}	7.4	{well formed}
прагматическое замечание	1.1.2	pragmatic remark
предикат	1.3.2	predicate
предшествовать (в текстуальном порядке)	1.1.3.2.j	precede (in the textual order)
{предысполнение}	2.1.4.1.c	{pre-elaboration}
прежде (в текстуальном порядке)	1.1.3.2.i	before (in the textual order)
прекращать (действие)	2.1.4.3.e	terminate (an action)
прервать (действие)	2.1.4.3.h	interrupt (an action)
{приводимость}	6	{sort}
приемлемо для (значение приемлемо для вида)	2.1.3.6.d	acceptable to (a value acceptable to a mode)
приостанавливать (действие)	2.1.4.3.f	halt (an action)
приписать (значение или сцену индикатору)	4.8.2.a	ascribe (a value or scene to an indicator)
присваивать (значение имени)	5.2.1.2.b	assign (a value to a name)
программа в строгом языке	1.1.1.b 10.1.2	program in the strict language
{произвольный доступ}	10.3.1.3.ff	{random access}
прообраз	1.1.3.2.f	original
просматривать	10.3.2.3.d	traverse
простая подстановка	1.1.3.3.d	simple substitute
простое значение	2.1.3.1.a	plain value
протопонятие	1.1.3.1.b	protonotion
процедура	2.1.3.5.a	routine
{процедура обработки события}	10.3.1.3	{event routine}
{процедура реакции}	10.3.1.3	{on routine}
процесс	10.2.4.	process
прочий синтаксический знак	1.1.3.1.a	other syntactic mark
прямой наследник	1.1.3.2.f	direct descendent
псевдоимя	2.1.3.2.a	nil
псевдопримечание	10.1.3.Шаг 7	pseudo-comment
пустое значение	2.1.3.1.h	void malue
развертывать (сцену из описателя)	4.6.2.c	develop (a scene from a declarer)

размер (арифметического значения)	2.1.3.1.b	size (or an arithmetic value)
{размещение}	10.3.4.1.1.ff	{alignment}
{разыменование}	6	{dereferencing}
{рамка}	10.3.5.1.bb	{frame}
{раскрытая (позиция)}	6.1.1	{meek (position)}
распроцедуривание	6	{deproceduring}
реализация (Алгола 68)	2.2.2.c	implementation (of ALGOL 68)
реализация эталонного языка	9.3.c	implementation of the reference language
{редактировать (строку)}	10.3.4.1.1.jj	{edit (a string)}
{свойство}	2.1.1.1.b 3.0.2	{property}
{связать (книгу с файлом)}	10.3.1.4.bb	{link (a book with a file)}
семантика	1.1.1	semantics
{сжимаемо}	10.3.1.3.ff	{compressible}
{сильная (позиция)}	6.1.1	{strong (position)}
символ	1.1.3.1.f	symbol
синтаксис	1.1.1	syntax
скрытый элемент	2.1.3.4.c	ghost element
{слабая (позиция)}	6.1.1	{weak (position)}
следовать (в текстуальном порядке)	1.1.3.2.j	follow (in the textual order)
смысл	1.1.4 2.1.4.1.a	meaning
снабжать подыменами	2.1.3.3.e 2.1.3.4.g	endow with subnames
{снять файл}	10.3.1.4.gg	{lock (a file)}
{совместно при вводе}	10.3.4.1.1.ii	{input compatible}
{совместимо при выводе}	10.3.4.1.1.hh	{output compatible}
совместное действие	2.1.4.2.a	collateral action
совместное исполнение	2.1.4.2.f	collateral elaboration
согласованная подстановка	1.1.3.4.e	consistent substitute
содержать (для гиперпонятий)	1.1.4.1.b	comtain (by a hypernotion)
содержать (для деревьев порождения)	1.1.3.2.g	contain (by a production tree)
содержать (для протопонятий)	1.1.4.1.b	contain (by a protonotion)
{создать (файл на канале)}	10.3.1.4.cc	{create (a file on a channel)}
соотношение	2.1.2.a	relationship
{составлять (строку)}	10.3.4.1.1.kk	{indit (a string)}
составляющий	1.1.4.2.d	constituent
составное имя	2.1.3.2.b	stowed name
составное значение	2.1.1.1.a	stowed value
{состояние}	10.3.1.3	{state}
специфицировать (вид описателем)	4.6.2.d	specify (a declarer specifying a mode)
{список данных}	10.3.3	{data list}
справедливо (для соотношений)	2.1.2.a	hold (of a relationship)
среда	3.0.2	nest
{стандартная функция}	10.2	{standard function}
стандартная языковая обстановка	1.1.1	standard environment
{стандартное обозначение-операции}	10.2	{standard operator}
{стандартный вид}	10.2	{standard mode}
старше чем (область действия)	2.1.2.f	older (of scopes)

строгий язык	1.1.1.b 1.1.3.2.e 10.1.2	strict language
{строка}	8.3	{string}
структура	2.1.3.3.a	structured value
сцена	2.1.1.1.d	scene
такая же, как и (область действия)	2.1.2.f	same as (of scopes)
текстуальный порядок	1.1.3.2.i	textual order
терминальное метапорождение (метапонятия)	1.1.3.3.c	terminal metaproduction (of a metanotion)
терминальное порождение (дерева порождения)	1.1.3.2.f	terminal production (of a production tree)
терминальное порождение (понятия)	1.1.3.2.f	terminal production (of a notion)
точка	1.1.3.1.a	point
точка с запятой	1.1.3.1.a	semicolon
транзитивное соотношение	2.1.2.a	transitive relationship
трансформация	10.3.4.1.2.b	transform
{трафарет}	10.3.4.1.1.cc	{pattern}
требоваться	1.1.4.3.b	required
тупик	1.1.3.2.d	blind alley
удлинять (для арифметических значений)	2.1.2.d 2.1.3.1.e	lengthening (of arithmetic values)
умножить (арифметические значения)	2.1.3.1.e	times (of frithmetic values)
{управление (строкой посредством трафарета)}	10.3.4.1.1.dd	{control (a string by a pattern)}
{уравнивание}	3.4.1	{balancing}
устанавливать (окружение вокруг окружения)	3.2.2.b	establish (an environ around an environ)
участок	2.1.1.1.b	locale
{файл}	10.3.1.3	{file}
{физический файл}	10.3.1.5.cc	{phyisical file}
фиксировать (для видов)	2.1.3.6.b	deflex (a mode to a mode)
{формат}	10.3.4	{format}
целое число	2.1.3.1.a	integer
целочисленный эквивалент (литеры)	2.1.3.1.g	integral equivalent (of a character)
численный анализ в смысле	2.1.3.1.e	numerical analysis, in the sense of
число добавочных размеров	10.2.1.j, l	number of extra widths
число добавочных удлинений	2.1.3.1.d	number of extra lingths
число добавочных укорочений	10.2.1.j, l 2.1.3.1.d	number of extra shorths
{шаблон}	10.3.4.1.1.cc	{picture}
эквивалентность (видов)	2.1.1.2.a	equivalence (of modes)
эквивалентность (литеры и целого)	2.1.2.d 2.1.3.1.g	equivalence (of a character and an integer)
эквивалентность (протопонятий)	2.1.1.2.a	equivalence (of protonotions)
элемент (массива)	2.1.3.4.a	element (of a multiple value)
эталонный язык	9.3.a	reference language
этого (для конструктов)	2.1.5.a	of (construct of a construct)
этого (конструкт этой сцены)	2.1.1.1.d	of (construct of a scene)
этого (окружение этой сцены)	2.1.1.1.d	of (environ of a scene)
этого (среда этого конструкта)	3.0.2	of (next of a construct)

язык представления	9.3.a	representation language
язык публикаций	9.3.a	publication language
язык реализации	9.3.a	hardware language

2. Парапонятия

Ниже даны со ссылками на соответствующие гиперправила короткие парапонятия, представляющие понятия, применяемые в настоящем стандарте.

альтернативы-ВЫБИРАЮЩЕГО-предложения	3.4.1.d	alternate-CHOICE-clause
АРНЫЙ-операнд	5.4.2.1.c	ADIC-operand
БИНАРНАЯ-формула	5.4.2.1.a	DYADIC-formula
блок	3.0.1.f	range
вакуум	3.3.1.k	vacuum
вариантное-предложение	3.4.1.p	case-clause
верхняя граница	4.6.1.n	upper-bound
ВНЕШНЕЕ-вступление	10.1.1.c	EXTERNAL-prelude
возможное-ПОНЯТИЕ	1.3.3.a	NOTION-option
вставка	10.3.4.1.1.d	insertion
вступление	10.1.1.b	preludes
вход-в-собственно-ВЫБИРАЮЩЕЕ	9.1.1.b	CHOICE-in
выбирающее-предложение	3.4.1.n	choice-clause
ВЫБИРАЮЩЕЕ-предложение	3.4.1.a	CHOICE-clause
ВЫБИРАЮЩИЙ-вариант	3.4.1.i	case-part-of-CHOICE
выборка	5.3.1.1.a	selection
вызов	5.4.3.1.a	call
выражение	3.0.1.b	expression
вырезка	5.3.2.1.a	slice
выход-собственно-ВЫБИРАЮЩЕГО	9.1.1.d	CHOICE-out
вытесняющее-предложение	3.4.1.c	enquiry-clause
генератор	5.2.3.1.a	generator
главная-часть-ВЫБИРАЮЩЕГО	3.4.1.e	in-CHOICE-clause
грань	5.3.2.1.j	boundscript
групповое-определение-ОБЪЕКТОВ	4.1.1.b,c	COMMON-joined-definition
групповое-определение-меток	10.1.1.h	joined-label-definition
групповой-описатель-!ЗНАЧЕНИЙ	4.6.1.t, u	MOIDS-joined-declarer
групповой-описатель-!ПАРАМЕТРОВ	4.6.1.q, r	PARAMETERS-joined-declarer
ДВОИЧНАЯ-цифра	8.2.1.h,i,j,k	RADIX-digit
ДВОИЧНОЕ	8.2.1.d,e,g,f	RADIX
действительное-число	8.1.2.1.e	floating-point-numeral
десятичная-цифра	8.1.1.1.c	digit-cypher
диапазон-вектора	4.6.1.j,k,l	row-rower
диапазон-вектора-МАССИВОВ	4.6.1.i	row-ROWS-rower
добавочный-элемент-ПОЯСНЕНИЯ	9.2.1.d	other-PRAGMENT-item
добавочный-элемент-строки	8.1.4.1.d	other-string-item
доопределение	5.1.a	hip
дробная-часть	8.1.2.1.d	fractional-part

заглавие-цикла	3.5.1.b	for-part
задание-аргументов	5.4.1.1.e	declarative
задание-генератора	5.2.3.1.b	sample-generator
задание-дробной-части	10.3.4.10.1.c	after-specification
задание-значимости	10.3.4.10.1.b	width-specification
задание-порядка	10.3.4.10.1.d	exponent-specification
задачи	10.1.1.d	tasks
задача-пользователя	10.1.1.f	user-task
замкнутое-предложение	3.1.1.a	closed-clause
запись-десятичного-основания	8.1.2.1.h	times-ten-to-the-power-choice
запись-массива	3.3.1.l	row-display
запись-составного	3.3.1.j	display
запись-структуры	3.3.1.h	structure-display
знак	9.1.1.g	token
знак-ПОНЯТИЕ	9.1.1.f	NOTION-token
изображаемое	8.0.1.a	denoter
изображение	8.1.0.1.a 8.1.1.1.a 8.1.2.1.a 8.1.3.1.a 8.1.4.1.a 8.1.5.1.a 8.2.1.a,b,c 8.3.1.a	denotation
изображение-битового	8.2.1.1	bits-denotation
изображение-простого	8.1.0.1.b	plain-denotation
изображение-строки	8.3.1.c	string-denotation
индекс	5.3.2.1.e	subscript
индексатор	5.3.2.1.i	indexer
индексатор-МАССИВА-составляющий-?МАССИВ	5.3.2.1.b,c,d	ROWS-leaving-ROWSETY-indexer
индексирующее-ПОНЯТИЕ	1.3.3.e	NOTION-bracket
индикатор	4.8.1.e	indicator
ИНДИКАТОР	4.8.1.a,b	INDICATOR
источник	5.2.1.1.c	source
источник-для-ПРОВИДА	4.4.1.d	source-for-MODINE
код-размещения	10.3.4.1.1.f	alignment-code
константа	3.0.1.d	constant
кортеж	3.2.1.в	series
литерал	10.3.4.1.1.i	literal
мантисса	8.1.2.1.f	stagnant-part
маркер	10.3.4.1.1.n	marker
маркер-знака	10.3.4.2.1.e	sign-marker
маркер-комплексного	10.3.4.5.1.в	complex-marker
маркер-литеры	10.3.4.6.1.в	character-marker
маркер-логического	10.3.4.4.1.в	boolean-marker
маркер-нуля	10.3.4.2.1.d	zero-marker
маркер-основания	10.3.4.7.1.c	radix-marker
маркер-порядка	10.3.4.1.e	exponent-marker
маркер-точки	10.3.4.3.1.d	point-marker
маркер-цифры	10.3.4.2.1.f	digit-marker
машинная цифра	8.2.1.m	radix-digit
набор	10.3.4.1.1.в	collection
настройка-цикла	3.5.1.c	intervals
настройка-ЦИКЛА	3.5.1.d	FROBYT-part

натуральное-число	8.1.1.1.в	fixed-point-numeral
?НЕПОДАВЛЯЕМАЯ-рамка-ПУНКТА	10.3.4.1.1.k	UNSUPPRESETY-COMARK-frame
?НЕПОДАВЛЯЕМАЯ-рамка-ТОЧКИ	10.3.4.1.1.j	UNSUPPRESETY-MARK-frame
неподавляемое-подавляемое	10.3.4.1.1.l	unsuppressible-suppression
?НЕПОДАВЛЯЕМОЕ-подавление	10.3.4.1.1.l	UNSUPPRESETY-suppression
неподавляемый-литерал	10.3.4.1.1.i	unsuppressible-literal
?НЕПОДАВЛЯЕМЫЙ-литерал	10.3.4.1.1.i	UNSUPPRESSETY-literal
неподавляемый-повторитель	10.3.4.1.1.h	unsuppressible-replicator
нижняя-граница	4.6.1.m	lower-bound
обозначение-бинарной-операции	5.4.2.1.e	dyadic-operator
обозначение-унарной-операции	5.4.2.1.f	monadic-operator
образец-действительного	10.3.4.3.1.c	floating-point-mould
образец-знака	10.3.4.2.1.c	sign-mould
образец-рационального	10.3.4.3.1.b	variable-point-mould
образец-целого	10.3.4.2.1.b	integral-mould
образец-основ	3.3.1.b	joined-portrait
образ-!ПОЛЕЙ	3.3.1.f,g	FIELDS-portrait
образ-!ПОЛЕЙ1-среди-!ПОЛЕЙ	4.6.1.e	FIELDS-portrayer-of-FIELDS1
ограничение	5.3.2.1.h	trimscript
операнд	5.4.2.1.g	operand
оператор	3.0.1.c	statement
описание	4.1.1.a	declaration
описание-вида	4.2.1.a	mode-declaration
описание-идентификатора	4.4.1.g	identifier-declaration
описатель-НОМЕР	4.2.1.c	TALLY-declarer
описание-операции	4.5.1.a	operation-declaration
описание-переменной	4.4.1.e	variable-declaration
описание-приоритета	4.3.1.a	priority-declaration
описание-тождества	4.4.1.a	identity-declaration
описатель	4.2.1.c 4.4.1.b 4.6.1.a,b	declarer
описатель-процедуры	4.4.1.b	routine-declarer
определение	4.1.1.d	definion
определение-вида	4.2.1.b	mode-definition
определение-метки	3.2.1.c	label-definition
определение-операции	4.5.1.c	operation-definition
определение-параметра	5.4.1.1.f	parameter-definition
определение-переменной	4.4.1.f	variable-definition
определение-ПОЛЯ-среди-!ПОЛЕЙ	4.6.1.f	FIELDS-definition-of-FIELD
определение-приоритета	4.3.1.b	priority-definition
определение-тождества	4.4.1.c	identity-definition
определитель	4.6.1.c,d,g,h,o,s	declarator
определяющее-предложение	3.2.1.i	establishing-clause
основа	3.2.1.d	unit
основное-предложение	3.2.1.h	unitary-clause
отношение-одноименности	5.2.2.1.a	identity-relation
отрезок	5.3.2.1.f	trimmer
параллельное-предложение	3.3.1.c	parallel-clause
параметр	5.4.1.1.g 5.4.3.1.c	parameter

!ПАРАМЕТРЫ	5.4.3.1.b	PARAMETERS
переменная	3.0.1.e	variable
переход	5.4.4.1.a	jump
план	4.5.1.b 4.6.1.p	plan
план-процедуры	4.5.1.b	routine-plan
плюс-или-минус	8.1.2.1.j	plusminus
повторитель	10.3.4.1.1.g	replicator
подавление	10.3.4.1.1.l	suppression
подчиненный-условию-цикл	3.5.1.f	while-do-part
получатель	5.2.1.1.b	destination
порядок	8.1.2.1.g	exponent-part
последовательное-предложение	3.2.1.a	serial-clause
последовательность-ПОНЯТИЙ	1.3.3.b	NOTION-sequence
пояснение	9.2.1.a	pragment
ПОЯСНЕНИЕ	9.2.1.b	PRAGMENT
поясняемый-литерал	10.3.4.8.1.c	praglit
приведенная-ФОРМА	6.1.1.a,b,c,d,e	FORM-coercee
приведенное	6.1.1.g	coercee
приводимое	6.1.1.h	coercend
присваивание	5.2.1.1.a	assignation
программа	2.2.1.a	program
продолжатель-ВЫБИРАЮЩЕГО	9.1.1.c	CHOICE-again
пропуск	5.5.2.1.a	skip
псевдоимя	5.2.4.1.a	nihil
размещение	10.3.4.1.1.e	alignment
рамка	10.3.4.1.1.m	frame
рамка-ДВОИЧНОГО	10.3.4.7.1.b	RADIX-frame
рациональное-число	8.1.2.1.b	variable-point-numeral
сдвинутая-нижняя-граница	5.3.2.1.g	revised-lower-bound
символ	9.1.1.h	symbol
системная-задача	10.1.1.e	system-task
собственно-выбор-ВЫБИРАЮЩЕГО	3.4.1.f,g,h	in-part-of-CHOICE
собственно-заключение	10.1.1.i	particular-postlude
собственно-программа	10.1.1.g	particular-program
совместное-предложение	3.3.1.a,d,e	collateral-clause
сопоставляющее-предложение	3.4.1.g	conformity-clause
состав-ВЫБИРАЮЩЕГО-предложения	3.4.1.b	chooser-CHOICE-clause
спецификация	3.4.1.j,k	specification
список-ПОНЯТИЙ	1.3.3.c	NOTION-list
сравнитель-имен	5.2.2.1.b	identity-relator
старт-ВЫБИРАЮЩЕГО	9.1.1.a	CHOICE-start
степень-десяти	8.1.2.1.i	power-of-ten
строка	8.3.1.b	string
текст-программы	10.1.1.a	program-text
текст-процедуры	5.4.1.1.a,b	routine-text
текст-формата	10.3.4.1.1.a	format-text
тело-цикла	3.5.1.h	do-part
трафарет	10.3.4.1.1.o	pattern
трафарет-бесформатного	10.3.4.10.1.a	general-pattern
трафарет-битового	10.3.4.7.1.a	bits-pattern
трафарет-вещественного	10.3.4.3.1.a	real-pattern
трафарет-выбора-по-логическому	10.3.4.8.1.b	boolean-choice-pattern

трафарет-выбора-по-целому	10.3.4.8.1.a	integral-choice-pattern
трафарет-комплексного	10.3.4.5.1.a	complex-pattern
трафарет-логического	10.3.4.4.1.a	boolean-pattern
трафарет-строки	10.3.4.6.1.a	string-pattern
трафарет-формата	10.3.4.9.1.a	format-pattern
трафарет-целого	10.3.4.2.1.a	integral-pattern
указание	5.4.4.1.b	go-to
указатель-поля	4.8.1.f	field-selector
указатель-поля-вида-ПРИЗНАК-среди-!ПОЛЕЙ	4.8.1.c,d	QUALITY-FIELDS-field-selector
УНАРНАЯ-формула	5.4.2.1.b	MONADIC-formula
упакованное-?ПОНЯТИЕ	1.3.3.d	NOTETY-pack
условие-цикла	3.5.1.g	while-part
условное-предложение	3.4.1.o	conditional-clause
УТВЕРЖДЕНИЕ1-либо-УТВЕРЖДЕНИЕ2	1.3.3.f	THING1-of-alternatively THING2
финиш-ВЫБИРАЮЩЕГО	9.1.1.e	CHOICE-finish
ФОРМА-после-векторизации	6.6.1.a	rower-to-FORM
ФОРМА-после-мягкого распроцедуривания	6.3.1.b	softly-deprocedured-to-FORM
ФОРМА-после-обобщения	6.5.1.a,b,c,d	widened-to-FORM
ФОРМА-после-объединения	6.4.1.a	united-to-FORM
ФОРМА-после-опустошения	6.7.1.a,b	voided-to-FORM
ФОРМА-после-разыменования	6.2.1.a	dereferenced-to-FORM
ФОРМА-после-распроцедуривания	6.3.1.a	deprocedured-to-FORM
ФОРМА-после-сохранения	6.1.1.f	unchanged-from-FORM
формула	5.4.2.1.d	formula
фраза	3.0.1.a	phrase
целая-часть	8.1.2.1.c	integral-part
цикл	3.5.1.e	replating-part
циклическое-предложение	3.5.1.a	loop-clause
шаблон	10.3.4.1.1.c	picture
элемент-основного-набора	8.1.4.1.c	character-glyph
элемент-ПОЯСНЕНИЯ	9.2.1.c	PRAGMENT-item
элемент-строки	8.1.4.1.b	string-item
ядро	5.5.1.1.a	cast

3. Предикаты

Ниже даны сокращенные формы предикатов, применяемых в настоящем стандарте.

'входят в'	7.3.1.1.m,n	'subset of'
'выводится из'	5.3.1.1.b,c	'is derived from'
'есть'	1.3.1.g	'is'
'и'	1.3.1.c,e	'and'
'идентифицировано в'	7.2.1.a	'identified in'
'или'	1.3.1.d,f	'or'
'истина'	1.3.1.a	'true'
'крепко связано'	7.1.1.k	'firmly related'
'ложь'	1.3.1.b	'false'
'меняет'	7.4.1.a,b,c,d	'shields'
'может следовать'	3.4.1.m	'may follow'

'находиться в'	7.2.1.b,c	'resides in'
'начинаться с'	1.3.1.h,i,j	'begins with'
'не зависеть'	7.1.1.a,b,c,d	'independent'
'оказаться'	5.4.1.1.c,d	'like'
'равно числу'	7.3.1.o,p	'number equals'
'развертываться из'	7.3.1.c	'develops from'
'родственны'	4.7.1.f	'incestuous'
'сводиться'	7.1.1.n	'deprefs to firm'
'связано'	7.1.1.e,f,g,h,i,j	'related'
'скреплено'	7.1.1.1.m	'is firm'
'служит'	6.4.1.b	'unites to'
'совпадать с'	1.3.1.k,l	'coincides with'
'содержать'	1.3.1.m,n	'contains'
'сплетено с'	4.7.1.g	'revels to'
'считать'	4.3.1.c,d	'counts'
'уравнивать'	3.2.1.f,g	'balances'
'фиксировать'	4.7.1.a,b,c,d,e	'deflexes to'
'эквивалентно'	7.3.1.a,b,d,e,f, f,g,i,j,k,q	'equivalent'

4. Указатель к стандартному вступлению

<	10.2.3.0.a, 10.2.3.3.a, 10.2.3.4.c, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.9.a, 10.2.3.10.a,g, h
< =	10.2.3.0.a, 10.2.3.3.b, 10.2.3.4.b, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.8.e, 10.2.3.9.a, 10.2.3.10.b,g,h
+	10.2.3.0.a, 10.2.3.3.i,j, 10.2.3.4.i,j, 10.2.3.5.a,b, 10.2.3.6.b, 10.2.3.7.j, k,p,q,r,s, 10.2.3.10.i,j,k
+:=	10.2.3.0.a, 10.2.3.11.d,e,f,o,p,q,s
+=:	10.2.3.0.a, 10.2.3.11.r,t
+X	10.2.3.0.a, 10.2.3.3.u, 10.2.3.4.s, 10.2.3.5.e,f
+*	10.2.3.0.a, 10.2.3.3.u, 10.2.3.4.s, 10.2.3.5.e,f
&	10.2.3.0.a, 10.2.3.2.b, 10.2.3.8.d
∧	10.2.3.0.a, 10.2.3.2.b, 10.2.3.8.d
⊓	10.2.3.0.a, 10.2.3.8.k, 10.2.3.9.b
⌜	10.2.3.0.a, 10.2.3.1.c,e
↓	10.2.3.0.a, 10.2.3.8.h
⌞	10.2.3.0.a, 10.2.3.1.b,d, 10.2.3.4.r
≥	10.2.3.0.a, 10.2.3.3.e, 10.2.3.4.e, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.8.f, 10.2.3.9.a, 10.2.3.10.e,g,h
≤	10.2.3.0.a, 10.2.3.3.b, 10.2.3.4.b, 10.2.3.5.c,d 10.2.3.6.a, 10.2.3.8.e, 10.2.3.9.a, 10.2.3.10.b,g,h
≠	10.2.3.0.a, 10.2.3.2.e, 10.2.3.3.d, 10.2.3.4.d, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.7.g,u,v,w,x 10.2.3.8.b, 10.2.3.9.a, 10.2.3.10.d,g,h
∨	10.2.3.0.a, 10.2.3.2.a, 10.2.3.8.c
⊥	10.2.3.0.a, 10.2.3.3.u, 10.2.3.4.s, 10.2.3.5.e,f
÷	10.2.3.0.a, 10.2.3.3.m
÷X	10.2.3.0.a, 10.2.3.3.n
÷X:=	10.2.3.0.a, 10.2.3.11.k
÷*	10.2.3.0.a, 10.2.3.3.n
÷*:=	10.2.3.0.a, 10.2.3.11.k
÷:=	10.2.3.0.a, 10.2.3.11.j
X	10.2.3.0.a, 10.2.3.3.l, 10.2.3.4.l, 10.2.3.5.a,b, 10.2.3.7.l,p,q,r,s, 10.2.3.10.l,m,n,o

×:=	10.2.3.0.a, 10.2.3.11.g,h,i,n,o,p,u	
~	10.2.3.2.c, 10.2.3.8.m	
↑	10.2.3.0.a, 10.2.3.3.p, 10.2.3.5.g, 10.2.3.7.t, 10.2.3.8.g	
*	10.2.3.0.a, 10.2.3.3.l, 10.2.3.4.l, 10.2.3.5.a,b, 10.2.3.7.l,p,q,r,s 10.2.3.10.l,m,n,o	
**	10.2.3.0.a, 10.2.3.3.p, 10.2.3.5.g, 10.2.3.7.t	
*:=	10.2.3.0.a, 10.2.3.11.g,h,i,n,o,p,u	
¬	10.2.3.2.c, 10.2.3.8.m	
–	10.2.3.0.a, 10.2.3.3.g,h, 10.2.3.4.g,h, 10.2.3.5.a,b, 10.2.3.7.h,i,p,q,r,s	
–:=	10.2.3.0.a, 10.2.3.11.a,b,c,n,o,p	
/	10.2.3.0.a, 10.2.3.3.o, 10.2.3.4.m, 10.2.3.5.a, b, 10.2.3.7.m,p,q,r,s	
/:=	10.2.3.0.a, 10.2.3.11.l,m,n,o,p	
/=	10.2.3.0.a, 10.2.3.2.e, 10.2.3.3.3.d, 10.2.3.4.d, 10.2.3.5.d,d, 10.2.3.6.a, 10.2.3.7.g,u,v,w,x, 10.2.3.8.b, 10.2.3.9.a, 10.2.3.10.d,g,h	
%	10.2.3.0.a, 10.2.3.3.m	
%×	10.2.3.0.a, 10.2.3.3.n	
%×:=	10.2.3.0.a, 10.2.3.11.k	
%*	10.2.3.0.a, 10.2.3.3.n	
%*:=	10.2.3.0.a, 10.2.3.11.k	
%:=	10.2.3.0.a, 10.2.3.11.j	
>	10.2.3.0.a, 10.2.3.3.f, 10.2.3.4.f, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.9.a, 10.2.3.10.f,g,h	
>=	10.2.3.0.a, 10.2.3.3.e, 10.2.3.4.e, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.8.f, 10.2.3.9.a, 10.2.3.10.e,g,h	
=	10.2.3.0.a, 10.2.3.2.d, 10.2.3.3.c, 10.2.3.4.c, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.7.f,u,b,w,x, 10.2.3.8.a, 10.2.3.9.a, 10.2.3.10.c,g,h	
абс	10.2.1.n, 10.2.3.f, 10.2.3.3.k, 10.2.3.4.к, 10.2.3.7.c, 10.2.3.8.i	abs
арг	10.2.3.7.d	arg
бин	10.2.3.8.j	bin
бит	10.2.2.g	bits
бш	10.2.3.0.a, 10.2.3.3.f, 10.2.3.4.f 10.2.3.5.c,d 10.2.3.6.a, 10.2.3.9.a, 10.2.3.10.f,g,h	gt
вверх	10.2.3.0.a, 10.2.3.3.p, 10.2.3.5.g, 10.2.3.7.t, 10.2.3.8.g, 10.2.4.e	up
вегр	10.2.3.0.a, 10.2.3.1.c,e	upb
вещ	10.2.2.d	real
вниз	10.2.3.0.a, 10.2.3.8.h, 10.2.4.d	down
вч	10.2.3.7.a	re
делпр	10.2.3.0.a, 10.2.3.11.l,m,n,o,p	divab
знак	10.2.3.3.t, 10.2.3.4.q	sign
и	10.2.3.0.a, 10.2.3.2.b, 10.2.3.8.d,	and
или	10.2.3.0.a, 10.2.3.2.a, 10.2.3.8.c	or
им	10.2.3.0.a, 10.2.3.3.u, 10.2.3.4.s, 10.2.3.5.e,f	i
канал	10.3.1.2.a	channel
компл	10.2.2.f	compl
лев	10.2.3.0.a, 10.2.3.8.g	shl
лит	10.2.2.e	char
лог	10.2.2.b	bool
минпр	10.2.3.0.a, 10.2.3.11.a,b,c,n,o,p	minusab
мод	10.2.3.0.a, 10.2.3.3.n	mod

модпр	10.2.3.0.a, 10.2.3.11.k	modab
мч	10.2.3.7.b	im
мш	10.2.3.0.a, 10.2.3.3.a, 10.2.3.4.a, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.9.a, 10.2.3.10.a,g,h	lt
нб	10.2.3.0.a, 10.2.3.3.b, 10.2.3.4.b, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.8.e, 10.2.3.9.a, 10.2.3.10.b,g,h	le
не	10.2.3.2.c, 10.2.3.8.m	not
нигр	10.2.3.0.a, 10.2.3.1.b,d	lwb
нм	10.2.3.0.a, 10.2.3.3.e, 10.2.3.4.e, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.8.f, 10.2.3.9.a, 10.2.3.10.e,g,h	ge
нр	10.2.3.0.a, 10.2.3.2.e, 10.2.3.3.d, 10.2.3.4.d, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.7.g,u,v,w,x, 10.2.3.8.b, 10.2.3.9.a, 10.2.3.10.d,g,h	ne
нчт	10.2.3.5.s	odd
окр	10.2.3.4.p	round
плюспр	10.2.3.0.a, 10.2.3.11.d,e,f,n,o,p,q,s	plusab
прав	10.2.3.0.a, 10.2.3.8.h	shr
пред	10.2.1.o	repr
прип	10.2.3.0.a, 10.2.3.11.r,t	plusto
пуст	10.2.2.a	void
рв	10.2.3.0.a, 10.2.3.2.d, 10.2.3.3.c, 10.2.3.4.c, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.7.f,u,v,w,x, 10.2.3.8.a, 10.2.3.9.a, 10.2.3.10.c,g;h	eq
сема	10.2.4.a	sema
слог	10.2.2.h	bytes
сопряж	10.2.3.7.e	conj
строк	10.2.2.i	string
удл	10.2.3.3.q, 10.2.3.4.n, 10.2.3.7.n, 10.2.3.8.n, 10.2.3.9.d	leng
укр	10.2.3.3.r, 10.2.3.4.o, 10.2.3.7.o, 10.2.3.8.o, 10.2.3.9.e	shorten
умпр	10.2.3.0.a, 10.2.3.11.g,h,i,n,o,p,u	timesab
уст	10.2.4.b,c	level
файл	10.3.1.3.a	file
формат фор	10.3.5.a	format
цед	10.2.3.0.a, 10.2.3.3.m	over
цел	10.2.2.c	int
цедпр	10.2.3.0.a, 10.2.3.11.j	overab
целч	10.2.3.4.r	entier
элем	10.2.3.0.a, 10.2.3.8.k, 10.2.3.9.b	elem
abs	10.2.1.n, 10.2.3.2.f, 10.2.3.3.k, 10.2.3.4.k, 10.2.3.7.c, 10.2.3.8.i	абс
and	10.2.3.0.a, 10.2.3.2.b, 10.2.3.8.d	и
arg	10.2.3.7.d	арг
bin	10.2.3.8.j	бин
bits	10.2.2.g	бит
bool	10.2.2.b	лог
bytes	10.2.2.h	слог
channel	10.3.1.2.a	канал
char	10.2.2.e	лит

compl	10.2.2.f	компл
conj	10.2.3.7.e	сопрж
divab	10.2.3.0.a, 10.2.3.11.l,m,n,o,p	делпр
down	10.2.3.0.a, 10.2.3.8.h, 10.2.4.d	вниз
elem	10.2.3.0.a, 10.2.3.8.k, 10.2.3.9.b	элем
entier	10.2.3.4.r	целч
eq	10.2.3.0.a, 10.2.3.2.d, 10.2.3.3.c, 10.2.3.4.c, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.7.f,u,v,w,x, 10.2.3.8.a, 10.2.3.9.a, 10.2.3.10.c,g,h	рв
file	10.3.1.3.a	файл
format	10.3.5.a	формат фор
ge	10.2.3.0.a, 10.2.3.3.e, 10.2.3.4.e, 10.2.3.5.c,d 10.2.3.6.a, 10.2.3.8.f, 10.2.3.9.a, 10.2.3.10.e,g,h	нм
gt	10.2.3.0.a, 10.2.3.3.f, 10.2.3.4.f, 10.2.3.5.e,d, 10.2.3.6.a, 10.2.3.9.a, 10.2.3.10.f,g,h	бш
i	10.2.3.0.a, 10.2.3.3.u, 10.2.3.4.s 10.2.3.5.e,f	им
im	10.2.3.7.b	мч
int	10.2.2.c	цел
le	10.2.3.0.a, 10.2.3.3.b, 10.2.3.4.b, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.8.e, 10.2.3.9.a, 10.2.3.10.b,g,h	нб
leng	10.2.3.3.q, 10.2.3.4.n, 10.2.3.7.n, 10.2.3.8.n, 10.2.3.9.d	удл
level	10.2.4.b,c	уст
lt	10.2.3.0.a, 10.2.3.3.a, 10.2.3.4.a, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.9.a, 10.2.3.10.a,g,h	мш
lwb	10.2.3.0.a, 10.2.3.1.b,d	нигр
minusab	10.2.3.0.a, 10.2.3.11.a,b,c,n,o,p	минпр
mod	10.2.3.0.a, 10.2.3.3.n	мод
modab	10.2.3.0.a, 10.2.3.11.k	модпр
ne	10.2.3.0.a, 10.2.3.2.e, 10.2.3.3.d, 10.2.3.4.d, 10.2.3.5.c,d, 10.2.3.6.a, 10.2.3.7.g,u,v,w,x, 10.2.3.8.b, 10.2.3.9.a, 10.2.3.10.d,g,h	нр
not	10.2.3.2.c, 10.2.3.8.m	не
odd	10.2.3.3.s	нчт
or	10.2.3.0.a, 10.2.3.2.a, 10.2.3.8.c	или
over	10.2.3.0.a, 10.2.3.3.m	цед
overab	10.2.3.0.a, 10.2.3.11.j	цедпр
plusab	10.2.3.0.a, 10.2.3.11.d,e,f,n,o,q,s	плюспр
plusto	10.2.3.0.a, 10.2.3.11.r,t	прип
re	10.2.3.7.a	вч
real	10.2.2.d	вещ
repr	10.2.1.o	пред
round	10.2.3.4.p	окр
sema	10.2.4.a	сема
shl	10.2.3.0.a, 10.2.3.8.g	лев
shorten	10.2.3.3.r, 10.2.3.4.o, 10.2.3.7.o, 10.2.3.8.o, 10.2.3.9.e	укр
shr	10.2.3.0.a, 10.2.3.8.h	прав
sign	10.2.3.3.t, 10.2.3.4.q	знак
string	10.2.2.i	строк
timesab	10.2.3.0.a, 10.2.3.11.g,h,i,n,o,p,u	умпр
up	10.2.3.0.a, 10.2.3.3.p, 10.2.3.5.g, 10.2.3.7.t, 10.2.3.8.g, 10.2.4.e	вверх
upb	10.2.3.0.a, 10.2.3.1.c,e	вегр

void 10.2.2.a		пуст
арккос	10.2.3.12.f	arccos
арксин	10.2.3.12.h	arcsin
арктанг	10.2.3.12.j	arctan
бит пак	10.2.3.8.l	bits pack
ввод	10.3.3.2.a	get
возм ввод	10.3.1.3.b	get possible
возм уст нач	10.3.1.3.f	reset possible
возм вывод	10.3.1.3.c	put possible
возм двоичн	10.3.1.3.d	bin possible
возм переобозначение	10.3.1.3.h	reidf possible
возм установка	10.3.1.3.g	get possible
вперед	10.3.1.6.a	space
вывод	10.3.3.1.a	put
да	10.2.1.r	flip
дв ввод	10.3.6.2.a	get bin
дв вывод	10.3.6.1.a	put bin
дв зап	10.5.1.h	write bin
дв чит	10.5.1.i	read bin
завести	10.3.1.4.b	establish
задать код	10.3.1.3.j	make conv
задать стопетроку	10.3.1.3.k	make term
закрыть	10.3.1.4.n	close
зап	10.5.1.d	write
заполнитель	10.2.1.q	null character
канал	10.3.1.3.i	chan
корень	10.2.3.12.b	sqrt
кос	10.2.3.12.e	cos
лг	10.2.3 12.d	ln
литера в строке	10.3.2.1.l	char in string
литера ошибки	10.2.1.t	errorchar
макс вещ	10.2.1.f	max real
макс лит	10.2.1.p	max abs char
макс цел	10.2.1.c	max int
можно завести	10.3.1.2.c	estab possille
назад	10.3.1.6.b	backspace
нет	10.2.1.s	flop
нов страница	10.3.1.6.d	newpage
нов строчка	10.3.1.6.c	newline
номер литеры	10.3.1.5.a	char number
номер страницы	10.3.1.5.c	page number
номер строчки	10.3.1.5.b	line number
открыть	10.3.1.4.d	open
переобозначение	10.3.1.3.s	reidf
печ	10.5.1.d	print
пи	10.2.3.12.a	pi
плав	10.3.2.1.d	float
пред псч	10.5.1.a	last random
при конце лог файла	10.3.1.3.l	on logical file end
при конце страницы	10.3.1.3.n	on page end
при конце строчки	10.3.1.3.o	on line end
при конце физ файла	10.3.1.3.m	on physical file end
при конце формата	10.3.1.3.p	on format end
при ошибке значения	10.3.1.3.q	on value error
при ошибке литеры	10.3.1.3.r	on char error

пробел	10.2.1.u	blank
псч	10.5.1.b	random
размер бит	10.2.1.j	bits width
размер слог	10.2.1.m	bytes width
разрядность вещ	10.3.2.1.n	real width
разрядность порядка	10.3.2.1.o	exp width
разрядность цел	10.3.2.1.m	int width
сжимаем	10.3.1.3.e	compressible
син	10.2.3.12.g	sin
след псч	10.2.3.12.k	next random
слог пак	10.2.3.9.c	bytes pack
снять	10.3.1.4.o	lock
соединить	10.3.1.4.e	associate
создать	10.3.1.4.c	create
стандввод	10.5.1.c	stand in
стандвывод	10.5.1.c	stand out
стандканал ввода	10.3.1.2.e	stand in channel
стандканал вывода	10.3.1.2.f	stand out channel
стандканал обмена	10.3.1.2.g	stand back channel
стандкод	10.3.1.2.d	standconv
стандобмен	10.5.1.c	stand back
стереть	10.3.1.4.p	scretch
стоп	10.5.2.a	stop
танг	10.2.3.12.i	tan
точность вещ	10.2.1.g	small real
установить	10.3.1.6.i	set
уст нач	10.3.1.6.j	reset
уст номер литеры	10.3.1.6.k	set char number
фикс	10.3.2.1.c	fixed
ф ввод	10.3.5.2.a	getf
ф вывод	10.3.5.1.a	putf
ф зап	10.5.1.f	writef
ф печ	10.5.1.f	printf
ф чит	10.5.1.g	readf
целое	10.3.2.1.b	whole
число длин бит	10.2.1.h	bits lengths
число длин вещ	10.2.1.d	real lengths
число длин слог	10.2.1.k	bytes lengths
число длин цел	10.2.1.a	int lengths
число кор бит	10.2.1.i	bits shorths
число кор вещ	10.2.1.e	real shorths
число кор слог	10.2.1.l	bytes shorths
число кор цел	10.2.1.b	int shorths
чит	10.5.1.e	real
эксп	10.2.3.12.c	exp
Д арккос	10.2.3.12.f	L arccos
Д арксин	10.2.3.12.h	L arcsin
Д арктанг	10.2.3.12.j	L arctan
Д бит	10.2.2.g	L bits
Д бит пак	10.2.3.8.l	L bits pack
Д вещ	10.2.2.d	L eal
Д компл	10.2.2.f	L compl
Д корень	10.2.3.12.b	L sqrt
Д кос	10.2.3.12.e	L cos
Д лг	10.2.3.12.d	L ln

Д макс вещ	10.2.1.f	L max real
Д макс цел	10.2.1.c	L max int
Д пи	10.2.3.12.a	L pi
Д пред псч	10.5.1.a	L last random
Д псч	10.5.1.b	L random
Д размер бит	10.2.1.j	L bits width
Д размер слог	10.2.1.m	L bytes width
Д разрядность вещ	10.3.2.1.n	L real width
Д разрядность порядка	10.3.2.1.o	L exp width
Д разрядность цел	10.3.2.1.m	L int width
Д син	10.2.3.12.g	L sin
Д след псч	10.2.3.12.k	L next random
Д слог	10.2.2.h	L bytes
Д слог пак	10.2.3.9.c	L bytes pack
Д танг	10.2.3.12.i	L tan
Д точность вещ	10.2.1.g	L small real
Д цел	10.2.2.c	L int
Д эксп	10.2.3.12.c	L exp
arccos	10.2.3.12.f	арккос
arcsin	10.2.3.12.h	арксин
arctan	10.2.3.12.j	арктанг
associate	10.3.1.4 e	соединить
backspace	10.3.1.6.b	назад
bin possible	10.3.1.3.d	возм двоичн
bits lengths	10.2.1.h	число длин бит
bits pack	10.2.3.8.l	бит пак
bits shorths	10.2.1.i	число кор бит
bits width	10.2.1.j	размер бит
blank	10.2.1.u	пробел
bytes lengths	10.2.1.k	число длин слог
bytes pack	10.2.3.9.c	слог пак
bytes shorths	10.2.1.l	число кор слог
bytes width	10.2.1.m	размер слог
chan	10.3.1.3.i	канал
char in string	10.3.2.1.l	литера в строке
char number	10.3.1.5.a	номер литеры
close	10.3.1.4.n	закрыть
compressible	10.3.1.3.e	сжимаем
cos	10.2.3.12.e	кос
create	10.3.1.4.c	создать
errorchar	10.2.1.t	литера ошибки
establish	10.3.1.4.b	завести
estab possible	10.3.1.2.c	можно завести
exp	10.2.3.12.c	эксп
exp width	10.3.2.1.o	разрядность порядка
fixed	10.3.2.1.c	фикс
flip	10.2.1.r	да
float	10.3.2.1.d	плав
flop	10.2.1.s	нет
get	10.3.3.2.a	ввод
get bin	10.3.6.2.a	дв ввод
getf	10.3.5.2.a	ф ввод
get possible	10.3.1.3.b	возм ввод
int lengths	10.3.1.2.a	число длин цел
int shorths	10.2.1.b	число кор цел

int width	10.3.2.1.m	разрядность цел
last random	10.5.1.a	пред псч
line number	10.3.1.5.b	номер строчки
ln	10.2.3.12.d	лг
lock	10.3.1.4.o	снять
make conv	10.3.1.3.j	задать код
make term	10.3.1.3.k	задать стопстроку
max abs char	10.2.1.p	макс лит
max int	10.2.1.c	макс цел
max real	10.2.1.f	макс вещ
newline	10.3.1.6.c	нов строчка
newpage	10.3.1.6.d	нов страница
next random	10.2.3.12.k	след псч
null character	10.2.1.q	заполнитель
on char error	10.3.1.3.r	при ошибке литеры
on format end	10.3.1.3.p	при конце формата
on line end	10.3.1.3.o	при конце строчки
on logical file end	10.3.1.3.l	при конце лог файла
on page end	10.3.1.3.n	при конце страницы
on physical file end	10.3.1.3.m	при конце физ файла
on value error	10.3.1.3.q	при ошибке значения
open	10.3.1.4.d	открыть
page number	10.3.1.5.c	номер страницы
pi	10.2.3.12.a	пи
print	10.5.1.d	печ
printf	10.5.1.f	ф печ
put	10.3.3.1.a	вывод
put bin	10.3.6.1.a	дв вывод
putf	10.3.5.1.a	ф вывод
put possible	10.3.1.3.c	возм вывод
random	10.5.1.b	псч
read	10.5.1.e	чит
read bin	10.5.1.i	дв чит
sin	10.2.3.12.g	син
readf	10.5.1.g	ф чит
real lengths	10.2.1.d	число длин вещ
real shorths	10.2.1.e	число кор вещ
real width	10.3.2.1.n	разрядность вещ
reidf	10.3.1.3.s	переобозначение
reidf possible	10.3.1.3.h	возм переобозначение
reset	10.3.1.6.j	уст нач
reset possible	10.3.1.3.f	возм уст нач
scratch	10.3.1.4.p	стереть
set	10.3.1.6.i	установить
set char number	10.3.1.6.k	уст номер литеры
set possible	10.3.1.3.g	возм установка
small real	10.2.1.g	точность вещ
space	10.3.1.6.a	вперед
sqrt	10.2.3.12.b	корень
stand back	10.5.1.c	станд обмен
stand back channel	10.3.1.2.g	станд канал обмена
stand conv	10.3.1.2.d	станд код
stand in	10.5.1.c	станд ввод
stand in channel	10.3.1.2.e	станд канал ввода
stand out	10.5.1.c	станд вывод

stand out channel	10.3.1.2.f	станд канал вывода
stop	10.5.2.a	стоп
tan	10.2.3.12.i	танг
whole	10.3.2.1.b	целое
write	10.5.1.d	зап
write bin	10.5.1.h	дв зап
writef	10.5.1.f	ф зап
L arccos	10.2.3.12.f	Д арккос
L arcsin	10.2.3.12.h	Д арксин
L arctan	10.2.3.12.j	Д арктанг
L bits	10.2.2.g	Д бит
L bits pack	10.2.3.8.l	Д бит пак
L bits width	10.2.1.j	Д размер бит
L bytes	10.2.2.h	Д слог
L bytes pack	10.2.3.9.c	Д слог пак
L bytes width	10.2.1.m	Д размер слог
L compl	10.2.2.f	аналог д компл
L cos	10.2.3.12.e	Д кос
L exp	10.2.3.12.c	Д эксп
L exp width	10.3.2.1.o	Д разрядность порядка
L int	10.2.2.c	Д цел
L int width	10.3.2.1.m	Д разрядность цел
L last random	10.5.1.a	Д пред псч
L ln	10.2.3.12.d	Д лг
L max int	10.2.1.c	Д макс цел
L max real	10.2.1.f	Д макс вещ
L next random	10.2.3.12.k	Д след псч
L pi	10.2.3.12.a	Д пи
L random	10.5.1.b	Д псч
L real	10.2.2.d	Д вещ
L real width	10.3.2.1.n	Д разрядность вещ
L sin	10.2.3.12.g	Д син
L small real	10.2.1.g	Д точность вещ
L sqrt	10.2.3.12.b	Д корень
L tan	10.2.3.12.i	Д танг
♀вводимое	10.3.2.2.d	♀ intype
♀вне	10.3.1.1.d	♀beyond
♀вставка	10.3.5.a	♀insertion
♀выводимое	10.3.2.2.b	♀outtype
♀кадр	10.3.5.a	♀piece
♀книга	10.3.1.1.a	♀book
♀код	10.3.1.2.b	♀conv
♀массив	10.2.3.1.a	♀rows
♀набор	10.3.5.a	♀collection
♀пакет	10.3.5.a	♀collitem
♀подвставка	10.3.5.c	♀sinsert
♀подвтекст	10.3.1.1.b	♀flextext
♀подрамк	10.3.5.e	♀sframe
♀позиция	10.3.1.1.c	♀pos
♀проввод	10.3.2.2.c	♀simplin
♀провывод	10.3.2.2.a	♀simplout
♀рамка	10.3.5.a	♀framee
♀связка	10.3.1.1.e	♀bfile
♀стройввод	10.3.2.3.b	♀straightin
♀стройвывод	10.3.2.3.a	♀straightout

⍰текст	10.3.1.1.b	⍰text
⍰трабесф	10.3.5.a	⍰gpattern
⍰травыб	10.3.5.a	⍰cpattern
⍰трафарет	10.3.5.a	⍰pattern
⍰трафор	10.3.5.a	⍰fpattern
⍰число	10.3.2.1.a	⍰number
⍰шаблон	10.3.5.a	⍰picture
⍰ввести вставку	10.3.5.h	⍰get insertion
⍰ввести литеру	10.3.3.2.b	⍰get char
⍰в двоичное	10.3.6.a	⍰to bin
⍰взять след шаблон	10.3.5.b	⍰get next picture
⍰вывести вставку	10.3.5.g	⍰put insertion
⍰вывести литеру	10.3.3.1.b	⍰ put char
⍰выполнить трафор	10.3.5.j	⍰do fpattern
⍰границы книги	10.3.1.5.e	⍰book bounds
⍰доступные книги	10.3.1.1.f	⍰chainbfile
⍰гномы	10.4.1.a	⍰ gremlins
⍰защита связей	10.3.1.1.h	⍰bfileprotect
⍰из двоичного	10.3.6.h	⍰from bin
⍰литеру в цифру	10.3.2.1.k	⍰char dig
⍰лог файл окончен	10.3.1.5.i	⍰logical file ended
⍰ложь	10.3.1.4.i	⍰false
⍰настроить	10.3.1.6.h	⍰set mood
⍰настроить на двоичное	10.3.1.4.m	⍰set bin mood
⍰настроить на запись	10.3.1.4.j	⍰set write mood
⍰настроить на литерное	10.3.1.4.i	⍰set char mood
⍰настроить на чтение	10.3.1.4.k	⍰set read mood
⍰не определено	10.3.1.4.a	⍰undefined
⍰нормализовать	10.3.2.1.g	⍰standardize
⍰Д нормализовать	10.3.2.1.g	⍰L standardize
⍰обозн приемлемо	10.3.1.4.g	⍰idf ok
⍰подготовить вставку	10.3.5.d	⍰staticize insertion
⍰подготовить рамки	10.3.5.f	⍰staticize frames
⍰предст рационального	10.3.2.1.f	⍰subfixed
⍰предст целого	10.3.2.1.e	⍰subwhole
⍰присоединить формат	10.3.5.k	⍰associate format
⍰проверить позицию	10.3.3.2.c	⍰check pos
⍰разместить	10.3.5.i	⍰alignment
⍰ред строку	10.3.5.1.b	⍰edit string
⍰след позиция	10.3.3.1.c	⍰next pos
⍰снятые книги	10.3.1.1.g	⍰lockedbfile
⍰соответствует	10.3.1.4.h	⍰match
⍰сост строку	10.3.5.2.b	⍰indit string
⍰страница хороша	10.3.1.6.f	⍰get good page
⍰строку в Д вещ	10.3.2.1.j	⍰string to L real
⍰строку в Д цел	10.3.2.1.i	⍰string to L int
⍰строчка окончена	10.3.1.5.f	⍰line ended
⍰строчка хороша	10.3.1.6.e	⍰get good line
⍰текущая позиция	10.3.1.5.d	⍰current pos
⍰файл доступен	10.3.1.4.f	⍰file available
⍰файл хорош	10.3.1.6.g	⍰get good file
⍰цифру в литеру	10.3.2.1.h	⍰dig char
⍰страница окончена	10.3.1.5.g	⍰page ended
⍰физ файл окончен	10.3.1.5.h	⍰physical file ended

ПРИЛОЖЕНИЕ 4
Справочное

СПИСОК МЕТАПРАВИЛ

АЛФАВИТ {942L ABC} ::
а; б; в; г; д; е; ж; з; и; й; к; л; м; н; о; п; р; с; т; у; ф; х; ц; ч; ш; щ; ъ; ь; э; ю; я; а лат; бе лат; це лат; де лат; е лат; эф лат; ге лат; аш лат; и лат; йот лат; ка лат; эль лат; эм лат; эн лат; о лат; пэ лат; ку лат; эр лат; эс лат; тэ лат; у лат; ве лат; дубль ве лат; икс лат; игрек лат; зэт лат.

АРНОЕ {542C ADIC} :: БИНАРНОЕ; УНАРНОЕ.

АФФИКС {45B TAO} :: ИНФИКС; ПРЕФИКС.

БИНАРНОЕ {542A DYADIC} :: приоритета ПРИОРИТЕТ.

БИТОВОЕ {65A BITS} ::структура содержащая ?МЕРНУЮ букву алеф для выборки вектора из логических в себе.

БУКВА {942B LETTER} ::буква АЛФАВИТА; буква алеф; буква АЛФАВИТА стиля НОМЕР.

ВАРИАНТНОЕ {34B CASE} ::выбирающее по целому; выбирающее по ПРЕДСТАВИТЕЛЮ.

ВИД {12A MODE} ::ПРОСТОЕ; СОСТАВНОЕ; ИМЯ ВИДА; ПРОЦЕДУРА; ПРЕДСТАВИТЕЛЬ; ЦИ определение ВИДА; использование ЦИ.

ВНЕШНЕЕ {A1A EXTERNAL} :: стандартное; библиотечное; системное; собственное.

ВСТАВКА {A341E INSERTION} :: вектор из структур содержащих букву п букву о букву в букву т для выборки процедуры вырабатывающей целое букву с букву т букву р для выборки объединения вектора из литерных литерного воедино в себе.

ВТОРИЧНОЕ {5C SECONDARY} :: приведенная выборка; приведенный ЛОКАЛИЗУЮЩИЙ генератор; ПЕРВИЧНОЕ.

ВЫБИРАЮЩЕЕ {34A CHOICE} ::выбирающее по логическому; ВАРИАНТНОЕ.

ДВОИЧНОЕ {82A RADIX} ::двоичное; четверичное; восьмеричное; шестнадцатеричное.

ДВУМЕСТНАЯ {123H DUO} ::процедура с ПАРАМЕТРОМ1 ПАРАМЕТРОМ2 вырабатывающая ЗНАЧЕНИЕ.

ДЕЙСТВУЮЩЕЕ {122B SOME} ::в СРЕДЕ ПРИВОДИМО выдающее ЗНАЧЕНИЕ.

?ДЛИННОЕ {12E LONGSETY} ::длинное ?ДЛИННОЕ; ПУСТО.

ДОЛГОЕ {65D LENGTH} ::буква д буква л буква и буква н.

?ДОЛГОЕ {65F LENGTHETY} ::ДОЛГОЕ ?ДОЛГОЕ; ПУСТО.

ЕСЛИ {13E WHETHER} ::если; если неверно что.

ЗАКРЫТОЕ {122A ENCLOSED} ::замкнутое; совместное; параллельное; ВЫБИРАЮЩЕЕ; циклическое.

ЗНАК {942G DYAD} ::ПРЕФИКСНЫЙ; НЕПРЕФИКСНЫЙ.

ЗНАКОВЫЙ {942M* DOP} ::ЗНАК; ЗНАК перед НЕПРЕФИКСНЫМ.

ЗНАЧЕНИЕ {12R MOID} ::ВИД; пустое значение.

!ЗНАЧЕНИЯ {46C MOIDS} ::ЗНАЧЕНИЕ; !ЗНАЧЕНИЯ ЗНАЧЕНИЕ.

?ЗНАЧЕНИЯ {47C MOIDSETY} ::!ЗНАЧЕНИЯ; ПУСТО.

ИМЯ {12M REF} ::имя; временное имя.

?ИМЯ {531A REFETY} ::ИМЯ; ПУСТО.

ИНДИКАНТ {942D TAB} ::выделенное СЛОВО; ?РАЗМЕРНОЕ СТАНДАРТНОЕ.

ИНДИКАТОР {48A INDICATOR} ::идентификатор; индикатор вида; обозначение операции.

ИНФИКС {942F TAD} ::выделенное СЛОВО: ЗНАК ?ПРИСВОЕНИЕ; ЗНАК перед НЕПРЕФИКСНЫМ ?ПРИСВОЕНИЕ.

КАДР {A341B PIECE} :: структура содержащая букву у букву т букву н для выборки целого букву с букву ч букву е букву т для выборки целого букву о букву у букву к для выборки целого букву н для выборки вектора из НАБОРОВ в себе.
КОРЕНЬ {71B NONPREF} :: ПРОСТОЕ; СОСТАВНОЕ; ПРЕДСТАВИТЕЛЬ; пустое значение; процедура с !ПАРАМЕТРАМИ вырабатывающая ЗНАЧЕНИЕ.
?КОРОТКОЕ {12F SHORTSETY} :: короткое ?КОРОТКОЕ; ПУСТО.
КРАТКОЕ {65E SHORTH} :: буква к буква о буква р.
КРАТКОЕ {65G SHORTHETY} :: КРАТКОЕ ?КРАТКОЕ; ПУСТО.
ЛИТЕРА {13B ALPHA} :: а; б; в; г; д; е; ж; з; и; й; к; л; м; н; о; п; р; с; т; у; ф; х; ц; ч; ш; щ; ъ; ы; ь; э; ю; я; I.
ЛОКАЛИЗУЮЩИЙ {44B LEAP} :: локальный; глобальный; первичный.
ЛЮБОЙ {46A VICTAL} :: НЕФОРМАЛЬНЫЙ; формальный.
МАССИВ {12L ROWS} :: вектор; МАССИВ векторов.
?МАССИВ {532A ROWSETY} :: МАССИВ; ПУСТО.
?МЕРНОЕ {65C SITHETY} :: ДОЛГОЕ ?ДОЛГОЕ; КРАТКОЕ ?КРАТКОЕ; ПУСТО.
МЕТКА {122K LAB} :: СЛОВО для метки.
!МЕТКИ {123J LABS} :: МЕТКА; ?МЕТКИ МЕТКА.
?МЕТКИ {123I LABSETY} :: !МЕТКИ; ПУСТО.
НАБОР {A341C COLLCTION} :: объединение ШАБЛОНА ПАКЕТА воедино.
?НЕПОДАВЛЯЕМОЕ {A341O UNSUPPREETY} :: неподавляемое; ПУСТО.
НЕПРЕФИКСНЫЙ {942I NOMAD} :: меньше; больше; разделить; равно; умножить; звездочка.
НЕРАСП{РОЦЕДУРИВАЕМОЕ 67A NONPROC} :: ПРОСТОЕ; СОСТАВНОЕ; ПРЕДСТАВИТЕЛЬ, ИМЯ НЕРАСП; ПРОЦЕДУРА С !ПАРАМЕТРАМИ вырабатывающая ЗНАЧЕНИЕ.
НЕСОСТАВНОЕ {47A NONSTOWED} :: ПРОСТОЕ; ИМЯ ВИДА; ПРОЦЕДУРА; ПРЕДСТАВИТЕЛЬ; пустое значение.
НЕФОРМАЛЬНЫЙ {46B VIRACT} :: виртуальный; фактический.
НОМЕР {12W TALLY} :: I· НОМЕР I.
?НОМЕР {542D TALLETY} :: НОМЕР; ПУСТО.
ОБОЗНАЧЕНИЕ {48G TAX} :: СЛОВО; ИНДИКАНТ; ИНФИКС; ПРЕФИКС.
ОБЪЕКТ {41A COMMON} :: вид; приоритет; тождество для ПРОВИДА; переменная как имя ПРОВИДА; операция как ПРОВИД; ПАРАМЕТР; поле вида ВИД среди !ПОЛЕЙ.
ОБЫЧНОЕ {12U MOOD} :: ПРОСТОЕ; СОСТАВНОЕ; имя ВИДА; ПРОЦЕДУРА; пустое значение.
!ОБЫЧНЫЕ {12T MOODS} :: ОБЫЧНОЕ; !ОБЫЧНЫЕ ОБЫЧНОЕ.
?ОБЫЧНЫЕ {47B MOODSETY} :: !ОБЫЧНЫЕ; ПУСТО.
ОДНОМЕСТНАЯ {123G MONO} :: процедура с ПАРАМЕТРОМ вырабатывающая ЗНАЧЕНИЕ.
ОПЕРАЦИЯ {45A PRAM} :: ДВУМЕСТНАЯ; ОДНОМЕСТНАЯ.
ОПИСАНИЕ {123E DEC} :: СЛОВО для ВИДА; ИНФИКС для приоритера ПРИОРИТЕТ; ИНДИКАНТ для ЗНАЧЕНИЯ НОМЕР; ИНФИКС для ДВУМЕСТНОЙ; ПРЕФИКС для ОДНОМЕСТНОЙ.
!ОПИСАНИЯ {123D DECS} :: ОПИСАНИЕ; !ОПИСАНИЯ ОПИСАНИЕ.
?ОПИСАНИЯ {123C DECSETY} :: !ОПИСАНИЯ; ПУСТО.
ОСНОВА {5A UNIT} :: приведенное присваивание; приведенное отношение одноименности; приведенный текст процедуры; переход; пропуск; ТРЕТИЧНОЕ.
ОФОРМЛЕННОЕ {133A STYLE} :: краткое; выделенное; стиля НОМЕР.
ПАКЕТ {A341D COLLITEM} :: структура содержащая букву в цифру один для выборки ВСТАВКИ букву п букву о букву в букву т для выборки процедуры вырабатывающей целое букву у букву д букву к для выборки целого букву в цифру два для выборки ВСТАВКИ в себе.
ПАРА {48E PROP} :: ОПИСАНИЕ; МЕТКА; ПОЛЕ.
!ПАРЫ {48D PROPS} :: ПАРА; !ПАРЫ ПАРА.

?ПАРЫ {48C PROPSETY} :: !ПАРЫ; ПУСТО.
ПАРАМЕТР {12Q PARAMETER} :: параметр вида ВИД.
?ПАРАМЕТРИЗОВАННАЯ {12D PARAMETY} :: с !ПАРАМЕТРАМИ; ПУСТО.
!ПАРАМЕТРЫ {12P PARAMETERS} :: ПАРАМЕТР; !ПАРАМЕТРЫ ПАРАМЕТР.
ПЕРВИЧНОЕ {5D PRIMATY} :: *приведенная вырезка: приведенный вызов*; приведенное изображаемое; приведенное ядро; приведенный текст формата; приведенный использующий СЛОВО идентификатор; ЗАКРЫТОЕ предложение.
?ПОДВИЖНОЕ {12K FLEXETY} :: подвижное; ПУСТО.
ПОЛЕ {12J FIELD} :: СЛОВО для выборки ВИДА.
!ПОЛЯ {12I FIELDS} :: ПОЛЕ. !ПОЛЯ ПОЛЕ.
ПОНЯТИЕ {13A NOTION} :: ЛИТЕРА; ПОНЯТИЕ ЛИТЕРА.
?ПОНЯТИЕ {13C NOTETY} :: ПОНЯТИЕ; ПУСТО.
ПОЯСНЕНИЕ {92A PRAGMENT} :: прагмат; примечание.
ПРЕДСТАВИТЕЛЬ {12S UNITED} :: объединение !ОБЫЧНЫХ воедино.
ПРЕФИКС {942K TAM} :: выделенное СЛОВО; ПРЕФИКСНЫЙ ?ПРИСВОЕНИЕ; ПРЕФИКСНЫЙ перед НЕПРЕФИКСНЫМ ПРИСВОЕНИЕ.
ПРЕФИКСНЫЙ {942H MONAD} :: или; и; амперсенд; не равно; не больше; не меньше; от до; процент; элемент; меньшее целое; большее целое; плюс и на; не; тильда; вниз; вверх; плюс; минус; префиксный стиля НОМЕР.
ПРИВОДИМО {122C SORT} :: сильно; крепко; раскрыто; слабо; мягко.
ПРИВОДИМОЕ {31A SOID} :: ПРИВОДИМО выдающее ЗНАЧЕНИЕ.
ПРИЗНАК {48F QUALITY} :: ВИД; ЗНАЧЕНИЕ НОМЕР; БИНАРНОЕ; метка; выборка ВИДА.
ПРИМЕНЯЮЩИЙ {48B DEFIED} :: определяющий; использующий.
ПРИОРИТЕТ {123F PRIO} :: I; II; III; III I; III II; III III; III III I; III III II; III III III.
?ПРИСВОЕНИЕ {942J BECOMESETY} :: перед присвоить; перед присвоить направо; ПУСТО.
ПРИСТАВКА {71A PREF} :: процедура вырабатывающая; ИМЯ.
?ПРИСТАВКИ {71C* PREFSETY} :: ПРИСТАВКА ?ПРИСТАВКИ; ПУСТО.
ПРОВИД {44A MODINE} :: процедура; ВИД.
ПРОЛОГ {73B HEAD} :: ПРОСТОЕ; ПРИСТАВКА; структура содержащая: ?ПОДВИЖНЫЙ МАССИВ из; процедура с; объединение; пустое значение.
ПРОСТОЕ {12B PLAIN} :: ЧИСЛОВОЕ; логическое; литерное.
ПРОЦЕДУРА {12N PROCEDURE} :: процедура ?ПАРАМЕТРИЗОВАННАЯ вырабатывающая ЗНАЧЕНИЕ.
ПРЯМОЕ {61G COMORF} :: присваивание в СРЕДЕ; ядро в СРЕДЕ; отношение одноименности в СРЕДЕ; изображаемое в СРЕДЕ; ЛОКАЛИЗУЮЩИЙ генератор в СРЕДЕ; текст формата в СРЕДЕ.
ПУНКТ {A341N COMARK} :: нуль; цифра; литера.
ПУСТО {12G EMPTY} :: .
РАЗМЕРНОЕ {810A SIZF} :: длинное; короткое.
?РАЗМЕРНОЕ {12D SIZETY} :: длинное ?ДЛИННОЕ; короткое ?КОРОТКОЕ; ПУСТО.
РАМКА {A341H FRAMF} :: структура содержащая букву в для выборки ВСТАВКИ букву п букву о букву в букву т для выборки процедуры вырабатывающей целое букву п букву о букву д букву а букву в для выборки логического букву м букву а букву р букву к для выборки литерного в себе.
РАСКРЫВАЕМОЕ {61F MORF} :: выборка в СРЕДЕ; вырезка в СРЕДЕ; вызов в СРЕДЕ; текст процедуры в СРЕДЕ; АРНАЯ формула в СРЕДЕ; использующий СЛОВО идентификатор в СРЕДЕ.
РАСКРЫТИЕ {61C MEEK} :: сохранение; разыменование; распроцедуривание.
СЛОВО {942A TAG} :: БУКВА; СЛОВО БУКВА; СЛОВО ЦИФРА.
СЛОГОВОЕ {65B BYTES} :: структура содержащая ?МЕРНУЮ букву алеф для выборки вектора из литерных в себе.
СЛОЙ {123B LAYER} :: новые ?ОПИСАНИЯ ?МЕТКИ.

СМЯГЧЕНИЕ {610D SOFT} :: сохранение; мягкое распроцедуривание.

СОСТАВНОЕ {12H STOWED} :: структура содержащая !ПОЛЯ в себе; ?ПОДВИЖНЫЙ МАССИВ из ВИДА.

СРЕДА {123A NEST} :: СЛОЙ; СРЕДА со СЛОЕМ.

?ССЫЛКА НА {531B REFLEXETY} :: ИМЯ; ИМЯ подвижного; ПУСТО.

СТАНДАРТНОЕ {942E STANDARD} :: целое; вещественное; логическое; литерное; формат; пустое значение; комплексное; комплексное лат; битовое; слоговое; строковое; сема; файл; канал.

СТОП {A1B STOP} :: буква эс лат буква тэ лат буква о лат буква пэ лат для метки буква с буква т буква о буква п для метки.

ТИП {A341P TYPE} :: целое; вещественное; логическое; комплексное; строковое; битовое; целого выбора; логического выбора; форматное; бесформатное.

ТОЧКА {A341M MARK} :: знак, точка; показатель; комплексное; логическое.

ТРАБЕСФ {ТРАБЕСФ БЕСФОРМАТНОГО A341K GPATTERN} :: структура содержащая букву в для выборки ВСТАВКИ букву с букву п букву е букву ц для выборки вектора из процедур вырабатывающих целое в себе.

ТРАВЫБ {ТРАФАРЕТ ВЫБОРА A341I CPATTERN} :: структура содержащая букву в для выборки ВСТАВКИ букву т букву и букву п для выборки целого букву с букву т букву р для выборки вектора из ВСТАВОК в себе.

ТРАФАРЕТ {A341G PATTERN} :: структура содержащая букву т букву и букву п для выборки целого букву р букву а букву м букву к букву и для выборки вектора из РАМОК в себе.

ТРАФОР {ТРАФАРЕТ ФОРМАТНОГО A341J FPATTERN} :: структура содержащая букву в для выборки ВСТАВКИ букву п букву р букву ф для выборки процедуры вырабатывающей ФОРМАТ в себе.

ТРЕТИЧНОЕ {5B TERTIARY} :: псевдоимя; приведенная АРНАЯ формула; ВТОРИЧНОЕ.

УКРЕПЛЕНИЕ {61B FIRM} :: РАСКРЫТИЕ; объединение.

УКРЫТИЕ {73A SAFE} :: укрытое; ЦИ помнит ВИД УКРЫТИЕ; инь УКРЫТИЕ; ян УКРЫТИЕ; запомненные ЗНАЧЕНИЕ1 ЗНАЧЕНИЕ2 УКРЫТИЕ.

УНАРНОЕ {542B MONADIC} :: приоритета III III III I.

УПАКОВКА {31B PACK} :: упакованное ОФОРМЛЕННОЕ.

УСИЛЕНИЕ {61A STRONG} :: УКРЕПЛЕНИЕ; обобщение; векторизация; опустошение.

УТВЕРЖДЕНИЕ {13D THING} :: ПОНЯТИЕ; (?ПОНЯТИЕ1) ?ПОНЯТИЕ2; УТВЕРЖДЕНИЕ (?ПОНЯТИЕ1) ?ПОНЯТИЕ2.

ФОРМА {61E FORM} :: РАСКРЫВАЕМОЕ; ПРЯМОЕ.

ФОРМАТ {A341A FORMAT} :: структура содержащая букву алеф для выборки вектора из КАДРОВ в себе.

ФОРПАТ {A341L FIVMAT} :: ци I определение структуры содержащей букву алеф для выборки вектора из структур содержащих букву у букву т букву н для выборки целого букву с букву ч букву е букву т для выборки целого букву о букву у букву к для выборки целого букву н для выборки вектора из объединения структуры содержащей букву т букву р букву а букву ф для выборки объединения ТРАФАРЕТА ТРАВЫБА структуры содержащей букву в для выборки ВСТАВКИ букву п букву р букву ф для выборки процедуры вырабатывающей использование ци I в себе ТРАБЕСФОРА пустого значения воедино букву в для выборки ВСТАВКИ в себе ПАКЕТА воедино в себе в себе.

ЦИ {12U MU} :: ци НОМЕР.

ЦИКЛ {35A FROBYT} :: нижний предел; шаг; верхний предел.

ЦИФРА {942C DIGIT} :: цифра нуль; цифра один; цифра два; цифра три; цифра четыре; цифра пять; цифра шесть; цифра семь; цифра восемь; цифра девять.

ЧАСТЬ {73E PART} :: ПОЛЕ; ПАРАМЕТР.
!ЧАСТИ {73 D PARTS} :: ЧАСТЬ; !ЧАСТИ ЧАСТЬ.
ЧИСЛО {810B* NUMERAL} :: натуральное число; рациональное число; действительное число.
ЧИСЛОВОЕ {12C INTREAL} :: ?РАЗМЕРНОЕ целое; ?РАЗМЕРНОЕ вещественное
ШАБЛОН {A341F PICTURE} :: структура содержащая букву т букву р букву а букву ф для выборки объединения ТРАФАРЕТА ТРАВЫБА ТРАФОРА ТРАБЕСФОРА *пустого значения воедино букву в для выборки* ВСТАВКИ в себе.
ЭПИЛОГ {73C TAILETY} :: ЗНАЧЕНИЕ; !ПОЛЯ в себе; !ПАРАМЕТРАМИ вырабатывающая ЗНАЧЕНИЕ; !ОБЫЧНЫХ воедино; ПУСТО.

ИНФОРМАЦИОННЫЕ ДАННЫЕ

1. ИСПОЛНИТЕЛИ

В.П.Морозов, д-р. техн. наук, профессор, **Г.С.Цейтин,** д-р. физ.-мат. наук, **А.Ф.Рар, А.Н.Терехов,** канд. физ.-мат. наук, **О.Е.Климова, В.В.Броль, В.Б.Яковлев, Н.Б.Скачков, И.Б.Гиндыш, Э.В.Оленьева, Д.Ю.Жуков, Ю.И.Карпов, Ф.Х.Кабалина, В.В.Зинкова, Н.И.Егерева.**

2. УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ постановлением Государственного комитета СССР по стандартам от 21.12.88 № 4380

3. Срок проверки – 1996 г., периодичность проверки – 5 лет.

4. ВВЕДЕН ВПЕРВЫЕ

5. *ССЫЛОЧНЫЕ НОРМАТИВНО-ТЕХНИЧЕСКИЕ ДОКУМЕНТЫ*

Обозначение НТД, *на который дана ссылка*	Номер приложения
27465–87	2

gost.e-ecolog.ru