УДК 681.3.06:006.354 Группа П85

ГОСУДАРСТВЕННЫЙ СТАНДАРТ СОЮЗА ССР

ЯЗЫК ПРОГРАММИРОВАНИЯ
Алгол 68 расширенный

ГОСТ
27975–88

Programming language ALGOL 68,
extended

ОКСТУ 4002

Дата введения 01.07.90

Настоящий стандарт распространяется на язык программирования Алгол 68 расширенный*, его варианты, а также варианты языка программирования Алгол 68, вводящие дополнительно к определению языка программирования Алгол 68 средства обеспечения модульности программ и раздельной трансляции программ, и устанавливает требования:

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

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

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

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

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

Издание официальное ★ Перепечатка воспрещена

gost.e-ecolog.ru

Требования к машинному представлению программы приведены в приложении 2.

1. ЯЗЫК И МЕТАЯЗЫК

Все прагматические замечания соответствуют ГОСТ 27974.

1.1. Метод описания

1.1.1. Введение соответствует ГОСТ 27974.

1.1.2. Прагматика соответствует ГОСТ 27974.

1.1.3. Синтаксис строгого языка соответствует ГОСТ 27974.

1.1.4. *Семантика*

Определение семантики соответствует ГОСТ 27974.

1.1.4.1. Гиперпонятия, обозначение и заложение соответствуют ГОСТ 27974.

1.1.4.2. Парапонятия.

Гиперправила a, b, d соответствуют ГОСТ 27974.

c) В правиле b) „опускаемые гиперпонятия" следующие:
„ОФОРМЛЕННОЕ" • „НОМЕР" • „ЛОКАЛИЗУЮЩИЙ" •
„ПРИМЕНЯЮЩИЙ" • „ЛЮБОЙ" • „ПРИВОДИМО" •
„ЗНАЧЕНИЕ" • „для ЗНАЧЕНИЯ" • „выдающее ИМЯ ПРОВИДА"
• „для метки" • „для процедуры" • „вида ПРОВИД"
• „с видом ПРОЦЕДУРА" • „в СРЕДЕ" • „!ПАРЫ" • „с ?ОПИСАНИЯМИ
?МЕТКАМИ" • „через ?ОПИСАНИЯ ?МЕТКИ" • „определяющее
СЛОЙ" • „ОБОЗНАЧЕНИЕ" • „как ИМЯ ПРОВИДА" • „без
?ОПИСАНИЙ" • „передающий ?СВЯЗИ" • „открывающий ?СВЯЗИ".

1.1.4.3. Неопределенности соответствуют ГОСТ 27974.

1.1.4.4. Восстанавливающие действия:

Для некоторых случаев, где говорится, что исполнение не определено {1.1.4.3 a, b}, заданы восстанавливающие действия. Это значит, что должно выполняться такое восстанавливающее действие, если только реализатор не предусмотрел более подходящего решения для данной ситуации. Однако реализатор должен сохранить для программиста возможность потребовать, чтобы выполняемым действием было в точности действие, указанное здесь.

{Обычно восстанавливающее действие включает в себя возбуждение соответствующей ситуации.}

b) Восстанавливающее действие состоит в вызове некоторой процедуры, возможно, со значениями {параметров}. Эта процедура задается при помощи представления некоторого использующего-идектификатора, выдающего эту процедуру в окружении собственного-вступления.

1.2. Общие метаправила

1.2.1. Метаправила для видов соответствуют ГОСТ 27974.

1.2.2. *Метаправила, связанные с фразами и приведением*

A) ЗАКРЫТОЕ: : замкнутое; совместное;
параллельное; ВЫБИРАЮЩЕЕ {34A}; циклическое;
подключающее.

Метаправила B, C соответствуют ГОСТ 27974.

1.2.3. *Метаправила, связанные со средами*

Метаправила, A, C, D, F, G, H, I, J, K соответствуют ГОСТ 27974.

B) СЛОЙ : : новые ?ОПИСАНИЯ ?МЕТКИ ?ПОДКЛЮЧЕНИЯ.

E) ОПИСАНИЕ : : СЛОВО {942A} для ВИДА;
ИНФИКС {942F} для приоритета ПРИОРИТЕТ;
ИНДИКАНТ {942F} для ЗНАЧЕНИЯ НОМЕР;
ИНФИКС {942F} для ДВУХМЕСТНОЙ;
ПРЕФИКС {942K} для ОДНОМЕСТНОЙ;
МОДУЛЬ; СЛОВО для СИТУАЦИИ с видом ПРОЦЕДУРА.

L) ?МОДУЛИ : : !МОДУЛИ; ПУСТО.

M) !МОДУЛИ : : МОДУЛЬ; !МОДУЛИ МОДУЛЬ.

N) МОДУЛЬ : : ИНДИКАНТ для !СВЯЗЕЙ с модулями.

O) ?СВЯЗИ : !СВЯЗИ; ПУСТО.

P) !СВЯЗИ : : СВЯЗЬ; !СВЯЗИ СВЯЗЬ.

Q) СВЯЗЬ : : КЛЮЧ для открытия ?ОПИСАНИЙ {и} !ПОДКЛЮЧЕНИЙ.

R) КЛЮЧ : : ЦИ.

S) ?ПОДКЛЮЧЕНИЯ : : !ПОДКЛЮЧЕНИЯ; ПУСТО.

T) !ПОДКЛЮЧЕНИЯ : : ПОДКЛЮЧЕНИЕ; !ПОДКЛЮЧЕНИЯ ПОДКЛЮЧЕНИЕ.

U) ПОДКЛЮЧЕНИЕ : : КЛЮЧ для запуска.

V) СИТУАЦИЯ : : ситуация; реакция

1.3. Общие гиперправила соответствуют ГОСТ 27974.

2. ВЫЧИСЛИТЕЛЬ И ПРОГРАММА

2.1. Терминология

2.1.1. *Объекты.*

Определение объекта и прагматическое замечание соответствуют ГОСТ 27974.

2.1.1.1 Значения, участки, окружения и сцены.

Гиперправила a, c, d соответствуют ГОСТ 27974.

b) Всякий „участок" {есть внутренний объект, который} соответствует каким-то '?ОПИСАНИЯМ ?МЕТКАМ ?ПОДКЛЮЧЕНИЯМ' {1.2.3 C, 1}. „Незанятый участок" – это участок, для которого '?ОПИСАНИЯ ?МЕТКИ ?ПОДКЛЮЧЕНИЯ' есть' ПУСТО'

{Каждое 'ОБОЗНАЧЕНИЕ для ПРИЗНАКА' (4.8.1. F, G), заложенное в данные '?ОПИСАНИЯ ?МЕТКИ ?ПОДКЛЮЧЕНИЯ', 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 „получать 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.}

2.1.1.2. Виды определены в ГОСТ 27974.

2.1.1.3. Области действия.

Прагматические замечания и гиперправило a) соответствуют ГОСТ 27974.

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

2.1.2. *Соотношения*

Гиперправила a, b, c, d, e, f, g соответствуют ГОСТ 27974.

h) „Реагировать" есть соотношение между значением {процедурой} и сценой {определением-ситуации}, которое может быть справедливым „внутри" определенного участка. Данное соотношение становится справедливым после исполнения определения-реакции.

i) Окружение может быть „связано" с другим окружением {со старшей областью действия} „посредством" некоторой сцены {определения-ситуации}. Это соотношение может быть справедливым для некоторого окружения, созданного в процессе исполнения вызова-ситуации.

2.1.3. Значения определены в ГОСТ 27974.

2.1.4. *Действия*

2.1.4.1. Исполнение соответствует ГОСТ 27974.

2.1.4.2. Последовательные и совместные действия соответствуют ГОСТ 27974.

2.1.4.3. Запуск, завершение и прекращение

Правила a, b, c, d, e, f, g и прагматическое замечание соответствуют ГОСТ 27974.

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. Для некоторых событий определены восстанавливающие действия (1.1.4.4.).}

i) Действие может прерваться, если вычислитель обнаруживает, что время (место в памяти), выделенное для исполнения программы, близко к исчерпанию. В таком случае восстанавливающим действием служит вызов процедуры восстановление после исчерпания времени {10.2.5.p} (восстановление после исчерпания памяти {10.2.5.q}). {Предполагается, что остающееся количество времени (памяти) будет достаточным, чтобы восстанавливающее действие обеспечило аккуратное завершение или же добыто дополнительные ресурсы.}

2.1.5. Сокращения определены в ГОСТ 27974.

2.2. Программа соответствует ГОСТ 27974.

3. ПРЕДЛОЖЕНИЯ

Все прагматические замечания соответствуют ГОСТ 27974.

3.0.1. *Синтаксис*

Гиперправила a, b, c, d, e соотстветствуют ГОСТ 27974.

f) * блок с 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};
определяющий СЛОЙ текст модуля в СРЕДЕ передающий !СВЯЗИ {49c,-};
модульный кортеж с ?ОПИСАНИЯМИ без ?ОПИСАНИЙ1 в СРЕДЕ с СЛОЕМ1 с СЛОЕМ2 {42d};
подключающее предложение ПРИВОДИМОЕ в СРЕДЕ {36a}.

3.0.2. Семантика соответствует ГОСТ 27974.

3.1. Замкнутые предложения определены в ГОСТ 27974.

3.2. Последовательные предложения

3.2.1. *Синтаксис*

Гиперправила a, b, c, d, e, f, g, h соответствуют ГОСТ 27974.

i) * определяющее предложение:
определяющее СЛОЙ последовательное предложение ПРИВОДИМОЕ в СРЕДЕ {32a};
определяющее СЛОЙ выясняющее предложение выдающее ВИД в СРЕДЕ {34c};
определяющий СЛОЙ текст модуля в СРЕДЕ передающий !СВЯЗИ {49c, -};
определяющее СЛОЙ подключение в СРЕДЕ передающее ?СВЯЗИ {36a, -}.

3.2.2. *Семантика*

а) 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, причем восстанавливающим действием служит вызов процедуры восстановление после ошибки области действия {10.2.5.n}.

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;

Для каждого составляющего определения-ситуации X этого C, если они вообще есть.

· сцена, составленная из

(i) X и

(ii) окружения E,

приписывается идентификатору-ситуации этого X в E;

Если каждая 'ПАРА', заложенная в '?ПАРЫ', есть 'ИНФИКС для БИНАРНОГО' или 'СЛОВО для метки' или 'СЛОВО для ситуации с видом ПРОЦЕДУРА',

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-букву-алеф-идентификатору-выдающему-целое;

Если C служит заглавием-цикла или спецификацией, то E 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

3.3.1. Синтаксис соответствует ГОСТ 27974.

3.3.2. *Семантика*

a) Гиперправило соответствует ГОСТ 27974.

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 $V_1, \ldots, V_m$ будут {совместными} выдачами составляющих основ из С;

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

· $V_1, \ldots, V_m$, взятые в их порядке, служат полями W;

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

· W состоит из

(i) паспорта $((1, m))$,

(ii) $V_1, \ldots, V_m$;

Для $i = 1, \ldots, m$

· V_i — элемент, выбираемый по индексу (i) в W;

Случай С: 'СОСТАВНОЕ есть вектор МАССИВОВ из ВИДА2':

· требуется, чтобы паспорта значений $V_1, \ldots, V_m$ были идентичны;

· пусть паспортом {,например,} V_1 будет $((L_1, u_1), \ldots, (L_n, u_n))$;

· W состоит из

(i) паспорта $((1, m), (L_1, u_1), \ldots, (L_n, u_n))$;

(ii) элементов этих $V_1, \ldots, V_m$;

Для $i = 1, \ldots, m$

· элементом, выбираемым по индексу $(i, i_1, \ldots, i_n)$ в W будет элемент, выбираемый по $(i_1, \ldots, i_n)$ в V_i.

Если не все паспорта значений $V_1, \ldots, V_m$ идентичны, восстанавливающим действием является следующее:

пусть U есть некоторый массив вида, специфицируемого описателем [ ] массив {10.2.3.1.a.} с паспортом $((1, m))$ и такой, что для i =

- 1, . . . , m, элементом, выбираемым по индексу (i) в U, является некоторый массив с паспортом, идентичным паспорту V_j:

• вызывается процедура восстановление после ошибки записи массива {10.2.5.k} со значениями {параметров} U и n {,где n есть число пар в паспорте V_1 }.

3.4. Выбирающие предложения определены в ГОСТ 27974.

3.5. Циклические предложения определены в ГОСТ 27974.

3.6. Подключающие предложения

3.6.1. *Синтаксис*

a) подключающее предложение ПРИВОДИМОЕ в СРЕДЕ {5D, 551a, A341h, A349a}:
определяющее СЛОИ подключение в СРЕДЕ передающее
ПУСТО {b},
ЗАКРЫТОЕ предложение в СРЕДЕ с СЛОЕМ ПРИВОДИМОЕ {a, 31a, 33a, c, d, e, 34a, 35a,-}.

b) определяющее новые ?ОПИСАНИЯ ?ПОДКЛЮЧЕНИЯ
подключение в СРЕДЕ передающее ?СВЯЗИ {a, 49c}
знак подключить {94d},
открывающий !СВЯЗИ групповой вызов модулей в СРЕДЕ передающий ?СВЯЗИ {c},
если ?ОПИСАНИЯ !ПОДКЛЮЧЕНИЯ открываются СВЯЗЯМИ {e, f} и в СРЕДУ проникают ?ПОДКЛЮЧЕНИЯ из ПОДКЛЮЧЕНИЙ {h}.

c) открывающий !СВЯЗИ групповой вызов модулей в СРЕДЕ передающий ?СВЯЗИ {b, c}:
открывающий !СВЯЗИ вызов модуля в СРЕДЕ передающий ?СВЯЗИ {d,-};
если (?СВЯЗИ) есть (?СВЯЗИ1 ?СВЯЗИ2) и (!СВЯЗИ) есть (!СВЯЗИ1 !СВЯЗИ2),
открывающий !СВЯЗИ1 вызов модуля в СРЕДЕ передающий ?СВЯЗИ1 {d,-},
знак а также {94f},
открывающий !СВЯЗИ2 групповой вызов модулей в СРЕДЕ передающий ?СВЯЗИ2 {c}.

d) открывающий !СВЯЗИ вызов модуля в СРЕДЕ
передающий ?СВЯЗИ {c}:
если (?СВЯЗИ) есть (ПУСТО),
использующий ИНДИКАНТ индикатор модуля в СРЕДЕ выдающий !СВЯЗИ с модулями {48b};
если (?СВЯЗИ) есть (!СВЯЗИ),
знак открытое {94d},
использующий ИНДИКАНТ индикатор модуля в СРЕДЕ выдающий !СВЯЗИ с модулями {48b}.

e) ЕСЛИ ?ОПИСАНИЯ1 ?ОПИСАНИЯ2 !ПОДКЛЮЧЕНИЯ1? ПОДКЛЮЧЕНИЯ2
открываются
КЛЮЧОМ для открытия ?ОПИСАНИЙ1 !ПОДКЛЮЧЕНИЙ1 ?СВЯЗЯМИ3
КЛЮЧОМ для открытия ?ОПИСАНИЙ1 !ПОДКЛЮЧЕНИЙ1 ?СВЯЗЯМИ4 {b, e, f }:
ЕСЛИ ?ОПИСАНИЯ1 ?ОПИСАНИЯ2 !ПОДКЛЮЧЕНИЯ1 ? ПОДКЛЮЧЕНИЯ2
открываются
КЛЮЧОМ для открытия ?ОПИСАНИЙ1 ПОДКЛЮЧЕНИЙ1 ?СВЯЗЯМИ3 ?СВЯЗЯМИ4 {e, f}.

f) ЕСЛИ ?ОПИСАНИЯ1 ?ОПИСАНИЯ2 !ПОДКЛЮЧЕНИЯ1: ? ПОДКЛЮЧЕНИЯ2
открываются
КЛЮЧОМ для открытия ?ОПИСАНИЙ1 !ПОДКЛЮЧЕНИЙ1 ?СВЯЗЯМИ2 {b, e, f}:
ЕСЛИ ?ОПИСАНИЯ2 ?ПОДКЛЮЧЕНИЯ2 открываются ?СВЯЗЯМИ2 и ?ОПИСАНИЯ1 не зависят от ?ОПИСАНИЙ2 {71a, b, c}.

g) ЕСЛИ ПУСТО открывается ПУСТО {e, f}: ЕСЛИ истина.

h) ЕСЛИ в СРЕДУ проникают ?ПОДКЛЮЧЕНИЯ1 из ? ПОДКЛЮЧЕНИЙ ПОДКЛЮЧЕНИЯ {b}:
если неверно, что ПОДКЛЮЧЕНИЕ идентифицировано в СРЕДЕ {72a},
ЕСЛИ (?ПОДКЛЮЧЕНИЯ1) есть (?ПОДКЛЮЧЕНИЯ2 ПОДКЛЮЧЕНИЕ) и в СРЕДУ {c} ПОДКЛЮЧЕНИЕМ проникают ?ПОДКЛЮЧЕНИЯ2 из ?ПОДКЛЮЧЕНИЙ {h,i,}:
если ПОДКЛЮЧЕНИЕ идентифицировано в СРЕДЕ {72a},
ЕСЛИ в СРЕДУ проникают ?ПОДКЛЮЧЕНИЯ1 из ? ПОДКЛЮЧЕНИЙ {h, i,}.

i) ЕСЛИ в СРЕДУ проникает ПУСТО из ПУСТО {h}: ЕСЛИ истина.

{Примеры:

a) подкл a, b (ввод (f, a); печ (a))

b) подкл a, b

c) a, b

d) a. откр b}

{В правиле b 'КЛЮЧИ для запуска', заложенные в '!ПОДКЛЮЧЕНИЯ', представляют те модули, запуск которых может потребоваться при любом вызове-модуля, использующий-индикатор-модуля которого идентифицирует конкретный определяющий-индикатор-модуля, в то время, как подобные гиперпонятия, заложенные в '?ПОДКЛЮЧЕНИЯ', представляют только те модули, которые нужно запускать в конкретном контексте, для осталь-

ных же это уже было исполнено, как можно статически определить по 'СРЕДЕ'. Наличие '?ПОДКЛЮЧЕНИЙ' в средах всех наследных конструктов подключающего-предложения гарантирует, что все запущенные к данному моменту модули никогда не будут заново запускаться внутри этих наследников.

Правило f обеспечивает независимость одновременно открываемых описаний, так, например,

модуль a = мд откр вещ x дм, b = мд откр вещ x дм;

подкл a, b (x)

не порождается. Тем не менее, правило e допускает, чтобы данное описание открывалось двумя открытыми подключениями одного и того же модуля, как в

модуль a = мд откр вещ x дм;

модуль b = подкл откр a мд вещ y дм;

c = подкл откр a мд вещ z дм;

подкл b c (x + y + z)

где определения-модуля и для b и для c открывают x посредством откр a в своих составляющих подключениях.}

3.6.2. *Семантика:*

а) подключающее-предложение-ПРИВОДИМОЕ-в-СРЕДЕ N в окружении E исполняется следующим образом:

Если, согласно 'СРЕДЕ', существует „первый незапущенный" {b} модуль M из подключения R этого N в E,

то

· пусть M состоит из определяющего-новые-?ПАРЫ-ПОДКЛЮЧЕНИЕ-текста-модуля T {вместе с необходимым окружением};

· M запускается {c} в E, развертывая новое окружение E4 {внутри которого участок 'ПОДКЛЮЧЕНИЕ' подключает результат запуска M};

· пусть Y – это выдача {a} в E4 подключающего-предложения-ПРИВОДИМОГО-в-СРЕДЕ- {c} -ПОДКЛЮЧЕНИЕМ, подобного N {, в котором относительно M известно, что он уже запущен};

· {M отключается, т. е.} кортеж составляющего заключения из T, если такой вообще есть, исполняется в E4;

· выдача N в E есть Y;

· требуется, чтобы Y по области действия было не младше E;

в противном случае

· пусть E2 – это окружение, устанавливаемое вокруг и наравне с E в соответствии с R ; участок этого E2 соответствует передаваемым свойствам модулей, подключаемых R};

· E2 „дополняется" {d} {значениями, передаваемыми составляющими вызовами-модулей} из R в E;

· выдача N в E есть выдача ЗАКРЫТОГО-предложения этого N в E2;

b) „Первый незапущенный" согласно некоторой 'СРЕДЕ' модуль из подключения R в окружении E определяется следующим образом:

Если существует некоторый составляющий открывающий-?СВЯЗИ-

КЛЮЧ-для открытия-?СВЯЗЕЙ-ПОДКЛЮЧЕНИЕ-вызов-модуля С из R, такой, что выполняется предикат 'если неверно что ПОДКЛЮЧЕНИЕ идентифицировано в СРЕДЕ,' и который текстуально является первым таким вызовом модуля, то

· пусть выдача использующего-индикатора-модуля этого С в Е есть {еще не запущенный} модуль М, состоящий из текста-модуля Т и окружения Е1 {необходимого (7.2.2.с) для Т}; Если Т содержит подключение S

и если согласно 'СРЕДЕ' существует первый незапущенный модуль М1 из S в Е1,

то М1 есть первый незапущенный модуль из R;

иначе М есть первый незапущенный модуль из R;

иначе не имеется никакого первого незапущенного модуля из R.

{ Выбор С среди вызовов-модулей из R зависит только от 'СРЕДЫ' и не зависит от Е. Из этого следует, что такой выбор всегда можно осуществить во время трансляции. Е требуется только для того, чтобы получить правильное окружение, необходимое для М.}

с) Модуль, состоящий из определяющего-новые-?ПАРЫ-ПОДКЛЮЧЕНИЕ-текста-модуля Т и окружения Е1 {, необходимого для Т}, запускается в окружении Е следующим образом:

Если Т содержит {уже запущенное} подключение S

то

· пусть Е2 есть окружение, устанавливаемое вокруг Е1, наравне с Е, согласно S;

· участок этого Е2 „дополняется" {d} {значениями, передаваемыми наследными вызовами-модулей,} из S в Е;

иначе, пусть Е2 есть Е1;

· пусть Е3 есть окружение, устанавливаемое вокруг Е2 и, если Е – это „окружение расположения модуля", то наравне с Е, а иначе по Е, согласно Т { ; участок этого Е3 соответствует всем свойствам (передаваемым или нет), описанным в Т};

· 'ПОДКЛЮЧЕНИЕ' получает доступ к модулю, состоящему из Т и Е3 внутри участка этого Е3 { , так что внутри самого Т будет видно, что Т уже запущен};

· в Е3 исполняется составляющее модульное-вступление из Т;

· пусть Е4 есть окружение, называемое „окружением расположения модуля", устанавливаемое вокруг Е, наравне с Е3, согласно некоторому определяющему-новое-ПОДКЛЮЧЕНИЕ-ПОНЯТИЮ;

· 'ПОДКЛЮЧЕНИЕ' получает доступ к модулю, состоящему из Т и Е3 внутри участка этого Е4;

· запуск М называется „развертывающим" окружение Е4.

{Все окружения, создаваемые в процессе запуска незапущенных модулей (b) из подключения некоторого подключающего-предложения N, имеют одну и ту же область действия, младшую, чем у окружения, в котором будем исполняться N, но старшую, чем у любого окружения, создаваемого в процессе исполнения ЗАКРЫТОГО-предложения этого N.}

d) Участок L „дополняется" из подключения R в окружении E следующим образом:

Для каждого наследного использующего-индикатора-модуля-выдающего-!СВЯЗИ-с-модулями этого R,

Для каждого 'КЛЮЧА для открытия !ПАР', заложенного {1.1.4.1.c} в 'СВЯЗИ',

· пусть модуль, „подключаемый" {e} 'КЛЮЧОМ для запуска внутри E {, он будет находиться в некотором окружении расположения модуля (c),} есть {уже запущенный} модуль, состоящий из текста-модуля T и окружения E3 {, в котором до этого было исполнено его модульное-вступление}:

Для каждого значения или сцены, подключаемых внутри участка этого E3 посредством некоторой 'ПАРЫ',

Если эта 'ПАРА' заложена в '!ПАРЫ' {, 'ПАРА' должна передаваемой},

то 'ПАРА' получает доступ к этому значению или сцене (если она еще не имеет такого доступа) также внутри L.

e) Значение или сцена, „подключаемые" 'ПАРОЙ' внутри окружения E состоящего из участка L и окружения E1, есть значение или сцена, доступная через 'ПАРУ' внутри L {2.1.2.c.}, если L соответствует '?ПАРАМ', в которые заложена {1.1.4.1.c.} эта 'ПАРА', а иначе значение или сцена, доступная через 'ПАРУ' внутри E1.

4. ОПИСАНИЯ, ОПИСАТЕЛИ И ИНДИКАТОРЫ

Все прагматические замечания соответствуют ГОСТ 27974.

4.1 Описания

4.1.1. *Синтаксис*

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

Гиперправила a, b, c, d соответствуют ГОСТ 27974.

e) описание с ?ОПИСАНИЯМИ без ?ОПИСАНИЙ1 в среде {49e}:
если (?ОПИСАНИЯ без ?ОПИСАНИЙ1)
если (ПУСТО без !ОПИСАНИЙ1),
описание ОБЪЕКТОВ через !ОПИСАНИЯ1 в СРЕДЕ {42a, 43a, 44a, e, 45a, 49a,-};
если (?ОПИСАНИЯ без ?ОПИСАНИЙ1) есть (!ОПИСАНИЯ без ПУСТО)
знак открытое {94d},
описание ОБЪЕКТОВ через !ОПИСАНИЯ в СРЕДЕ {42a, 43a, 44a, e, 45a, 49a,-};
если (?ОПИСАНИЯ без ?ОПИСАНИЙ1) есть
(?ОПИСАНИЯ без !ОПИСАНИЙ1 ?ОПИСАНИЯ2),

описание ОБЪЕКТОВ через !ОПИСАНИЯ1 в СРЕДЕ {42a, 43a, 44a, e, 45a, 49a,-},
знак а также {94f},
описание с ?ОПИСАНИЯМИ без ?ОПИСАНИЙ2 в СРЕДЕ {e};
если (?ОПИСАНИЯ без ?ОПИСАНИЙ1) есть
(!ОПИСАНИЯ ?ОПИСАНИЯ3 без ?ОПИСАНИЙ1),
знак открытое {94f},
описание ОБЪЕКТОВ через !ОПИСАНИЯ в СРЕДЕ {42a, 43a, 44a, e, 45a, 49a,-},
знак а также {94f},
описание с ?ОПИСАНИЯМИ3 без ?ОПИСАНИЙ1 в СРЕДЕ {e}.
{Модули могут запускаться подключающими-предложениями.}

4.1.2. Семантика соответствует ГОСТ 27974.

4.2. Описания видов соответствуют ГОСТ 27974.

4.3. Описания приоритетов соответствуют ГОСТ 27974.

4.4. Описания идентификаторов соответствуют ГОСТ 27974.

4.5. Описания операций соответствуют ГОСТ 27974.

4.6. Описатели определены в ГОСТ 27974.

4.7. Соотношения между видами определены в ГОСТ 27974.

4.8. Индикаторы и указатели полей

4.8.1. *Синтаксис*

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

Метаправила B, C, D соответствуют ГОСТ 27974.

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

F) ПРИЗНАК : : ВИД; ЗНАЧЕНИЕ НОМЕР; БИНАРНОЕ; метка; выборка ВИДА; !СВЯЗИ с модулями; запуск; СИТУАЦИЯ с видом ПРОЦЕДУРА.

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

Гиперправила a, b, c, d, e, f соответствуют ГОСТ 27974.

4.8.2. Семантика определена ГОСТ 27974.

4.9. Описания модулей

4.9.1. *Синтаксис*

a) описание модулей через !МОДУЛИ в СРЕДЕ1 {41a, e}:
знак модуль {94d},
групповое определение модулей через !МОДУЛИ в СРЕДЕ1 {41b, c}.

b) определение модуля через ИНДИКАНТ для ?СВЯЗЕЙ СВЯЗИ с модулями в СРЕДЕ1 {41c}:
если (СВЯЗЬ) есть (КЛЮЧ для открытия ?ОПИСАНИЙ ?ПОДКЛЮЧЕНИЙ {и} КЛЮЧ для запуска)
и (ИНДИКАНТ) есть (выделенное СЛОВО),
если (СРЕДА1) есть (ПОНЯТИЕ1 КЛЮЧ для запуска ?ПОНЯТИЕ2),

если неверно что (ПОНЯТИЕ1 ?ПОНЯТИЕ2)
содержит (КЛЮЧ для запуска),
определяющий ИНДИКАНТ индикатор модуля в СРЕДЕ1
выдающий ?СВЯЗИ СВЯЗЬ с модулями {48a},
знак определяется как {94d},
определяющий СЛОЙ текст модуля в СРЕДЕ1
передающий ?СВЯЗИ СВЯЗЬ {c,-}.

c) определяющий новые ?ОПИСАНИЯ1 ?ОПИСАНИЯ ПОДКЛЮЧЕНИЕ
текст модуля в СРЕДЕ1 передающий ?СВЯЗИ
КЛЮЧ для открытия ?ОПИСАНИЙ ?ПОДКЛЮЧЕНИЙ
ПОДКЛЮЧЕНИЯ {b}:
если (?ПОДКЛЮЧЕНИЯ) есть (ПУСТО) и (?СВЯЗИ) есть
(ПУСТО),
знак начало модуля ОФОРМЛЕННЫЙ {94d},
модульный кортеж с ?ОПИСАНИЯМИ без ?ОПИСАНИЙ1
в СРЕДЕ1 с новым {пустым СЛОЕМ}
с новыми ?ОПИСАНИЯМИ1 ?ОПИСАНИЯМИ ПОДКЛЮЧЕНИЕМ
{d},
знак конец модуля ОФОРМЛЕННЫЙ {94d};
определяющее СЛОЙ подключение в СРЕДЕ1
передающее ?СВЯЗИ {36b},
знак начало модуля ОФОРМЛЕННЫЙ {94d},
модульный кортеж с ?ОПИСАНИЯМИ без ?ОПИСАНИЙ1 в
СРЕДЕ с СЛОЕМ с новыми ?ОПИСАНИЯМИ1
?ОПИСАНИЯМИ ПОДКЛЮЧЕНИЕМ {d},
знак конец модуля ОФОРМЛЕННЫЙ {94d},
если (СЛОЙ) есть (новые ?ОПИСАНИЯ2 ?ПОДКЛЮЧЕНИЯ).

d) модульный кортеж с ?ОПИСАНИЯМИ без ?ОПИСАНИЙ1 в СРЕДЕ3
{c}:
модульное вступление с ?ОПИСАНИЯМИ без ?ОПИСАНИЙ1 в
СРЕДЕ3 {e},
возможное модульное заключение в СРЕДЕ3 {f}.

e) модульное вступление с ?ОПИСАНИЯМИ1 без ?ОПИСАНИЙ2 в
СРЕДЕ3 {d, e}:
основа в СРЕДЕ3 сильно выдающая пустое значение {32d},
знак продолжать {94f},
модульное вступление с ?ОПИСАНИЯМИ1 без ?ОПИСАНИЙ2
в СРЕДЕ3 {e}:
если (?ОПИСАНИЯ1 без ?ОПИСАНИЙ2) есть
(?ОПИСАНИЯ3 ?ОПИСАНИЯ4 без ?ОПИСАНИЙ5 ?ОПИСА-
НИЙ6),
описание с ?ОПИСАНИЯМИ3 без ?ОПИСАНИЙ 5 в СРЕДЕ {41e},
знак продолжать {94f},
модульное вступление с ?ОПИСАНИЯМИ4 без ?ОПИСАНИЙ 6 в
СРЕДЕ3 {e};

если (?ОПИСАНИЯ1 без ?ОПИСАНИЙ2) есть (ПУСТО) без (ПУСТО),
основа в СРЕДЕ3 сильно выдающая пустое значение {32d};
описание с ?ОПИСАНИЯМИ1 без ?ОПИСАНИЙ 2 в СРЕДЕ3 {41e}.

f) модульное заключение в СРЕДЕ {d}:
знак сброс модуля ОФОРМЛЕННЫЙ {94d},
кортеж с ПУСТО сильно выдающий пустое значение в СРЕДЕ3 {32b}.

g)* текст модуля:
определяющий СЛОЙ текст модуля в СРЕДЕ передающий !СВЯЗИ {c}.

{Примеры:

```
a) модуль a = мд строк s; чит (s);
                    откр строк t = "файл" + s, откр вещ a дм,
   b = подкл a мд откр файл f;
                    открыть (f, t, стандканал ввода);
                    сброс закрыть (f) дм.
b) a = мд строк s; чит (s);
          откр строк t = "файл" + s, откр вещ a дм,
   b = подкл a мд откр файл f;
                    открыть (f, t, стандканал ввода);
                    сброс закрыть (f) дм.
c) мд строк s; чит (s);
       откр строк t = "файл" + s, откр вещ a дм;
   подкл a мд откр файл f;
                 открыть (f, t, стандканал ввода);
                 сброс закрыть (f) дм.
d) строк s; чит (s) : откр строк t = "файл" + s, откр вещ a;
   откр файл f; открыть (f, t, стандканал ввода);
                 сброс закрыть (f).
e) строк s; чит (s); откр строк t = "файл" + s, откр вещ a;
   откр файл f: открыть (f, t, стандканал ввода).
f) сброс закрыть (f).}.
```

{Правило b гарантирует, что с каждым текстом-модуля, подключаемым в любом заданном месте программы, связывается единственный 'КЛЮЧ'. Это используется для того, чтобы обеспечить возможность идентификации (7.2.1.a) 'КЛЮЧА для запуска' в средах всех наследных конструктов любого подключающего-предложения или текста-модуля, которые запускают этот текст-модуля.

Вообще говоря, определяющий-СЛОЙ-текст-модуля-передающий-!СВЯЗИ Т делает 'СЛОЙ' видимым внутри самого Т и делает свойства, открываемые '!СВЯЗЯМИ', видимыми всюду, где подключается Т. 'СЛОЙ' включает в себя как '?ОПИСАНИЯ', соответствующие его открытым описаниям (например, t и a в первом тексте-модуля из примера с и '?ОПИСАНИЯ1', соответствующие его скрытым описаниям (например, s в

этом же примере), так и 'ПОДКЛЮЧЕНИЕ', связывающее Т с единственным соответствующим ему 'КЛЮЧОМ' и означающее, что в данной среде теперь известно о наличии запуска этого Т.

'!СВЯЗИ' всегда открывают '?ОПИСАНИЯ ?ПОДКЛЮЧЕНИЯ ПОДКЛЮЧЕНИЕ' (но не '?ОПИСАНИЯ1'), где '?ПОДКЛЮЧЕНИЯ' означают запуск любых других модулей, подключаемых Т, если их вызовы-модулей внутри Т содержат знак-открытое }

4.9.2. *Семантика*

a) „Модуль" – это сцена {2.1.1.1.d}, состоящая из текста-модуля вместе с окружением {2.1.1.1.c}.

b) Описание-модулей D исполняется следующим образом:

· совместно исполняются составляющие тексты-модулей этого D;

Для каждого составляющего определения-модуля D1 из D,

· выдача {c} текста-модуля этого D1 приписывается {4.8.2.a} определяющему-индикатору-модуля этого D1.

c) Выдача текста-модуля Т в окружении Е – это модуль, состоящий из

(i) Т и

(ii) окружения, необходимого для {7.2.2.c} Т в Е.

d) Модульное-вступление С в окружении Е исполняется следующим образом:

· в Е исполняется его основа или описание;

Если его прямым наследником является другое модульное-вступление D, то D исполняется в Е

{; в противном случае исполнение С завершено}.

4.10. Ситуации и реакции

{Ситуация – это такое состояние, обнаруживаемое реализацией или программой пользователя, при котором требуется действие, зависящее от текущего окружения. Это действие представляет собой вызов некоторой процедуры. Определение-ситуации вводит новый тип ситуации и задает вид процедуры, которую нужно вызывать для этой ситуации. Определение-реакции задает конкретную процедуру, которая должна использоваться для этой ситуации на время жизни текущего окружения, исключая производные окружения, которые могут задавать свои процедуры для той же ситуации. Определяющий-идентификатор-реакции некоторого определения-реакции рассматривается как использующий-идентификатор-ситуации, который должен идентифицировать определяющий-идентификатор-ситуации из соответствующего определения-ситуации. Использующих-идентификаторов-реакции не существует.}

4.10.1. *Синтаксис*

а) описание ситуаций с видом ПРОЦЕДУРА через !ОПИСАНИЯ в СРЕДЕ {41 a}:
знак СИТУАЦИЯ {94F}, групповое определение ситуаций с видом ПРОЦЕДУРА через !ОПИСАНИЯ в СРЕДЕ {41 b, c}.

в) определение ситуации с видом ПРОЦЕДУРА через СЛОВО для ситуации с видом ПРОЦЕДУРА в СРЕДЕ {41 c}: определяющий СЛОВО

идентификатор в СРЕДЕ выдающий ситуацию с видом ПРОЦЕДУРА {48a}.

c) описание реакций через !ОПИСАНИЯ в СРЕДЕ {41 a}:
знак РЕАКЦИЯ {94 f}, групповое определение реакций через !ОПИСАНИЯ в СРЕДЕ {41 b, c}.

d) определение реакции через СЛОВО для реакции с видом ПРОЦЕДУРА в СРЕДЕ {41c}:
если СЛОВО для ситуации с видом ПРОЦЕДУРА идентифицировано в среде {72a},
определяющий СЛОВО идентификатор в СРЕДЕ выдающий реакцию с видом ПРОЦЕДУРА {48a}, знак двоеточие {94f}, источник вида ПРОЦЕДУРА в СРЕДЕ {521c}.

{Примеры:

a) ситуация (вещ) пуст недопустимый аргумент
b) (вещ) пуст недопустимый аргумент
c) присит недопустимый аргумент: (вещ x) пуст: финиш
d)недопустимый аргумент: (вещ x) пуст: финиш}

4.10.2. *Семантика*

a) исполнение описания-ситуаций {не требует действий, не выдает значения и тем самым} завершено.

b) описание-реакций D в окружении E исполняется следующим образом:

- совместно исполняются составляющие источники из D в E;
  для каждого составляющего определения-реакции D1 из D
- пусть V есть выдача источника из D1:
- пусть X есть выдача использующего-идентификатора-ситуации, подобного определяющему-идентификатору-реакции из D1 в E;
- V начинает реагировать на X внутри участка из E.

5. ОСНОВЫ

Все прагматические замечания соответствуют ГОСТ 27974.

5.1. Синтаксис

A) ОСНОВА {32d}: : приведенное присваивание {521a};
приведенное отношение одноименности {522a};
приведенный текст процедуры {541a, b}; переход {544a};
пропуск {552a}; ТРЕТИЧНОЕ {B};
формальная заготовка;
виртуальная заготовка.

Метаправила B, C соответствуют ГОСТ 27974.

D) ПЕРВИЧНОЕ {C 532a, 543a}: : приведенная вырезка {532a};
приведенный вызов {551a}; приведенный вызов ситуации {545a};
приведенное изображаемое {80a}; приведенное ядро {551a},
приведенный текст формата {A341a}; приведенный использующий СЛОВО идентификатор {48 b};

ЗАКРЫТОЕ предложение {31a, 33a, c, d, e, 34a, 35a}.

Гиперправило a соответствует ГОСТ 27974.

5.2. Основы, связанные с именами

5.2.1. *Присваивания.*

5.2.1.1. Синтаксис определен в ГОСТ 27974.

5.2.1.2. Семантика.

Гиперправило a соответствует ГОСТ 27974.

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;

Если N есть псевдоимя, то восстанавливающим действием служит вызов процедуры восстановление после ошибки псевдоимени {10.2.5.l};

Если W по области действия младше N, то восстанавливающим действием служит вызов процедуры восстановление после ошибки области действия {10.2.5.m}:

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;

Если дескрипторы W и V не идентичны, восстанавливающим действием является следующее:

- пусть n есть число пар в паспорте W;
- пусть i есть некоторое целое число, такое, что $1 \leqslant i \leqslant n$ и пары с номером i в паспортах W и V не совпадают;
- вызывается процедура восстановление после ошибки присваивания {10.2.5.h} со значениями {параметров} N, W, n, i;

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 I, выбирающего элемент в W,

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

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

- N начинает именовать {2.1.3.2a} W.

5.2.2. Отношения одноименности определены в ГОСТ 27974.

5.2.3. Генераторы определены в ГОСТ 27974.

5.2.4. Псевдоимена соответствуют ГОСТ 27974.

5.3. Основы, связанные с составными значениями

5.3.1. *Выборки*

5.3.1.1. Синтаксис соответствует ГОСТ 27974.

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, причем восстанавливающим действием служит вызов процедуры восстановление после ошибки псевдоимени {10.2.5.l};

• W – значение, выбираемое в {2.1.3.3a, e, 2.1.3.4.k}, или имя, генерируемое из {2.1.3.4.l} V по указателю-поля этого S.

5.3.2. *Вырезки*

5.3.2.1. Синтаксис соответствует ГОСТ 27974.

5.3.2.2. Семантика.

а) выдача W вырезки S определяется посредством следующих шагов:

Шаг 1:

• пусть V и $(I_1, \ldots, I_n)$ – {совместные} выдачи ПЕРВИЧНОГО вырезки S и индексатора {b} из S;

• требуется, чтобы V {, если оно имя,} не было псевдоименем, восстанавливающим действием служит вызов процедуры восстановление после ошибки псевдоимени {10.2.5.l};

• пусть $((r_1, s_1), \ldots, (r_n, s_n))$ – паспорт выдачи V или значения, именуемого V;

Шаг 2: для i = 1, . . . , n

случай A: I_i – целое число:

• требуется, чтобы $r_i \leqslant I_i \leqslant s_i$;

случай B: I_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 $U \leqslant s_i$;

• I_i заменяется на (L, U, l').

Восстанавливающее действие для этого шага состоит в следующем:

• пусть i и b – некоторые числа, такие, что $1 \leqslant i \leqslant n$ и либо случай A1 : I_i – целое число, $I_i < r_i$ или $I_i > s_i$ и $b = I_i$, либо случай B1 : I_i – тройка (l, u, l'), возможно, измененная на предыдущих шагах,

l не отсутствует, $l < r$ и $b = l$, либо случай B2: I_i – тройка (l,u,l'), возможно, измененная на предыдущих шагах,

u не отсутствует, $u > s_i$ и $b = u$;

• пусть R есть процедура восстановление после ошибки границы имени {10.2.5.j}, если V имя, и процедура восстановление после ошибки границы {10.2.5.i} в противном случае:

• пусть b′ – выдача вызова со значениями {параметров} V, n, i и b.

· для случая A1 : I_i заменяется на b′;
· для случая B1 : l из I_i заменяется на b′;
· для случая B2: u из I_i заменяется на b′;
· шаг 2 выполняется сначала.

Шаг 3: для i = 1, . . . , n, если I_i тройка (l, u, l′),
· 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′ заменяется на D.

Шаг 4: W — значение, выбираемое в {2.1.3.4.a.g, i}, или имя, генерируемое из {2.1.3.4.j} V по (I_1, . . . , I_n).

b) Гиперправило соответствует ГОСТ 27974.

5.4. Основы, связанные с процедурами

5.4.1. Тексты процедур соответствуют ГОСТ 27974.

5.4.2. Формулы соответствуют ГОСТ 27974.

5.4.3. Вызовы определены в ГОСТ 27974.

5.4.4. Переходы определены в ГОСТ 27974.

5.4.5. *Вызовы ситуаций*

{Вызов-ситуации служит для того, чтобы возбудить ситуацию и таким образом вызвать процедуру, назначенную для реагирования на эту ситуацию в текущем окружении. Вызов-ситуации может обеспечивать параметры для этой процедуры. Процедура реакции ищется, начиная с текущего окружения, по всем окружениям со старшими областями действия, исключая тот случай, когда во время исполнения некоторого вызова-ситуации вновь возбуждается та же самая ситуация. В последнем случае внутренний вызов-ситуации не использует процедуру реакции, найденную для внешнего вызова-ситуации, и поиск процедуры реакции продолжается с окружения, область действия которого является следующей старшей после области действия окружения с участком, содержащим первую процедуру реакции. В некоторых языках программирования подобный процесс называется распространением ситуации. }

5.4.5.1. Синтаксис.

а) вызов ситуации в СРЕДЕ выдающий ЗНАЧЕНИЕ {5 D}: возможный знак возбудить {94 f}, использующий СЛОВО идентификатор в СРЕДЕ выдающий ситуацию с видом процедура ?ПАРАМЕТРИЗОВАННАЯ вырабатывающая ЗНАЧЕНИЕ {48 b},
параметризация ?ПАРАМЕТРИЗОВАННАЯ в СРЕДЕ {b, c}.

b) параметризация с !ПАРАМЕТРАМИ в СРЕДЕ {a}: упакованные кратким фактические ! ПАРАМЕТРЫ в СРЕДЕ {543 b, c}.

c) параметризация в СРЕДЕ {a}: ПУСТО.

{Примеры:

а) возбуд недопустимый аргумент (x) }

5.4.5.2. Семантика.

а) выдача W вызова-ситуации Y в окружении E определяется следующим образом:

• пусть X есть выдача использующего-идентификатора-ситуации из Y в E;
• пусть H и F – это, соответственно, процедура и окружение реакции {b} для X в E;
пусть E1 – новое {локализующее, см. 3.2.2.2.b} окружение, устанавливаемое вокруг E; E1 называется связанным с F посредством X,
• пусть $V_1, \ldots, V_n$ будут {совместными} выдачами составляющих фактических параметров этого Y, если они вообще есть, в E1;
• W есть выдача „вызова" {5.4.3.2.b} H в E1, возможно, со значениями {параметров} $V_1, \ldots, V_n$;
• требуется, чтобы W по области действия не была младше E, причем восстанавливающим действием служит вызов процедуры восстановление после ошибки области действия {10.2.5.n}.

b) процедура реакции H и окружение реакции F для сцены X в окружении E определяются следующим образом:
• требуется, чтобы E по области действия не было младше окружения из X, причем восстанавливающим действием является вызов процедуры восстановление после общей ситуации {10.2.5.o};
• если существует значение R, реагирующее на X внутри участка из E, то H есть R и F есть E;
• иначе
  • пусть E1 есть отсчетное окружение {c} для X в E;
  • пусть E2 есть окружение, по которому {3.2.2.b} установлено окружение E;
  • H и F – это процедура и окружение реакции для X в E2.

c) отсчетное окружение F для сцены X в окружении E определяется следующим образом:
• если E связано посредством X с другим окруженим E1, то F есть E1;
• иначе F есть E.

5.5. Основы, связанные со значениями любого вида, соответствуют ГОСТ 27974.

5.6. Заготовки

5.6.1. *Синтаксис*

A) ЯЗЫК : : алгол шестьдесят восемь.

К вышеприведенному метаправилу могут быть добавлены дополнительные гиперпонятия {например, „фортран"}.

B) АЛГОЛ 68 : : алгол шестьдесят восемь.

a) виртуальная заготовка в СРЕДЕ сильно выдающая ЗНАЧЕНИЕ {5A}:
  символ виртуальная среда,
    замкнутое предложение в СРЕДЕ
      сильно выдающее ЗНАЧЕНИЕ {31a}.

b) формальная заготовка в СРЕДЕ сильно выдающая ЗНАЧЕНИЕ {5A}:
  знак формальная среда {94d},
  индикатор ЯЗЫКА выдающий ЗНАЧЕНИЕ {e, f, ·},

индикатор заготовки {d}.

c) фактическая заготовка в СРЕДЕ выдающая ЗНАЧЕНИЕ {A6a}:
ЗАКРЫТОЕ предложение в СРЕДЕ сильно выдающее ЗНАЧЕНИЕ {31a,33a,c,34a,35a,-,36a,-}.

d) индикатор заготовки {b}
изображение литерного {814a};
изображение вектора из литерных {83a}.

e) индикатор АЛГОЛ 68 выдающий ЗНАЧЕНИЕ {b}: ПУСТО.

f) для каждого дополнительного терминального метапорождения „ЯЗЫКА" должны быть добавлены гиперправила для гиперпонятий формы „индикатор ЯЗЫКА выдающий ЗНАЧЕНИЕ"; каждое из таких правил содержит ровно одну альтернативу, которая должна быть отличимым знаком выделенное СЛОВО.

{Примеры:

b) среда "abc"

c) подкл a, b (x: = 1; y: = 2; печ (x + y))

d) " a " "abc"}

{Поскольку для символа-виртуальная-среда не задано никакого представления, пользователь не может сам создавать виртуальные-заготовки, однако задан механизм (10.6.2.a) для построения их из формальных- и фактических-заготовок.}

{Выдачей виртуальной-заготовки является, вследствие предыполнения (2.1.4.1.c), выдача ее замкнутого-предложения. Для формальных- или фактических-заготовок не задано никакой семантики, поскольку их исполнение никогда не требуется.}

6. ПРИВЕДЕНИЕ

Все прагматические замечания соответствуют ГОСТ 27974.

6.1. Приведенные

6.1.1. *Синтаксис*

Метаправила A, B, C, D, E, G соответствуют ГОСТ 27974.

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

Гиперправила a, b, c, d, e, f, g, h соответствуют ГОСТ 27974.

6.2. Разыменование

6.2.1. Синтаксис соответствует ГОСТ 27974.

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, причем восста-

навливающим действием служит вызов процедуры восстановление после ошибки псевдоимени {10.2.5.1};

• W – значение, именуемое этим N.

6.3. Распроцедуривание соответствует ГОСТ 27974.

6.4. Объединение соответствует ГОСТ 27974.

6.5. Обобщение соответствует ГОСТ 27974.

6.6. Векторизация соответствует ГОСТ 27974.

6.7. Опустошение соответствует ГОСТ 27974.

7. ВИДЫ И СРЕДЫ

Все прагматические замечания соответствуют ГОСТ 27974.

7.1. Независимость свойств

7.1.1. *Синтаксис*

Метаправила A, B соответствуют ГОСТ 27974.

C) ЕСЛИ ОБОЗНАЧЕНИЕ1 для ПРИЗНАКА1 не зависит от ОБОЗНАЧЕНИЯ 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};

если (ОБОЗНАЧЕНИЕ1) есть (ОБОЗНАЧЕНИЕ2) и (ОБОЗНАЧЕНИЕ1) есть (СЛОВО),

если (ПРИЗНАК1) есть (СИТУАЦИЯ1 с видом ПРОЦЕДУРА1) и (ПРИЗНАК2) есть (СИТУАЦИЯ2 с видом ПРОЦЕДУРА2),

ЕСЛИ (СИТУАЦИЯ1 СИТУАЦИЯ2) есть (ситуация реакция) или (СИТУАЦИЯ1 СИТУАЦИЯ2) есть (РЕАКЦИЯ ситуация).

Гиперправила a, b, c, d, e, f, g, h, i, j, k, l, m, n, соответствуют ГОСТ 27974.

7.2. Идентификация в средах

7.2.1. *Синтаксис*

Гиперправила a, b соответствуют ГОСТ 27974.

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), или (ПРИЗНАК1) есть (!СВЯЗИ с модулями) или (ПРИЗНАК1) есть (запуск),

ЕСЛИ (ПРИЗНАК1) есть (ПРИЗНАК2);

если (ПРИЗНАК1) есть (ЗНАЧЕНИЕ1 ?НОМЕР) и (ПРИЗНАК2) есть (ЗНАЧЕНИЕ2 ?НОМЕР),

ЕСЛИ ЗНАЧЕНИЕ1 эквивалентно ЗНАЧЕНИЮ2 {73a};

если (ПРИЗНАК1) есть (СИТУАЦИЯ с видом ПРОЦЕДУРА1) и (ПРИЗНАК2) есть (СИТУАЦИЯ с видом ПРОЦЕДУРА2),

ЕСЛИ ПРОЦЕДУРА1 эквивалентна ПРОЦЕДУРЕ2 {73a}.

7.2.2. *Семантика*

Гиперправила a, d соответствуют ГОСТ 27974.

b) Определяющий блок-в СРЕДЕ {a} каждого использующего-ОБОЗНАЧЕНИЕ-индикатора-выдающего-ПРИЗНАК I1 {по необходимости} содержит либо определяющий-ОБОЗНАЧЕНИЕ-индикатор-в-СРЕДЕ-с-СЛОЕМ-выдающий-ПРИЗНАК I2, либо один или {возможно} несколько использующих-индикаторов-модулей I3, являющихся прямыми наследниками открывающих-!СВЯЗИ-вызовов-модулей-в-СРЕДЕ, в '!СВЯЗИ' которых заложено 'ОБОЗНАЧЕНИЕ для ПРИЗНАКА'. В этом случае говорится, что I1 идентифицирует это I2 или каждое из этих I3.

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-ОБОЗНАЧЕНИЕ-индикатор-выдающий-ПРИЗНАК,

- не идентифицирующий {b} никакого определяющего-индикатора, содержащегося в C,
- не являющийся прямым наследным индикатором-вида для формального-или виртуального-описателя и
- такой, что предикат 'если ОБОЗНАЧЕНИЕ для ПРИЗНАКА находится в ?ПАРАХ' {7.2.1.b} выполняется,

или если C содержит виртуальную-заготовку,

то E есть E1;

иначе {L не необходимо для C и} E – окружение, необходимое для C в E2.

7.3. Эквивалентность видов соответствует ГОСТ 27974.

7.4. Правильность построения соответствует ГОСТ 27974.

8. ИЗОБРАЖЕНИЯ

Изображения определены в ГОСТ 27974.

9. ЗНАКИ И СИМВОЛЫ

Все прагматические замечания соответствуют ГОСТ 27974.

9.1. Знаки соответствуют ГОСТ 27974.

9.2. Примечания и прагматы соответствуют ГОСТ 27974.

9.3. Представления соответствуют ГОСТ 27974.

9.4. Эталонный язык

Гиперправила a, b, c, d соответствуют ГОСТ 27974.

9.4.1. *Представления символов.*

Списки символов a, b, c, e, g, h соответствуют ГОСТ 27974.

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	
символ модуль {49a}	module	модуль	
символ подключить {36b}	access	подкл	
символ начало модуля выделенный {49c}	def		
символ начало модуля стиля I		мд	
символ конец модуля выделенный {49c}	fed		
символ конец модуля стиля I		дм	
символ открытое {36d, 41e}	pub	откр	
символ сброс модуля {49f}	postlude	сброс	
символ формальная среда {56b}	nest	среда	
символ сегмент {A, b, a, c}	cgg	сегмент	

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
символы начало стиля II {133d}		нач
символ конец стиля II {133d}		кон
символ а также {133c, 33b, f, 34h, 41a, b 46e, i, g, t, 532b, 541e, 543b, A348b, A34A, c, 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 {544 b}		goto		
символ на краткий {544b}			на	
символ иди выделенный {544b}		go		
символ пропуск {552a}		skip	пропуск	скип
символ форматор {A341a}		$	Ф	
символ ситуация		exception		ситуация
символ реакция		on		присит
символ возбудить		raise		возбуд

9.4.2. Символы прочих обозначений соответствуют ГОСТ 27974.

10. СТАНДАРТНАЯ ЯЗЫКОВАЯ ОБСТАНОВКА

Прагматическое замечание соответствует ГОСТ 27974.

10.1. Тексты программ

Все прагматические замечания соответствуют ГОСТ 27974.

10.1.1. *Синтаксис*

A) ВНЕШНЕЕ : : стандартное;

библиотечное; системное; собственное; личное.

Метаправило B и гиперправила a, b, c, d, e, g, h, i соответствуют ГОСТ 27974.

f) задача пользователя в СРЕДЕ1 {d}:

собственное вступление с !ОПИСАНИЯМИ в СРЕДЕ2 {e},
личное вступление с ?МОДУЛЯМИ в СРЕДЕ2 {e},
УПАКОВКА собственно-программы в СРЕДЕ2 {g},
знак продолжать {94 f},
собственное-заключение в СРЕДЕ2 {i},
если (СРЕДА2) есть (СРЕДА1 с
новыми !ОПИСАНИЯМИ ?МОДУЛЯМИ {и} СТОПОМ).

10.1.2. *Соответствие языковой обстановке*

Определения, a, b, c, d, e соответствуют ГОСТ 27974.

f) Если явно не оговорено противное {10.6.2.a}, то каждое составляющее личное-вступление всех текстов-программ есть ПУСТО.

10.1.3. Способ описания стандартной языковой обстановки соответствует ГОСТ 27974.

10.2. Стандартное вступление

Все прагматические замечания соответствуют ГОСТ 27974.

10.2.1. *Запросы к обстановке*

Описания a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u соответствуют ГОСТ 27974.

v) проц Д цел переполнение включено =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры Д цел восстановление после переполнения {10.2.3.13}, реализация действительно выполняет такое действие; ложь в противном случае с;
проц лог L int overflow enabled = Д цел переполнение включено;
проц Д вещ переполнение включено =лог:
с истина, если при возникновении ситуации, для которой в качестве, восстанавливающего действия задан вызов процедуры Д вещ восстановление после переполнения {10.2.3.13}, реализация действительно выполняет такое действие; ложь в противном случае с ;
проц лог L real overflow enabled = Д вещ переполнение включено;
проц Д вещ потеря значимости включена =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры Д вещ восстановление после потери значимости {10.2.3.13}, реализация действительно выполняет такое действие: ложь в противном случае с;
проц лог L real underflow enabled = Д вещ потеря значимости включена;
проц Д цел ошибка аргумента включена =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры Д цел восстановление после ошибки аргумента {10.2.3.13}, реализация действительно выполняет такое действие; ложь в противном случае с;
проц лог L int argument error enabled = Д цел ошибка аргумента включена;
проц Д вещ ошибка аргумента включена =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры Д вещ восстановление после ошибки аргумента {10.2.3.13}, реализация действительно выполняет такое действие; ложь в противном случае с;
проц лог L real argument error enabled = Д вещ ошибка аргумента включена;
w) проц ошибка присваивания включена =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры восстановление после ошибки присваивания{5.2.1.2.b}, реализация действительно выполняет такое действие: ложь в противном случае с;
проц лог assignment error enabled = ошибка присваивания включена;
проц ошибка границы включена =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры восстановление после ошибки границы {5.3.2 2.*a*}, реализация действительно выполняет такое действие; ложь в противном случае с;
проц лог bound error enabled = ошибка границы включена;
проц ошибка записи массива включена =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры восстановление после

ошибки записи массива {3.3.2.b}, реализация действительно выполняет такое действие; ложь в противном случае с;
проц лог row display error enabled = ошибка записи массива включена;
проц ошибка псевдоимени включена =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры восстановление после ошибки псевдоимени {5.2.1.2.b, 5.3.1.2, 5.3.2.2.a, 6.2.2}, реализация действительно выполняет такое действие;
ложь в противном случае с;
проц лог nil error enabled = ошибка псевдоимени включена;
проц ошибка области действия включена =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры восстановления после ошибки области действия {3.2.2.a, 5.2.1.2.b; 5.4.5.2.b}, реализация действительно выполняет такое действие; ложь в противном случае с;
проц лог scope error enabled = ошибка области действия включена;
проц тупик включен =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры восстановление после тупика {10.2.4.d}, реализация действительно выполняет такое действие; ложь в противном случае с;
проц лог deadlock enabled = тупик включен;
проц исчерпание времени включено =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры восстановление после исчерпания времени {2.1.4.3.i}, реализация действительно выполняет такое действие; ложь в противном случа с;
проц лог time eshaustion enabled = исчерпание времени включено;
проц исчерпание памяти включено =лог:
с истина, если при возникновении ситуации, для которой в качестве восстанавливающего действия задан вызов процедуры восстановление после исчерпания памяти {2.1.4.3.i}, реализация действительно выполняет такое действие; ложь в противном случае с;
проц лог space exhaustion enabled = исчерпание памяти включено;

10.2.2. Стандартные виды соответствуют ГОСТ 27974.

10.2.3. *Стандартные обозначения операций и функции*

10.2.3.0. Стандартные приоритеты определены в ГОСТ 27974.

10.2.3.1. Массивы и связанные с ними операции соответствуют ГОСТ 27974.

10.2.3.2. Операции над логическими операндами определены в ГОСТ 27974.

10.2.3.3. Операции над целыми операндами определены в ГОСТ 27974.

10.2.3.4. Операции над вещественными операндами определены в ГОСТ 27974.

10.2.3.5. Операции над арифметическими операндами определены в ГОСТ 27974.

10.2.3.6. Операции над литерными операндами определены в ГОСТ 27974.

10.2.3.7. Операции над комплексными операндами определены в ГОСТ 27974.

10.2.3.8. Битовые и связанные с ними операции соответствуют ГОСТ 27974.

10.2.3.9. Слоговые и связанные с ними операции соответствуют ГОСТ 27974.

10.2.3.10. Строковые и связанные с ними операции соответствуют ГОСТ 27974.

10.2.3.11. Операции, соединенные с присваиваниями, соответствуют ГОСТ 27974.

10.2.3.12. Стандартные математические константы и функции соответствуют ГОСТ 27974.

10.2.3.13. Восстанавливающие действия для стандартных операций и функций.

Для случаев, когда операции и функции из п. 10.2.3 не дают осмысленного результата, определяются восстанавливающие действия в виде вызовов процедур из 10.2.5.g.

Процедура Д цел восстановление после переполнения (Д вещ восстановление после переполнения) вызывается при неудаче, если ожидается, что аналогичное вычисление могло бы быть успешным в другой реализации с большим значением Д макс цел (Д макс вещ).

Процедура Д восстановление после потери значимости вызывается при неудаче, вызванной тем, что результат вида Д вещ арифметической операции с ненулевыми значениями операндов не может быть представлен в форме, позволяющей определить, равен ли он нулю.

Процедура Д цел восстановление после ошибки аргумента (Д вещ восстановление после ошибки аргумента) вызывается при неудаче со значением {параметра} х вида Д цел (Д вещ). Если это значение было использовано в качестве значения фактического параметра или операнда и неудача связана с тем, что для этого значения результат не определен математически.

10.2.4. *Операции синхронизации*

Описания a, b, c, e соответствуют ГОСТ 27974.

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 P будет таким процессом, что исполнение данного псевдопримечания {10.1.3. Шаг 7} есть наследное действие этого P, но никакого другого процесса, наследного для P; данный процесс P приостанавливается {2.1.4.3.f},

если все процессы, наследные для собственно-программы, приостановлены таким образом и ни один из них не возобновлен, то последующее исполнение не определено, а восстанавливающее действие определяется как вызов процедуры восстановление после тупика {10.2.5.п} с выдачей 'эдсгер' {в качестве значения параметра} с; истина

все
прагм конец несовместимой части прагм
цк пропуск кц
конец;

10.2.5. *Стандартные ситуации и восстанавливающие действия*

а) ситуация пуст Д цел переполнение,
пуст Д вещ переполнение,
пуст Д потеря значимости,
ситуация (Д цел) пуст Д цел ошибка аргумента,
ситуация (Д вещ) пуст Д вещ ошибка аргумента;
ситуация пуст L int overflow,
пуст L real overflow,
пуст L underflow,
ситуация (Д цел) пуст L argument error,
ситуация (Д вещ) пуст L real argument error;
присит L int overflow: пуст: возбуд Д цел переполнение,
L real overflow: пуст: возбуд Д вещ переполнение,
L underflow: пуст: возбуд Д потеря значимости,
L int argument error:
(Д цел а) пуст: возбуд Д цел ошибка аргумента (а),
L real argument error:
(Д вещ а) пуст: возбуд Д вещ ошибка аргумента (а);

b) ситуация (цел) пуст ошибка присваивания,
ситуация (цел, имя цел) пуст ошибка границы,
ситуация пуст ошибка записи массива,
пуст ошибка псевдоимени,
пуст ошибка области действия,
ситуация (сема) пуст тупик,
ситуация пуст общая ситуация;
ситуация (цел) пуст assignment error,
ситуация (цел, имя цел) пуст bound error,
ситуация пуст row display error,
пуст nil error,
пуст scope error,
ситуация (сема) пуст deadlock,
ситуация пуст general exception;

присит assignment error:
(цел a) пуст: возбуд ошибка присваивания (a),
bound error: (цел a, имя цел b) пуст: возбуд ошибка границы (a, b),
row display error: пуст: возбуд ошибка записи массива,
nil error: пуст: возбуд ошибка псевдоимени,
scope error: пуст:возбуд ошибка области действия,
deadl ock: (сема s) пуст: возбуд тупик (s),
general exception: пуст: возбуд общая ситуация;

{Ситуация общая ситуация возбуждается, если не найдена процедура реакции для некоторой возбужденной ситуации.}

c) ситуация пуст исчерпание времени,
пуст исчерпание памяти,
пуст окончание,
пуст ? немедленное окончание;
ситуация пуст time exhaustion;
присит time exhaustion: пуст: возбуд исчерпание времени;
ситуация пуст space exhaustion;
присит space exhaustion: пуст: возбуд исчерпание памяти;
ситуация пуст termination;
присит termination: пуст: возбуд окончание;

{Ситуация окончание возбуждается в случаях неисправимых ошибок для того, чтобы дать программисту возможность обеспечить необходимые завершающие действия, определяя реакцию на эту ситуацию в некотором окружении. Однако предполагается, что исполнение этой реакции закончится тем, что то же самое исключение будет возбуждено вновь, чтобы обеспечить завершающие действия в старших окружениях.}

d) вид * имя массива = с фактический-описатель, специфицирующий вид, объединенный из {2.1.3.6.a} достаточного набора видов, каждый из которых начинается с 'имя массива' или 'имя подвижного массива' с;

e) ситуация массив образец массива,
массив образец получателя,
ситуация [ ] массив список образцов массива,
ситуация (имя массива) лог вырезано из,
ситуация цел размерность,
ситуация лог это имя;
ситуация массив row specimen,
массив destination specimen,
ситуация [ ] массив row specimen list,
ситуация (имя массива) лог isslice of,
ситуация цел dimension,
ситуация лог is name;

присит row specimen: массив: возбуд образец массива,
destination specimen: массив: возбуд образец получателя,
row specimen list: [ ] массив: возбуд список образцов массива.
is slice of: (имя массива а) лог: возбуд вырезано из (а),
dimension: цел: возбуд размерность,
is name: лог: возбуд это имя;
{Эти ситуации используются в некоторых процедурах восстановления для других ситуаций.}
f) проц закончить = пуст:
(с некоторое системное действие, помогающее идентифицировать текущее окружение с;
возбуд окончание:
возбуд немедленное окончание);
проц пуст terminate = закончить;
g) проц Д цел восстановление после переполнения = пуст:
(возбуд Д цел переполнение; закончить),
Д вещ восстановление после переполнения = пуст:
(возбуд Д вещ переполнение; закончить),
Д восстановление после потери значимости = пуст:
(возбуд Д потеря значимости; закончить),
Д цел восстановление после ошибки аргумента = пуст:
(возбуд Д цел ошибка аргумента; закончить),
Д вещ восстановление после ошибки аргумента = пуст:
(возбуд Д вещ ошибка аргумента; закончить);
h) проц восстановление после ошибки присваивания = (имямассива получатель, массив источник, цел n, i) пуст:
(присит образец получателя: массив:
с некоторый массив, паспорт которого идентичен паспорту значения, именуемого именем, выдаваемым идентификатором 'получатель' с,
образец массива: массив:
с некоторый массив, паспорт которого идентичен паспорту значения, выдаваемого идентификатором 'источник' с,
вырезано из: (имямассива r r), лог:
если с имя, выдаваемое идентификатором 'r r',
не было сгенерировано {2.1.3.4.j, 1} из другого имени с
то с истина, если любое подымя имени, выдаваемого идентификатором 'получатель', является подыменем имени, выдаваемого идентификатором 'r r', или же может быть получено из такого подымени в результате выполнения одной или нескольких последовательных операций выборки по 'СЛОВУ' {2.1.3.3.e}:
ложь в противном случае с
иначе пропуск
все.

```
                размерность: цел: n;
возбуд ошибка присваивания (i);
закончить);
i) проц восстановление после ошибки границы =
      (массив значение, цел n, i, граница) пуст:
      (присит образец массива: массив:
                  с некоторый массив, паспорт которого идентичен
                  паспорту значения, выдаваемого идентификатором
                  'значение' с.
               вырезано из: (имямассива rr) лог: пропуск,
               размерность: цел: n,
               это имя: лог: ложь;
   цел b: = граница;
   возбуд ошибка границы (i, b);
   b);
j) проц восстановление после ошибки границы имени =
      (имямассива имя, цел n, i, граница) пуст:
      (присит образец массива: массив:
            с некоторый массив, паспорт которого идентичен паспор-
            ту значения, именуемого именем, выдаваемым иденти-
            фикатором 'имя' с,
         вырезано из: (имямассива rr) лог:
            если с имя, выдаваемое идентификатором 'rr',
               не было сгенерировано {2.1.3.4. j, 1} из другого
               имени с
            то с истина, если любое подымя имени, выдаваемого
               идентификатором 'имя', является подыменем имя-
               ни, выдаваемого идентификатором 'rr', или же мо-
               жет быть получено из такого подымени в результате
               выполнения одной или нескольких последователь-
               ных операций выборки по 'СЛОВУ' {2.1.3.3.e};
                ложь в противном случае с
            иначе пропуск
            все.
                     размерность: цел: n.
                     это имя: лог: истина;
               цел b: = граница;
               возбуд ошибка границы (i, b);
               b);
k) проц восстановление после ошибки записи массива = ([ ] массив
      образец, цел n) пуст:
      (присит список образцов массива: [ ] массив: образец, размер-
         ность: цел: n;
      возбуд ошибка записи массива;
      закончить);
```

l) проц ↑ восстановление после ошибки псевдоимени =
пуст: (возбуд ошибка псевдоимени; закончить);
m) проц ↑ восстановление после ошибки области действия =
пуст: возбуд ошибка области действия; закончить);
n) проц ↑ восстановление после тупика =
(сема s) пуст: (возбуд тупик (s); закончить);
o) ситуация лог ↑ рекурсия общей ситуации; {см. 10.5.1.1}
проц ↑ восстановление после общей ситуации =
пуст: если рекурсия общей ситуации
то возбуд немедленное окончание
иначе присит рекурсия общей ситуации: лог:
истина; закончить
все;
p) проц ↑ восстановление после исчерпания времени =
пуст: (возбуд исчерпание времени; закончить);
q) проц ↑ восстановление после исчерпания памяти =
пуст: (возбуд исчерпание памяти; закончить);

10.3. Описания обмена

Прагматическое замечание соответствует ГОСТ 27974.

10.3.1. *Книги, каналы и файлы*

Прагматическое замечание соответствует ГОСТ 27974.

10.3.1.1. Книги и связки определены в ГОСТ 27974.

10.3.1.2 Каналы определены в ГОСТ 27974.

10.3.1.3 *Файлы.*

Прагматические замечания aa, bb, dd, ee, ff, gg, hh соответствуют ГОСТ 27974.

{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 истину, предполагается, что данное условие было некоторым образом исправлено, и, если возможно, обмен продолжается, но, если процедура реакции вырабатывает ложь, то предпринимается еще одна попытка восстановления путем возбуждения соответствующей ситуации. Это приводит к вызову еще одной процедуры, аналогичной по назначению процедуре реакции, но связанной с текущим окружением, а не с данным файлом. Если же и эта процедура 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 называется, 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 успеш-

ным 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 программистом, 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 текущие 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.c)).

- 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.

Описания a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s соответствуют ГОСТ 27974;

t) ситуация (имя файл) лог конец лог файла,
(имя файл) лог конец физ файла,
(имя файл) лог конец страницы,
(имя файл) лог конец строчки,
(имя файл) лог конец формата,
(имя файл) лог ошибка значения,
ситуация (имя файл, имя лит) лог ошибка литеры;

```
ситуация (имя файл) лог logical file end,
         (имя файл) лог physical file end,
         (имя файл) лог page end,
         (имя файл) лог line end,
         (имя файл) лог format end,
         (имя файл) лог value error,
ситуация (имя файл, имя лит) лог char error;
присит logical file end:
         (имя файл f) лог: возбуд конец лог файла (f),
         physical file end:
         (имя файл f) лог: возбуд конец физ файла (f),
         page end:
         (имя файл f) лог: возбуд конец страницы (f),
         line end:
         (имя файл f) лог: возбуд конец строчки (f),
         format end:
         (имя файл f) лог: возбуд конец формата (f),
         value error:
         (имя файл f) лог: возбуд ошибка значения (f),
         char error:
         (имя файл f, имя лит c) лог: возбуд ошибка литеры (f, c);
u) проц ^конец лог файла исправлен = (имя файл f) лог:
     если (испр лог файл из f) (f) то истина
     иначе возбуд конец лог файла (f)
     все;
   проц ^конец физ файла исправлен = (имя файл f) лог:
     если (испр физ файл из f) (f) то истина
     иначе возбуд конец физ файла (f)
     все;
   проц ^конец страницы исправлен = (имя файл f) лог:
     если (испр страница из f) (f) то истина
     иначе возбуд конец страницы (f)
     все;
   проц ^конец строчки исправлен = (имя файл f) лог:
     если (испр строчка из f) (f) то истина
     иначе возбуд конец сторочки (f)
     все;
   проц ^конец формата исправлен = (имя файл f) лог:
     если (испр формат из f) (f) то истина
     иначе возбуд конец формата (f)
     все;
   проц ^ошибка значения исправлена = (имя файл f) лог:
     если (испр ошибка значения из f) (f) то истина
     иначе возбуд ошибка значения (f)
     все;
```

```
    проц ошибка литеры исправлена = (имя файл f, имя лит c) лог:
      если (испр ошибка литеры из f) (f, c), то истина
      иначе возбуд ошибка литеры (f, c)
      все;
```

10.3.1.4. Открытие и закрытие файлов соответствуют ГОСТ 27974.

10.3.1.5. Запросы позиции соответствуют ГОСТ 27974.

10.3.1.6. Процедуры расположения.

Прагматические замечания и описания a, b, c, d, h, j, k, соответствуют ГОСТ 27974.

```
  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 ( ¬ конец строчки исправлен (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 ( ¬ конец страницы исправлен (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 оконч | физ файл окончен) (f)
      цк не оконч: = (чтение | конец лог файла исправлен | конец физ
      файла исправлен) (f) кц;
      не оконч
      конец;
  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 = заполи из книга из f;
       позиция раб тпоз = тпоз;
       если (тпоз: = (p, l, c)) вне ппоз
       то тпоз: = ппоз;
          ( ¬ конец лог файла исправлен (f) |
          не определено);
```

```
        настроить (f, чтение)
      инес позиция границы = границы книги (f);
       p < 1 ∨ p > p  из границы + 1 ∨
       l < 1 ∨ l > i  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;
```

10.3.2. Значения для обмена соответствуют ГОСТ 27974.

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 конец строчки исправлен (или, где это целесообразно, конец страницы исправлен, конец физ файла исправлен и.и конец лог файла исправлен), 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.

Прагматические замечания и описание с соответствуют ГОСТ 27974.

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) пусть 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 т поз = 1 | n | n + 1) >
           с из границы книги (f) + 1
        цк ( ¬ конец строчки исправлен (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 = т поз из f, п поз = заполн из книга из f;
    настроить на литерное (f); натроить на запись (f);
    имя цел p = p из тпоз, l = l из т поз, c = c из т поз;
    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 t 1) :  t1  [p]  [l]  [c] : = k,
          (подтекст t 2) : t2  [p]  [l]  [c] : = k
      быв;
      c + : = 1;
      если т поз вне  п поз  то  п поз : = т поз
      инес ¬ возм установка  (f)  1 /\
        позиция  (р из п поз, l из п поз, 1)  вне  т поз
      то п поз : = т поз;
        (сжимаем  (f) |
         с размер строчки и страницы, содержащей логический размер
            данной книги, и всех последующих строчек и страниц мо-
            жет увеличиться {например, до размеров, с которыми кни-
            га была заведена  (10.3.1.4.cc)  первоначально, или до раз-
            меров, предполагаемых из макс позиция из кан из f} с)
        все
      иначе k: = "⊥" ;
        если ¬ ошибка литеры исправлена  (f, k)
        то не определено;  k: = "⊥"
        все;
        проверить позицию  (f) ;  вывести литеру  (f, k)
    все
  иначе не определено
  все {настройка на запись сохраняется};
```

10.3.3.2. Бесформатный ввод

Прагматические замечания aa, bb, cc, dd, ee, ff, gg, соответствуют ГОСТ 27974.

{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 конец строчки исправлен (или, где это целесообразно, конец страницы исправлен, конец физ файла исправлен или конец лог файла исправлен); 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 присваивает-
              ся '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 с) лог : ? строк (с);
        прио! = 8;
        оп! = (строк s, лит с) лит:
          запрашивает литеру, содержащуюся в 's'; если читается литера,
            не входящая в 's', вызывается процедура обработки события,
            соответствующая 'при ошибке литеры', с предлагаемой литерой
            'с'.
        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: =  с;
           если ошибка литеры исправлена (f, предл)
             то
             (литера в строке (предл, лок цел, s)
             | предл | не определено; с)
           иначе не определено; с
           все;
           настроить на чтение (f)
        все;
        оп! = (лит s, с) лит: строк (s) !с;
        проц проп  уск  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 t: ="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 ⫽ ественное ⫽ = строк:
   (строк t: = чит знак;
   ( – ? " . " | t плюспр чит циф | k пусто: = ложь);
   ( ? "  " t плюспр " . " + чит циф);
   ( ? "10\c"|t плюспр "10" + чит чис); t);
для j до вегр у
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 изб) [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 (¬ ошибка значения исправлена (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 т поз = т поз из f;
       настроить на литерное (f); настроить на чтение (f);
       цел p = p из т поз, l = l из т поз, c = c из т поз;
       c из т поз + : = 1;
       лит: = выб текст из f в
         (текст t 1): t 1 [p] [l] [c],
         (подтекст t 2):
               (лит k: = t 2 [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 ошибка литеры исправлена (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: = конец строчки исправлен (f) кц
конец;

Прагматическое замечание соответствует ГОСТ 27974.

10.3.4. *Тексты формата*

Прагматическое замечание соответствует ГОСТ 27974.

10.3.4.1. Наборы и шаблоны.

10.3.4.1.1. Синтаксис.

Прагматическое замечание, метаправила A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P; гиперправила a, b, c, d, e, f, g, h, i, j, k, l, m, n, o и разъяснения aa, bb, cc, dd, ee, ff, jj, kk, соответствуют ГОСТ 27974.

(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 h 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 конец формата исправлен, 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 ошибка значения исправлена;
· 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 (см. соответствующий раздел),

• 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 ошибка значения исправлена;

• если она вырабатывает ложь, вызывается не определено:

• осуществляется вставка из P.

ll) Элемент множества литер S „ожидается" с предлагаемой литерой C следующим образом:

• читается некоторая литера;

если это одна из ожидаемых литер (т. е. принадлежит множеству S), то она подается,

а иначе вызывается процедура ошибка литеры исправлена с предлагаемой литерой C; если эта процедура вырабатывает истина, а (возможно, измененная) литера C есть одна из ожидаемых литер, то подается C, а иначе вызывается не определено .}

10.3.4.1.2. Семантика соответствует ГОСТ 27974.

10.3.4.2. Трафареты целого соответствуют ГОСТ 27974.

10.3.4.3. Трафареты вещественного соответствуют ГОСТ 27974.

10.3.4.4. Трафареты логического соответствуют ГОСТ 27974.

10.3.4.5. Трафареты комплексного соответствуют ГОСТ 27974.

10.3.4.6. Трафареты строкового соответствуют ГОСТ 27974.

10.3.4.7. Трафареты битового соответствуют ГОСТ 27974.

10.3.4.8. Трафареты выбора.

10.3.4.8.1. Синтаксис.

Гиперправила a, b, c соответствуют ГОСТ 27974.

{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 $V \geqslant 0$, и если число составляющих литералов в упакованном-списке-поясняемых-литералов этого C не меньше V,

· подготавливается и осуществляется литерал, выдаваемый V -м из этих литералов;

а иначе

· вызывается процедура ошибка значения исправлена;

· 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 ошибка значения исправлена;

· если она вырабатывает ложь, вызывается не определено;

· подготавливается и осуществляется вставка из P.

cc) Разъяснение соответствует ГОСТ 27974.

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 ошибка значения исправлена;

· 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, выдаваемых составляющими литералами из C;

Если вид этого N специфицируется посредством имя лог и первый (второй) литерал окажется искомым,

а иначе

· вызывается процедура ошибка значения исправлена;

· если она вырабатывает ложь, вызывается не определено;

· подготавливается и осуществляется вставка из P!}

10.3.4.8.2. Семантика соответствует ГОСТ 27974.

10.3.4.9. Трафареты форматного соответствуют ГОСТ 27974.

10.3.4.10. Трафареты бесформатного

10.3.4.10.1. Синтаксис.

Гиперправила a, b, c, d соответствуют ГОСТ 27974.

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 ошибка значения исправлена;
- если она вырабатывает ложь, V выводится с помощью вывод и вызывается не определено;
- осуществляется вставка из P.

bb) Разъяснение соответствует ГОСТ 27974.

10.3.4.10.2. Семантика соответствует ГОСТ 27974.

10.3.5. *Форматный обмен*

Описания a, c, d, e, f, g, j, k соответствуют ГОСТ 27974.

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 ¬ конец формата исправлен (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 фбрмат 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 (в 1 из пак)] подвставка пв;
        оук из алеф [удк из пак ]: = укфор; укфор: = пропуск;
        (подготовить вставку (в 1 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: 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]] ,
          (пакет пак):
            (оук из алеф [удк из пак]: = 0; в 2 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;
h) проц ввести вставку = (имя файл f, [ ] подставка пв) пуст:
    начало настроить на чтение (f);
    для k до вегр пв
    цк
      выб стр из пв [k] в
        (лит а) : разместить (f, повт из пв [k], а, 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]
                  (¬ ошибка литеры исправлена  (f, c:=s [i ] )
                  |не определено);
                  настроить на чтение (f))
                  кц
                кц)
              быв
            кц
          конец;
    i) проц ? разместить = (имя файл f, цел повт, лит a, лог чит) пуст):
        если a = "x" то до повт цк вперед (f) кц
        инес a = "y" то до повт цк назад (f) кц
        инес a = "]" то до повт цк нов строчка (f) кц
        инес a = "p" то до повт цк нов страница (f) кц
        инес a = "k" то уст номер литеры (f, повт)
        инес a = "q"
        то до повт
          цк
            если чит
            то лит c; проверить позицию (f); ввести литеру (f, c);
              (c ≠ пробел|
               (¬ ошибка литеры исправлена  (f, c:=пробел)
               |не определено); настроить на чтение (f))
               иначе проверить позицию (f); вывести литеру (f, пробел)
               все
            кц
          все;
```

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 ⅈ азатель ⅈ рам ⅈ ок ⅈ: = 1;
      [1: вегр (рамки из трафарет)] подрамка подрамки;
      ( подготовить. рамки (рамки из трафарет, подрамки),
        подготовить вставку (в из шаблон, подвест));
    строк s;
   оп? = (строк s) лог:
  ⅈ истина, если следущий маркер есть один из элементов
    's', а иначе ложь ⅈ
  если украм > вегр подрамки
  то ложь
  иначе подрамка пр = подрамки [украм];
     повт: = повт из пр;
     если литера в строке (марк из пр, лок 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 ⅈ активность ⅈ 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- вегр t) X "0" прип s;
      (образец знака |(i < Д0| "–" | "+") прип s)
   все) ≯;
  ≮ проц 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 (y, b, a, пор);
       ред цел (пор);
        "ю" 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 + : β = 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 ∉
       (y [j] |
         ∢ (Д компл дк): ред Д компл (дк) ↗,
         ∢ (Д вещ дв): ред Д вещ (дв) ↗,
```

```
        ⊄ (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 [с 1]
       |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 [1], спец [3]) , ( ) )):
        вывести вставку (f, пв):
        если n = 0 то вывод (f, y [j])
            иначе
              число yj =
                (y [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):
          (¬ошибка значения исправлена (f) | вывод (f, y [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) Описание соответствует ГОСТ 27974.

10.3.5.2. Форматный ввод.

```
а) проц ф ввод = (имя файл f, [ ] об (вводимое, формат) х) пуст:
    если открыт из f то
    для k до вегр х
    цк выб настроить на чтение (f);
    настроить на литерное (f): х [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 b 1, b 2;  литера в строке ("1 ", i, s);
               b1: = строку в Д вещ (s [: i – 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 из 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) х 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: = кк = s [jj] кц;
          (¬ (есть: = лог) |уст номер литеры (f, с))
```

```
                кц;
                если ¬ есть то не конч: = истина
                иначе
                    выб тип из выбор в
                ≠ логическое ≠
                    (у [j] |
                    (имя цел 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 подв ст 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);
                 (¬ ошибка значения исправлена (f) |
                 не определено)
             все;
             ввести вставку (f, подвст)
        кц
      конец
    быв кц
    иначе не определено
    все;
proc (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, п ≠одавляемые ≠ н ≠ули ≠: = истина, есть знак: = ложь,
```

```
    есть пробел: = ложь, нет знака: = ложь,
    цел ук ц азатель ц эн ц ака ц: = 1, повт;
  прио! = 8;
  оп! = (строк s, лит c) лит:
    ц запрашивает некоторую литеру, содержащуюся в 's'; если читае-
    мая литера не входит в 's', то вызывается процедура обработки со-
    бытия, соответствующая при ошибке литеры, с предлагаемым 's' ц
    если лит k; проверить позицию (f); ввести литеру (f, k) литера
    в строке (k, лок цел, s)
    то
    иначе лит предл: = c;
    если ошибка литеры исправлена (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 c = ((пн | "-" | " ") + хор цифры) ! "0";
             (c ≠ "-" | пн: = ложь); c)
       инес маркер = "u" ∨ маркер = "+" то
         если есть знак
         то пн: = ложь; s плюспр ("0123456789"! "0")
         иначе лит c= ("+ -" + (маркер = "u" | "-" | " " )) ! "+";
           (c = "+" ∨ c = "-" | есть знак: = истина; s [укзн]: = c)
         все
       инес маркер = "v" ∨ маркер = "- " то
         если есть знак
         то пн: = ложь; s плюспр ("0123456789"! "0")
         инес лит c; есть пробел
         то c: = " + - . 0123456789"! "+";
             (c = "+" ∨ c = "-" | есть знак: = истина; s [укзн]: = c
             | c: ≠ "-" | пн: = ложь; есть знак: = истина;
             s плюстр c)
         иначе c: = "+ - . " ! "+";
             (c = "+" ∨ c = "-" | есть знак: = истина; s [укзн]: = c
```

```
        |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 |"1" | "i1" |"1", ");
      есть знак: =ложь; пн: =истина;
      s плюспр "+"; укзн: =вегр s
    инес маркер = "b" то
      s плюспр (да + нет) ! нет; нет знака: =истина
        инес маркер = "a" то s плюспр
          (подав |"." | лит с; проверить позицию (f);
          вывести лйтеру (f, с);
          с);
          нет знака: =истина
        инес маркер = "r"
        то пропуск
        все
      кц
    кц;
    если нет знака то s = s [2:] все
  конец;
```

10.3.6. Двоичный обмен соответствует ГОСТ 27974.

10.4. Системное вступление и список задач соответствуют ГОСТ 27974.

10.5. Собственные вступления и заключения

10.5.1. *Собственные вступления*

Описания a, b, c, d, e, f, g, h, i, соответствуют ГОСТ 27974.

j) присит окончание: пуст: стоп;

k) присит немедленное окончание: пуст: стоп;

l) присит рекурсия общей ситуации: лог: ложь;

m) присит конец лог файла: (имя файл f) лог: ложь,
конец физ файла: (имя файл f) лог: ложь,
конец страницы: (имя файл f) лог: ложь,
конец строчки: (имя файл f) лог: ложь,
конец формата: (имя файл f) лог: ложь,
ошибка значения: (имя файл f) лог: ложь,
ошибка литеры: (имя файл f, имя лит с) лог; ложь;

10.5.2. Собственные заключения определены в ГОСТ 27974.

10.6. Сегменты

10.6.1. *Синтаксис*

а) вставляемый сегмент на АЛГОЛЕ 68
в СРЕДЕ с новыми ?МОДУЛЯМИ выдающий ЗНАЧЕНИЕ {A7a}:
знак сегмент {94 d} ,

индикатор заготовки {56 d},
знак определяется как {94 d},
фактическая заготовка в СРЕДЕ с новыми ?МОДУЛЯМИ выдающая ЗНАЧЕНИЕ {56 c}.

b) Для каждого дополнительного {5.6.1.A} терминального, метапорождения „ЯЗЫКА" следует добавить дополнительные гиперправила для гиперпонятий формы „вставляемый сегмент на ЯЗЫКЕ в СРЕДЕ с новыми ?МОДУЛЯМИ выдающий ЗНАЧЕНИЕ". Должен быть определен механизм {с помощью стандарта, определяющего этот другой язык}, посредством которого все такие вставляемые-сегменты-на ЯЗЫКЕ могут быть преобразованы во вставляемые-сегменты-на АЛГОЛЕ 68 {с тем же самым смыслом}.

c) сегмент определяющий модули через !МОДУЛИ
в СРЕДЕ с новыми ?МОДУЛЯМИ1 !МОДУЛЯМИ {A7a}:
знак сегмент {94 d},
индикатор заготовки {56 d},
знак определяется как {94 d},
описание модулей через !МОДУЛИ
в СРЕДЕ с новыми ?МОДУЛЯМИ1 !МОДУЛЯМИ {49a},
если !МОДУЛИ не охвачены СРЕДОЙ {e}.

d) сегмент для вступления задающий !МОДУЛИ
в новом СЛОЕ1 с новыми !ОПИСАНИЯМИ ?МОДУЛЯМИ1
!МОДУЛЯМИ {и} СТОПОМ {A7a}:
описание модулей через !МОДУЛИ в новом СЛОЕ1
с новыми !ОПИСАНИЯМИ ?МОДУЛЯМИ1 !МОДУЛЯМИ
{и} СТОПОМ {49a},
если !МОДУЛИ не охвачены новым СЛОЕМ1 {e}.

e) ЕСЛИ ?МОДУЛИ МОДУЛЬ не охвачены СРЕДОЙ {c,d}:
ЕСЛИ ?МОДУЛИ не охвачены СРЕДОЙ {e,f}
и МОДУЛЬ не зависит от ?ПАР {71a, b, c},
если в ?ПАРАХ собраны свойства из СРЕДЫ {g, h}.

f) ЕСЛИ ПУСТО не охвачено СРЕДОЙ {e}: ЕСЛИ истина.

g) ЕСЛИ в ?ПАРАХ1 ?ПАРАХ2 собраны свойства
из СРЕДЫ с новыми ?ПАРАМИ2 {e, g}:
ЕСЛИ в ?ПАРАХ1 собраны свойства из СРЕДЫ {g, h}.

h) ЕСЛИ в ПУСТО собраны свойства из нового ПУСТО {e, g}:
ЕСЛИ истина.

i) * сегмент в СРЕДЕ с новыми ?ПАРАМИ:
вставляемый сегмент на ЯЗЫКЕ
в СРЕДЕ с новыми ?ПАРАМИ выдающий ЗНАЧЕНИЕ {a, b};
сегмент определяющий модули через !МОДУЛИ
в СРЕДЕ с новыми ?ПАРАМИ {c};
собственно-программа в СРЕДЕ с новыми ?ПАРАМИ {A1 g};
сегмент для вступления задающий !МОДУЛИ
в СРЕДЕ с новыми ?ПАРАМИ {d}.

j) * символ буква: символ БУКВА {94a}.

k) * символ цифра · символ ЦИФРА {94b} .

{ Примеры:

a) сегмент "a b c" = подкл a, b (x: = 1; y: = 2; печ (x + y))

c) сегмент "a b c" =модуль a = мд откр вещ x дм

d) модуль b = мд откр вещ g дм

Эти три примера могли бы представить собой согласованный набор сегментов (10.6.2.a), будучи объединены с собственно-программой

начало среда "a b c" конец }

{ В вышеприведенном правиле a 'МОДУЛИ', заложенные в '?МОДУЛИ', определяются всеми сегментами-определяющими-модули, которые должны вставляться вместе с данным вставляемым-сегментом. В правилах c и d требуется только, чтобы в '?МОДУЛИ1' были заложены 'МОДУЛИ' для всех тех модулей, которые на самом деле доступны изнутри данного сегмента. Нижеследующая семантика определена только в тех случаях, когда для набора совместно вставляемых сегментов, все 'МОДУЛИ', заложенные в различные '?МОДУЛИ1', заложены и в '?МОДУЛИ'}

10.6.2. *Семантика*

{ Сегменты – это компоненты для раздельной трансляции. Необходимо определить смысл набора сегментов. Это осуществляется преобразованием набора в эквивалентную собственно-программу. Необходимо, конечно, чтобы сегменты в наборе были согласованы друг с другом. Ровно один сегмент в наборе должен быть собственно-программой. }

a) Смысл собственно-программы P в контексте набора T других присоединяемых сегментов { , не являющихся собственно-программами, } определяется следующим образом:

· Личное вступление-с-?МОДУЛЯМИ UP из задачи-пользователя UT, наследником которой является P { 1.1.1.e и 10.1.1.f } , должно быть составлено следующим образом:

Для каждого сегмента-для-вступления-в-новом-СЛОЕ1-с-новыми-!ОПИСАНИЯМИ-?МОДУЛЯМИ1- { и= } СТОПОМ M, если таковое имеется, в T,

· UP содержит составляющее описание-модулей-в-новом-СЛОЕ1-с-новыми-!ОПИСАНИЯМИ-?МОДУЛЯМИ- { и= } СТОПОМ, подобное описанию-модулей из M; { в '?МОДУЛИ' должны быть заложены все 'МОДУЛИ', заложенные во все такие и только такие '?МОДУЛИ1', которые синтаксически корректны по отношению к личному-вступлению из UT; }

· UP не содержит никаких иных составляющих описаний-ОБЪЕКТОВ, и его единственная составляющая основа состоит из пропуска { 5.5.2.1.a } ;

Если T содержит какие-либо вставляемые-сегменты-на-ЯЗЫКЕ, где 'ЯЗЫК' не есть 'АЛГОЛ68',

то эти сегменты преобразуются { 10.6.1.b } во вставляемые-сегменты-на-АЛГОЛЕ 68 { с тем же самым смыслом };

Пока в UT остаются какие-либо формальные-заготовки,

· пусть H есть такая формальная-заготовка-в-СРЕДЕ-выдающая-ЗНАЧЕНИЕ, и пусть I есть индикатор-заготовки;

· если I подобен какому-либо такому I, который уже рассматривался, то смысл P не определяем;

· H замещается {в UT} виртуальной-заготовкой-в-СРЕДЕ-выдающей-ЗНАЧЕНИЕ, составляющее последовательное-предложение-в-СРЕДЕ S которой составляется следующим образом:

Для каждого сегмента-определяющего-модуля-в-СРЕДЕ-с-новыми ?МОДУЛЯМИ1 M, если таковой имеется, в T, индикатор-заготовки которого „соответствует" { b } I,

· S содержит составляющее описание-модулей-в-СРЕДЕ-с-новыми-?МОДУЛЯМИ, подобное описанию-модулей из M;

{в '?МОДУЛИ' должны быть заложены все 'МОДУЛИ', заложенные во все такие и только такие '?МОДУЛИ1', которые синтаксические корректны по отношению к S;}

· S не содержит никаких иных составляющих описаний-ОБЪЕКТОВ, и его единственная составляющая основа состоит из составляющего ЗАКРЫТОГО-предложения из { единственного } вставляемого-сегмента-на-АЛГОЛЕ 68-в-СРЕДЕ-с-новыми-?МОДУЛЯМИ-выдающего-ЗНАЧЕНИЕ из T, индикатор-заготовки которого соответствует I;

Если в T остались какие-либо сегменты, которые не были включены в UT,

то смысл P не определен ;

иначе { в UT не содержится никаких формальных-заготовок, и поэтому } смысл P таков, каким он определен в другом месте { 1.1.1.e} семантикой стандарта.

b) Если { текстуально } первый составляющий элемент-строки из индикатора-заготовки I состоит из некоторого символа-буква, и всякий другой составляющий элемент-строки, если таковой имеется, состоит из некоторого символа-буква или некоторого символа-цифра, то I „соответствует" любому другому индикатору-заготовки, которому он подобен {; иначе его соответствие другим индикаторам-заготовок (подобным ему или нет) здесь не определяется, однако, оно может определяться местными соглашениями реализации, учитывающими особенности местной операционной языковой обстановки.}

{ Стандартная языковая обстановка расширяется включением личного-вступления для каждой собственно-программы, в которую пользователь может вставить свои собственным сегменты-для-вступления. }

ПРИЛОЖЕНИЕ 1
Справочное

ИСТОРИЧЕСКАЯ СПРАВКА

„Пересмотренное сообщение об алгоритмическом языке Алгол 68", официально принятое в 1974 г. в качестве окончательного определяющего документа для языка

Алгол 68, не подлежало изменениям, но развитие программирования показало необходимость включения в язык дополнительных средств, обеспечивающих:

раздельную трансляцию фрагментов программ;

модульное программирование;

обработку исключительных ситуаций.

Были выдвинуты многочисленные предложения по реализации этих средств и некоторые из них реализованы.

В 1977 г. были опубликованы предложения Ч. Линдси и Х. Боома по модулям и раздельной трансляции „A Modules and Separate Compilation Facility for ALGOL 68".

Предложения были одобрены рабочей группой 2.1 (РГ2.1) Международной федерации по обработке информации (ИФИП) и, во избежание чрезмерного распространения диалектов, рекомендованы для преимущественного использования в реализациях, вводящих подобные средства. Настоящий стандарт включает предложения Ч. Линдси и Х. Боома в переводе В.В. Броля.

Предложений по обработке исключительных ситуаций, имеющих такой же статус, как упомянутые выше, в настоящее время не существует. В связи с этим рабочая группа по алгоритмическому языку Алгол 68 (РГ А68) научно-технической комиссии по системам математического обеспечения при ГКВТ АН СССР решила разработать оригинальный вариант этих средств. Механизм обработки исключительных ситуаций был предложен Г.С. Цейтиным. Это предложение обсуждалось на заседаниях РГ А68, было одобрено и вошло в настоящий стандарт. От имени РГ 2.1 ИФИП это предложение рассматривал Ч. Линдси, сделавший ряд полезных замечаний.

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 " # $ % ' ( ) * + , – . / : ; < = > @ [ ] _ |

Представленные программы на Алголе 68 определяют как разделенную на строчки последовательность абстрактных литер.

1.2. Конкретная литера – эта 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 (подраздел 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 строчки литер с о п р и к а с а ю т с я, если между ними нет литер или особенностей типографского набора. Если одна из строк литер с л е д у е т з а или п р е д ш е с т в у е т другой, то они также соприкасаются

1.5. В ы д е л е н н о е с л о в о – 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. С л о в о – это символ-СЛОВО (подпункт 9.4.2.2.а), например, – „конец файла" – это слово.

1.7. С л о г – 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 „!" 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 данного приложения, а также введение альтернативных обозначений из разд. 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. Э л е м е н т ы с т р о к и

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. Символ-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 устройстве ввода-вывода, 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-ПОЯСНЕНИЯ-ОФОРМЛЕННЫХ (п. 9.2.1.с) 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 Алгола 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-строках.

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 „конец файла".

3.4.1.1. В режиме выделения точкой (ТЧК) выделенные слова представляются 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 элемента-основного-набора (подпункт 8.1.4.1.с).

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 символов языка), составленными из букв русского алфавита

БИТ БЫВ В ВЕЩ ВИД ВОЗБУД ВСЕ ВЫБ ВЫХОД ГЛОБ ДЛИН ДЛЯ ДМ ДО ЕСЛИ ЕСТЬ ИЗ ИМЕНИ ИМЯ ИНАЧЕ ИНЕС ИСТИНА КАНАЛ КОМПЛ КОН КОНЕЦ КОР КЦ ЛИБО ЛИВЫБ ЛИТ ЛОГ ЛОЖЬ ЛОК МД МОДУЛЬ НА НАЧ НАЧАЛО НЕСТЬ НИЛ ОБ ОП ОТ ОТКР ПАР ПОДВ ПОДКЛ ПОКА ПРАГМ ПРИМ ПРИО ПРИСИТ ПРОПУСК ПРОЦ ПУСТ ПУСТОБ С СБРОС СЕМА СЕГМЕНТ СИТУАЦИЯ СКИП СЛОГ СРЕДА СТ СТРУКТ ТО Ф ФАЙЛ ФОРМАТ ЦЕЛ ЦК ЧЕРЕЗ ШАГ

и букв латинского алфавита

ACCESS AT BEGIN BITS BOOL BY BYTES CASE CHANNEL CHAR CO COMMENT COMPL DEF DO EGG ELIF ELSE EMPTY END ESAC ECCAPTION EXIT FALSE FED FI FILE FLEX FOR FORMAT FROM GO GOTO HEAP IF IN INT IS ISNT LOC LONG MODE NEST NIL OD OF ON OP OUSE OUT PAR POSTLUDE PR PRAGMAT PRIO PROC PUB RAISE REAL REF SEMA SHORT 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-ПОНЯТИЕ2, (символы-ПОНЯТИЕ1-перед-ПОНЯТИЕМ2 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. Представления объектов для обмена должны использовать только абстрактные литеры (с тем, чтобы ввод можно было подготавливать, а вывод интерпретировать без ссылки на конкретную реализацию). Запросы к обстановке зависят от абстрактных литер следующим образом:

да „ Т " 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 „ Е " { 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.

ИНФОРМАЦИОННЫЕ ДАННЫЕ

1. ИСПОЛНИТЕЛИ

В.П. Морозов, д-р техн. наук; **Г.С. Цейтин**, д-р физ.-мат. наук; **А.Ф. Рар**; **А.Н. Терехов**, канд. физ.-мат. наук; **Ч. Линдси**; **О.Е. Климова**; **Н.Б. Скачков**; **В.В. Броль**; **В.Б. Яковлев**; **И.Б. Гиндьш**; **Э.В. Оленьева**; **Д.Ю. Жуков**; **Ю.И. Карпов**; **Ф.Х. Кабалина**; **О.К. Александрова**; **Е.Г. Грозная**

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 пункта, приложения
ГОСТ 27465–87 ГОСТ 27974–88	Приложение 2 (п. 3.1.1) 1, 1.1.1 –1.1.4,1.1.4.1– 1.1.4.3,1.2.1,1.2.3,1.3,2.1.1,2.1.1.1–2.1.1.3,2.1.2,2.1.3,2.1.4.1–2.1.4.3,2.1.5,2.2.3,3.0.1,3.0.2,3.1, 3 1,3.2.1,4,4.1.1,3.3.1,3.3.2,3.4,3.5,4.1.2,4.2–4.7,4.8.1,4.8.2, 5,5.1,5.2.1.1,5.2.1.2,5.2.2–5.2.4,5.3.1.1,5.3.2.1,5.3.2.2,5.4.1–5.4.4,5.5,6,6.1.1,6.2.1,6.3–6.7,7,7.1.1,7.2.1,7.2.2,7.3,7.4,8,9, 9,1.–9.4,9.4.1,9.4.2,10,10.1,10.1.1,10.1.2,10.1.3,10.2,10.2.2, 10.2.3.0 – 10.2.3.12, 10.2.4, 10.3, 10,3.1, 10.3.1.1, 10.3.1.2, 10.3.1.3–10.3.1.6,10.3.2,10.3.3.1,10.3.3.2,10.3.4,10.3.4.1.1, 10.3.4.1.2,10.3.4.2–10.3.4.7,10.3.4.8.1,10.3.4.8.2,10.3.4.9, 10.3.4.10.1, 10.3.4.10.2, 10.3.5, 10.3.5.1, 10.3.6, 10.4, 10.5.1, 10.5.2

СОДЕРЖАНИЕ

ГОСТ 27974–88

1. Язык и метаязык 4

1.1. Метод описания 4
1.1.1. Введение 4
1.1.2. Прагматика 5
1.1.3. Синтаксис строгого языка 6
1.1.4. Семантика 14
1.2. Общие метаправила 18
1.2.1. Метаправила для видов 18
1.2.2. Метаправила, связанные с фразами и приведением 19
1.2.3. Метаправила, связанные со средами 19
1.2. Общие гиперправила 19
1.3.1. Синтаксис общих предикатов 19
1.3.2. Выполнимость предикатов 20
1.3.3. Синтаксис общих конструкций 21

2. Вычислитель и программа 22

2.1. Терминология 22
2.1.1. Объекты 22
2.1.2. Соотношения 25
2.1.3. Значения 26
2.1.4. Действия 33
2.1.5. Сокращения 35
2.2. Программа 37
2.2.1. Синтаксис 37
2.2.2. Семантика 37

3. Предложения 38

3.0.1. Синтаксис 38
3.0.2. Семантика 39
3.1. Замкнутые предложения 39
3.1.1. Синтаксис 39
3.2. Последовательные предложения 40
3.2.1. Синтаксис 40
3.2.2. Семантика 42
3.3. Совместные и параллельные предложения 43
3.3.1. Синтаксис 43
3.3.2. Семантика 45
3.4. Выбирающие предложения 45
3.4.1. Синтаксис 47
3.4.2. Семантика 50
3.5. Циклические предложения 50
3.5.1. Синтаксис 51
3.5.2. Семантика 52

4. Описания, описатели и индикаторы 54

4.1. Описания 54
4.1.1. Синтаксис 54
4.1.2. Семантика 55
4.2. Описания видов 55
4.2.1. Синтаксис 55
4.2.2. Семантика 55
4.3. Описания приоритетов 56
4.3.1. Синтаксис 56

4.3 2. Семантика . . . 56
4 4. Описания идентификаторов . . . 56
4.4.1. Синтаксис . . . 57
4.4.2. Семантика . . . 58
4.5. Описания операций . . . 59
4.5.1. Синтаксис . . . 59
4.5.2. Семантика . . . 59
4.6. Описатели . . . 59
4.6.1. Синтаксис . . . 60
4.6.2. Семантика . . . 62
4.7. Соотношения между видами . . . 64
4.7.1. Синтаксис . . . 64
4.8. Индикаторы и указатели полей . . . 65
4.8.1. Синтаксис . . . 65
4.8.2 Семантика . . . 66

5. Основы . . . 67
5.1. Синтаксис . . . 67
5.2. Основы, связанные с именами . . . 68
5.2.1. Присваивания . . . 68
5.2.2. Отношения одноименности . . . 69
5.2.3. Генераторы . . . 70
5.2.4. Псевдоимена . . . 71
5.3. Основы, связанные с составными значениями . . . 71
5.3.1. Выборки . . . 71
5.3.2. Вырезки . . . 72
5.4. Основы, связанные с процедурами . . . 75
5.4.1. Тексты процедур . . . 75
5.4.2. Формулы . . . 77
5.4.3. Вызовы . . . 78
5.4.4. Переходы . . . 79
5.5. Основы, связанные со значениями любого вида . . . 80
5.5.1. Ядра . . . 80
5.5.2. Пропуски . . . 80

6. Приведение . . . 81
6.1. Приведенные . . . 81
6.1.1. Синтаксис . . . 82
6.2. Разыменование . . . 83
6.2.1. Синтаксис . . . 84
6.2.2. Семантика . . . 84
6.3. Распроцедуривание . . . 84
6.3.1. Синтаксис . . . 84
6.3.2. Семантика . . . 84
6.4. Объединение . . . 84
6.4.1. Синтаксис . . . 85
6.5. Обобщение . . . 85
6.5.1. Синтаксис . . . 85
6.5.2. Семантика . . . 86
6.6. Векторизация . . . 86
6.6.1. Синтаксис . . . 87
6.6.2. Семантика . . . 87
6.7. Опустошение . . . 88
6.7.1. Синтаксис . . . 88
6.7.2. Семантика . . . 88

7. Виды и среды . . . 88
7.1. Независимость свойств . . . 89
7.1.1. Синтаксис . . . 89

7.2. Идентификация в средах . . . 92
7.2.1. Синтаксис . . . 92
7.2.2. Семантика . . . 92
7.3. Эквивалентность видов . . . 94
7.3.1. Синтаксис . . . 94
7.4. Правильность построения . . . 98
7.4.1. Синтаксис . . . 98

8. Изображения . . . 99

8.0.1. Синтаксис . . . 99
8.1. Изображения простого . . . 99
8.1.1. Изображения целого . . . 100
8.1.2. Изображения вещественного . . . 100
8.1.3. Изображения логического . . . 101
8.1.4. Изображения литерного . . . 101
8.1.5. Изображение пустого значения . . . 102
8.2. Изображения битового . . . 102
8.2.1. Синтаксис . . . 102
8.2.2. Семантика . . . 104
8.3. Изображения строки . . . 104
8.3.1. Синтаксис . . . 104
8.3.2. Семантика . . . 105

9. Знаки и символы . . . 105

9.1. Знаки . . . 105
9.1.1. Синтаксис . . . 106
9.2. Примечания и прагматы . . . 107
9.2.1. Синтаксис . . . 106
9.3. Представления . . . 107
9.4. Эталонный язык . . . 108
9.4.1. Представления символов . . . 109
9.4.2. Символы прочих обозначений . . . 115

10. Стандартная языковая обстановка . . . 117

10.1. Тексты программ . . . 117
10.1.1. Синтаксис . . . 117
10.1.2. Соответствие языковой обстановке . . . 119
10.1.3. Способ описания стандартной языковой обстановки . . . 120
10.2. Стандартное вступление . . . 122
10.2.1. Запросы к обстановке . . . 122
10.2.2. Стандартные виды . . . 124
10.2.3. Стандартные обозначения операций и функции . . . 124
10.2.4. Операции синхронизации . . . 133
10.3. Описание обмена . . . 134
10.3.1. Книги, каналы и файлы . . . 134
10.3.2. Значения для обмена . . . 155
10.3.3. Бесформатный обмен . . . 162
10.3.4. Тексты формата . . . 171
10.3.5. Форматный обмен . . . 191
10.3.6. Двоичный обмен . . . 207
10.4. Системное вступление и список задач . . . 209
10.4.1. Системное вступление . . . 209
10.4.2. Список системных задач . . . 209
10.5. Собственные вступления и заключения . . . 210
10.5.1. Собственные вступления . . . 210
10.5.2. Собственные заключения . . . 211
Приложение 1. Историческая справка . . . 211
Приложение 2. Требования к машинному представлению программы . . . 213

Приложение 3. Указатель применяемых в стандарте понятий . . . 217
Приложение 4. Список метаправил . . . 239
Информационные данные . . . 244

ГОСТ 27975–88

1. Язык и метаязык . . . 246
1.1. Метод описания . . . 246
1.1.1. Введение . . . 246
1.1.2. Прагматика . . . 246
1.1.3. Синтаксис строгого языка . . . 246
1.1.4. Семантика . . . 246
1.2. Общие метаправила . . . 246
1.2.1. Метаправила для видов . . . 246
1.2.2. Метаправила, связанные с фразами и приведением . . . 246
1.2.3. Метаправила, связанные со средами . . . 247
1.3. Общие гиперправила . . . 247
2. Вычислитель и программа . . . 247
2.1. Терминология . . . 247
2.1.1. Объекты . . . 247
2.1.2. Соотношения . . . 247
2.1.3. Значения . . . 248
2.1.4. Действия . . . 248
2.1.5. Сокращения . . . 248
2.2. Программа . . . 248
3. Предложения . . . 249
3.0.1. Синтаксис . . . 249
3.0.2. Семантика . . . 249
3.1. Замкнутые предложения . . . 249
3.2. Последовательные предложения . . . 249
3.2.1. Синтаксис . . . 249
3.2.2. Семантика . . . 249
3.3. Совместные и параллельные предложения . . . 251
3.3.1. Синтаксис . . . 251
3.3.2. Семантика . . . 251
3.4. Выбирающие предложения . . . 252
3.5. Циклические предложения . . . 252
3.6. Подключающие предложения . . . 252
3.6.1. Синтаксис . . . 252
3.6.2. Семантика . . . 254
4. Описания, описатели и индикаторы . . . 256
4.1. Описания . . . 256
4.1.1. Синтаксис . . . 256
4.1.2. Семантика . . . 257
4.2. Описания видов . . . 257
4.3. Описания приоритетов . . . 257
4.4. Описания идентификаторов . . . 257
4.5. Описания операций . . . 257
4.6. Описатели . . . 257
4.7. Соотношения между видами . . . 257
4.8. Индикаторы и указатели полей . . . 257
4.8.1. Синтаксис . . . 257
4.8.2. Семантика . . . 257
4.9. Описание модулей . . . 257
4.9.1. Синтаксис . . . 257

4.9.2. Семантика . . . 260
4.10. Ситуации и реакции . . . 260
4.10.1. Синтаксис . . . 260
4.10.2. Семантика . . . 261

5. Основы . . . 261

5.1. Синтаксис . . . 261
5.2. Основы, связанные с именами . . . 262
5.2.1. Присваивания . . . 262
5.2.2. Отношения одноименности . . . 262
5.2.3. Генераторы . . . 263
5.2.4. Псевдоимена . . . 263
5.3. Основы, связанные с составными значениями . . . 263
5.3.1. Выборки . . . 263
5.3.2. Вырезки . . . 263
5.4. Основы, связанные с процедурами . . . 264
5.4.1. Тексты процедур . . . 264
5.4.2. Формулы . . . 264
5.4.3. Вызовы . . . 264
5.4.4. Переходы . . . 264
5.4.5. Вызовы ситуаций . . . 264
5.5. Основы, связанные со значениями любого вида . . . 265
5.6. Заготовки . . . 265
5.6.1. Синтаксис . . . 265

6. Приведение . . . 266

6.1. Приведенные . . . 266
6.1.1. Синтаксис . . . 266
6.2. Разыменование . . . 266
6.2.1. Синтаксис . . . 266
6.2.2. Семантика . . . 266
6.3. Распроцедуривание . . . 267
6.4. Объединение . . . 267
6.5. Обобщение . . . 267
6.6. Векторизация . . . 267
6.7. Опустошение . . . 267

7. Виды и среды . . . 267

7.1 Независимость свойств . . . 267
7.1.1. Синтаксис . . . 267
7.2. Идентификация в средах . . . 267
7.2.1. Синтаксис . . . 267
7.2.2. Семантика . . . 268
7.3. Эквивалентность видов . . . 268
7.4. Правильность построения . . . 268

8. Изображения . . . 268

9. Знаки и символы . . . 268

9.1. Знаки . . . 268
9.2. Примечания и прагматы . . . 268
9.3. Представления . . . 268
9.4. Эталонный язык . . . 268
9.4.1. Представления символов . . . 269
9.4.2. Символы прочих обозначений . . . 271

10. Стандартная языковая обстановка . . . 271

10.1. Тексты программ . . . 271
10.1.1. Синтаксис . . . 271

10.1.2. Соответствие языковой обстановке . . . 271
10.1.3. Способ описания стандартной языковой обстановки . . . 271
10.2. Стандартное вступление . . . 271
10.2.1. Запросы к обстановке . . . 271
10.2.2. Стандартные виды . . . 273
10.2.3. Стандартные обозначения операций и функций . . . 273
10.2.4. Операции синхронизации . . . 274
10.2.5. Стандартные ситуации и восстанавливающие действия . . . 275
10.3. Описания обмена . . . 279
10.3.1. Книги, каналы и файлы . . . 279
10.3.2. Значения для обмена . . . 284
10.3.3. Бесформатный обмен . . . 284
10.3.4. Тексты формата . . . 290
10.3.5. Форматный обмен . . . 293
10.3.6. Двоичный обмен . . . 304
10.4. Системное вступление и список задач . . . 304
10.5. Собственные вступления и заключения . . . 304
10.5.1. Собственные вступления . . . 304
10.5.2. Собственные заключения . . . 304
10.6. Сегменты . . . 304
10.6.1. Синтаксис . . . 304
10.6.2. Семантика . . . 306
Приложение 1. Историческая справка . . . 307
Приложение 2. Требования к машинному представлению программ . . . 308
Информационные данные . . . 313
Содержание . . . 314

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

АЛГОЛ 68 И АЛГОЛ 68 РАСШИРЕННЫЙ

ГОСТ 27974–88, ГОСТ 27975–88

Редактор *В.П. Огурцов*
Технический редактор *О.Н. Власова*
Корректор *Л.А. Пономарева, В.Ф. Малютина*

Сдано в набор 25.01.89. Подп. к печати 19.06.89. Формат 60х90/16. Бум. типографская № 2. Гарнитура Пресс-Роман. Печать офсетная. 20 усл. печ. л., 20,13 усл. кр.-отт., 25,35 уч.-изд. л. Тир. 15000. Зак. 1361 Цена 1р. 30коп.

знак Почета" Издательство стандартов, 123840
москва, ГСП, Новопресненский пер., 3

Набрано в Издательстве стандартов на композере
Калужская типография стандартов. Калуга, ул. Московская, 256.

gost.e-ecolog.ru