1 Структура программы

1.1 Программа

1.1.1 Общие сведения

Основная структурная единица языка КуМир — *алгоритм*. Программа на языке КуМир в простейшем случае состоит из нескольких алгоритмов, следующих один за другим. Перед первым алгоритмом может располагаться *вступление* — любая неветвящаяся последовательность команд. Например, это могут быть строки с комментариями, описаниями общих величин программы, командами присваивания им начальных значений и пр. После последнего алгоритма могут располагаться опсания *исполнителей* (см. 1.4).

Алгоритмы в программе должны располагаться вплотную друг к другу, между ними могут быть только пустые строки и строки с комментариями.

1.1.2 Выполнение программы

Схема программы без вступления и исполнителей:

```
алг первый алгоритм
|
кон

алг второй алгоритм
|
кон
...
алг последний алгоритм
|
кон
```

Выполнение такой программы состоит в выполнении первого алгоритма (он называется *основным* алгоритмом программы). Остальные алгоритмы будут выполняться при вызове из первого алгоритма или из других ранее вызванных алгоритмов.

Схема программы со вступлением и без исполнителей:

```
вступление
алг первый алгоритм
|
кон

алг второй алгоритм
|
кон
...
алг последний алгоритм
|
кон
```

Выполнение такой программы состоит в выполнении вступления, а затем первого алгоритма.

1.1.3 Примеры

```
| Пример 1.
| Это вступление
```

```
цел длина, ширина
длина := 10
ширина := 15
| Это - основной алгоритм.
| У него может не быть имени
алг
нач
· вывод "Площадь равна ", площадь
кон
| Это - вспомогательный алгоритм. При выполнении он вызывается из основного.
| У вспомогательного алгоритма обязательно должно быть имя и могут быть параметры.
алг цел площадь
нач
· знач := длина*ширина
кон

| Пример 2.
| Это вступление
вещ длина, ширина, масса
длина := 10
ширина := 15
алг
нач
· вещ S
· S := площадь
· вещ плотность, масса
· плотность := 6.8 | г/см**2
· найти массу пластинки(плотность, S, масса)
· вывод "Масса пластинки равна ", масса
кон
|
| Это - вспомогательный алгоритм.
| При выполнении он вызывается из основного алгоритма.
| У вспомогательного алгоритма
| обязательно должно быть имя.
алг вещ площадь
нач
· знач := длина*ширина
кон
|
| Это - еще один вспомогательный алгоритм.
| При выполнении он вызывается из
| другого вспомогательного алгоритма.
| У вспомогательного алгоритма
| обязательно должно быть имя.
| У вспомогательного алгоритма могут быть параметры
алг найти массу пластинки(арг вещ p, S, рез вещ m)
нач
· m := p*S
кон
```

1.2 Описание алгоритма

1.2.1 Общий вид описания

Алгоритм на языке КуМир записывается так:
алг тип_алгоритма имя_алгоритма (описание_параметров)
· **дано** условие_применимости_алгоритма
· **надо** цель_выполнения_алгоритма
нач
· последовательность команд
кон

Описание алгоритма состоит из:

- заголовка (часть до служебного слова **нач**)
- тела алгоритма (часть между словами **нач** и **кон**)

1.2.2 Алгоритмы-процедуры и алгоритмы-функции

Алгоритмы делятся на *алгоритмы-процедуры* и *алгоритмы-функции*. Алгоритм-функция после выполнения возвращает значение-результат. Правила описания алгоритмов-процедур и алгоритмов-функций имеют два отличия.

Во-первых, для алгоритмов-функций на месте `тип_алгоритма` должен быть указан один из простых типов алгоритмического языка (**вещ**, **цел** и т.д.), определяющий тип значений, которые принимает данная функция. Для алгоритмов-процедур `тип_алгоритма` должен быть опущен.

Во-вторых, в теле алгоритма-функции необходимо использовать служебную величину **знач**, в которую записывается вычисленное значение функции. В теле алгоритма-процедуры величину **знач** использовать нельзя.

Алгоритмы-функции и алгоритмы-процедуры отличаются также по способу вызова. См. 2.5 и 3.4.

Пример алгоритма-процедуры:
алг гипотенуза (**вещ** a, b, **рез вещ** c)
дано a>=0 **и** b>=0 | длины катетов треугольника
надо | c = длина гипотенузы этого треугольника
нач
· c := sqrt(a**2 + b**2)
кон

Пример алгоритма-функции:
алг вещ площадь (**вещ** a, b, c)
дано a>=0 **и** b>=0 **и** c>=0 | длины сторон треугольника
надо | значение функции равно площади этого треугольника
нач
· **вещ** p | полупериметр
· p := (a+b+c)/2
· **знач** := sqrt(p*(p-a)*(p-b)*(p-c))
кон

Значение, которое должно стать результатом алгоритма-функции, надо присвоить особой величине с именем **знач**. Ее описанием служит заголовок алгоритма, но в остальном величина **знач** используется так же, как и любая другая промежуточная величина. Вызов алгоритма-функции производится путем указания его имени в выражении. Встретив это

имя при вычислении выражения, КуМир выполняет алгоритм-функцию и затем подставляет в выражение вместо имени алгоритма значение величины **знач**.

1.3 Параметры алгоритма

Если алгоритм не имеет параметров (аргументов и результатов), то в строке **алг** записывается только имя алгоритма.

Если у алгоритма есть параметры, то их описание заключается в круглые скобки после имени алгоритма в строке **алг**. Описание содержит информацию о типах параметров и о том, являются они аргументами или результатами:

- **арг** — описания параметров-аргументов
- **рез** — описания параметров-результатов
- **аргрез** (или **арг рез**) — описания параметров, которые одновременно являются и аргументами, и результатами

После каждого из служебных слов **арг**, **рез**, **аргрез** должно располагаться одно или несколько описаний одной или нескольких величин. Имена величин и описания разделяются запятыми. Если в начале описания нет служебных слов **арг**, **рез**, **аргрез**, то предполагается, что первыми идут описания аргументов (**арг**).

Пример.

```
алг
нач
· вещ число
· цел целое, сотые
· лит запись
· число := 3.14
· тест(целое, сотые, запись, число)
кон

алг тест (рез цел m, n, лит т, арг вещ y)
нач
· вещ r
· m := int(y)
· r := (y - m)*100
· n := int(r)
· т := вещ_в_лит(y)
кон
```

В заголовке алгоритма **алг** тест (**рез цел** m, n, **лит** т, **арг вещ** y) служебное слово **рез** относится к описаниям **цел** m, n и **лит** т, а параметр **вещ** y будет аргументом.

ВНИМАНИЕ: Запрещается писать в теле алгоритма команды, изменяющие значения параметров-аргументов (описанных как **арг**). Результаты алгоритма (**рез**, но не **аргрез**) в начале выполнения алгоритма принимают неопределенные значения.

1.4 Описание исполнителя

После последнего алгоритма программы может идти одна или несколько конструкций *исполнитель*. Таким образом, в самом общем виде программа имеет такой вид:

```
вступление программы
первый алгоритм
второй алгоритм
```

...
последний алгоритм
первый исполнитель
второй исполнитель
...
последний исполнитель

Конструкция *исполнитель* на языке КуМир записывается так:

```
исп имя_исполнителя
вступление_исполнителя
алг первый_алгоритм_исполнителя
|
кон
...
алг последний_алгоритм_исполнителя
|
кон
кон_исп
```

В алгоритмах программы и исполнителей могут использоваться алгоритмы программы и любых исполнителей. В алгоритмах исполнителя могут использоваться общие величины этого исполнителя, но не общие величины программы и других исполнителей. Во вступлении данного исполнителя могут использоваться алгоритмы программы и алгоритмы исполнителей, записанных по тексту выше этого исполнителя.

При выполнении программы вначале выполняется ее вступление, затем, по порядку, вступления всех исполнителей. После этого начинает выполняться *основной алгоритм* (т. е. первый по порядку). Выполнение программы заканчивается, когда заканчивается выполнение основного алгоритма.

1.5 Команды и строки

В простейшем случае каждая простая команда и каждое ключевое слово в составных командах пишется на отдельной строке. Однако, чтобы сделать программу более компактной, можно «склеивать» несколько строк в одну. Это можно сделать в следующих случаях.

1.5.1 Использование точки с запятой

Точка с запятой приравнивается к переносу строки.

Пример:

Программа 1 и Программа 2 имеют одинаковый смысл.

```
| Программа 1 — сжатое написание
алг
нач
· цел а; вещ в
· а := 5; в := 0.1
кон

| Программа 2 — полное написание
алг
нач
· цел а
· вещ в
```

· а := 5
· в := 0.1
кон

1.5.2 Неявные переносы строк

Для многих *ключевых* слов можно догадаться, что перед ними или после них должен быть перевод строки.

«Неявные» переносы строк вставляются в следующих случаях:

- перед словами **все**, **кц**, **кц_при**
- после слов **нач**, **выбор**, **нц** (только в случае цикла *нц-кц*), **раз**
- перед и после слов **то**, **иначе**, **при**
- перед словом **при** и после двоеточия в *при*-строке

Пример:

алг
нач цел знак, **вещ** модуль
· **вещ** щ
· **ввод** щ
· модуль :=0; знак := 0
· **если** щ > 0 **то**
· · модуль :=щ; знак := 1
· **все**
· **если** щ < 0 **то** модуль :=щ; знак := 1 **все**
кон

1.6 Комментарии

алг
· # Это алгоритм вычисления суммы двух чисел
нач
· **цел** а, б | объявляем величины
· **ввод** а, б | вводим значения с клавиатуры
· **вывод** а+б | посчитаем сумму чисел
кон

В этом алгоритме после знака | в некоторых строках записаны комментарии. Такие комментарии разрешается помещать в конце любой строки, отделяя их знаком |. Если комментарий занимает несколько строк, то знак | перед комментарием надо ставить в каждой строке. Комментарии могут записываться в любой удобной для человека форме. При выполнении алгоритма компьютер полностью пропускает комментарии — алгоритм выполнятеся так же, как если бы комментариев вообще не было.

Таким образом, комментарии предназначены исключительно для человека — они облегчают понимание алгоритма.

Кроме того, существует особый вид комментария – он может располагаться только между строками **алг** и **нач** алгоритма и начинается с символа #. С помощью этого комментария описывается весь алгоритм в целом. Данная информация отображается в пункте главного меню Инструменты – Алгоритмы пользователя.

2 Имена, величины и выражения

2.1 Имена

2.1.1 Общие сведения

Имя бывает у величин, таблиц, алгоритмов и исполнителей. Имя – это последовательность слов, разделенных пробелами. Первое слово имени не должно начинаться с цифры. Ни одно из слов не должно быть ключевым словом.

Примеры имен: `m`, `погода на завтра`, `Ноябрь 7`, `Седьмое ноября`, `дом_57б`.

Примеры неправильных имен:

- `7е ноября` (первое слово начинается с цифры)
- `альфа-бета` ("-" — недопустимый символ)
- `альфа или омега` (**или** — ключевое слово)

Примечание. Ключевое слово **не** можно вставлять внутрь многословных логических имен (см. 2.1.4).

2.1.2 Слова

Слово — это последовательность разрешенных (словарных) символов. *Словарными* символами являются:

- буквы (кириллические и латинские, прописные и строчные)
- цифры
- два специальных знака: `@ _`

Примеры слов: `бета123`, `3кг`, `мама`, `Linux`, `КоСтЯ`, `kumir@infomir_ru`.

Примеры не слов: `альфа-123`, `ма%ма`, `C++`.

2.1.3 Ключевые слова

Ключевые слова языка КУМИР — это: **алг нач кон исп кон_исп дано надо арг рез аргрез знач цел вещ лог сим лит таб целтаб вещтаб логтаб симтаб литтаб и или не да нет утв выход ввод вывод нс если то иначе все выбор при нц кц кц_при раз пока для от до шаг**.

2.1.4 Многословные не-имена

В отрицаниях логических величин, таблиц и алгоритмов функций ключевое слово **не** можно вставлять между словами многословного имени.

Пример:

лог л, завтра будет четверг
л := **не** завтра будет четверг | Правильно
л := завтра **не** будет четверг | Правильно
л := завтра будет **не** четверг | Правильно
л := завтра будет четверг **не** | Неправильно
л := **не** завтра **не** будет четверг | Неправильно

Первые три присваивания присваивают логической величине л значение, противоположное значению логической величины завтра будет четверг. Четвертая строка синтаксически неверна — **не** нельзя ставить после имени. Последняя строка также неверна: нельзя использовать более одного **не**.

2.2 Типы величин

Величины, с которыми работает КуМир-программа, подразделяются на несколько *типов*. Величина каждого из типов может принимать свой набор значений. В языке КуМир предусмотрены следующие типы величин:

- **цел** — принимает целые значения от -2147483647 до 2147483647
- **вещ** — принимает вещественные значения между -2^{1023} и 2^{1023}
- **лог** — принимает значения **да** или **нет** (внутреннее представление – **да**=1, **нет**=0)
- **сим** — значением может быть любой литеральный символ (практически любой символ, см. 2.1.2)
- **лит** — значением может быть строка литеральных символов

Типы **цел** и **вещ** называются *числовыми*; типы **сим** и **лит** — *текстовыми*.

Язык КуМир содержит встроенные функции преобразования числовых типов в текстовые и наоборот (см. 5.1). При необходимости значения целого типа автоматически переводятся в вещественные, а символьные — в текстовые. Для преобразования вещественных значений в целые используется встроенная функция int (см. 5.2.20).

2.3 Константы

2.3.1 Виды констант

Константы бывают целые, вещественные, логические, символьные и литеральные.

2.3.2 Целые константы

Целые константы бывают положительные и отрицательные. Целая константа по абсолютной величине должна быть строго меньше 2^{31}. Целые константы можно записывать в десятичной и 16-ричной форме. Шестнадцатеричные константы начинаются с символа $.

Примеры: 123, -100000, $100.

2.3.3 Вещественные константы

Вещественные константы бывают положительные и отрицательные. Вещественная константа по абсолютной величине должна быть меньше 2^{1023}. Вещественные константы можно записывать в десятичной и экспоненциальной форме. В качестве разделителя в экспоненциальной записи можно использовать любой вариант буквы *e*: строчный или прописной, латинский или кириллический.

Ограничения для вещественных констант определяются стандартом *IEEE 754-2008*.

Примеры: 1.23, -0.56, 1e+4, 5E-7.

2.3.4 Логические константы

Логическая константа — это одно из ключевых слов **да**, **нет**.

2.3.5 Символьные и литеральные константы

В символьной константе допустим любой символ, который можно набрать на стандартной клавиатуре. Такие символы называются *допустимыми*.

Символьная константа имеет вид 'c' или "c" (здесь c — допустимый символ).

Примеры: 'a', "%", ' " ', " ' ", 'Это я', "It's me".

Литеральная (текстовая) константа имеет вид 'T' или "T". Здесь T — строка, состоящая из допустимых символов. При этом, если константа T ограничена простыми кавычками, то T не содержит простую кавычку, а если T ограничена двойными кавычками, то она не содержит двойную кавычку.

2.4 Величины

2.4.1 Общие сведения

Каждая величина имеет *имя*, *тип*, *вид* и *значение*.

Имя величины служит для обозначения величины в алгоритме (см. 2.1).

Тип величины показывает, какие значения может принимать величина, и какие операции можно с ней выполнять (см. 2.2).

Вид величины показывает ее информационную роль в алгоритме. Например, аргументы содержат исходную информацию, необходимую для работы алгоритма, а *промежуточные величины* предназначены для хранения текущей информации, которую обрабатывает алгоритм.

Во время выполнения алгоритма в каждый конкретный момент величина имеет какое-то *значение* либо *не определена*.

Имя, тип и вид величины можно однозначно определить по тексту алгоритма.

Это *статические* характеристики величины. *Значение* определяется только во время выполнения. Это *динамическая* характеристика.

2.4.2 Простые величины и таблицы. Описания величин

В языке КУМИР используются *простые* и *табличные* величины (*таблицы*).

Характеристики простых величин описаны в 2.4.1. Для таблиц кроме того определена *размерность* (бывают таблицы размерностей 1, 2 и 3). Для каждого измерения определены границы изменения *индекса* таблицы по этому измерению — два целых числа.

2.4.3 Описания величин

Каждая величина должна иметь описание. Это может быть сделано:

- с помощью оператора описания
- при задании формальных параметров алгоритма (см. 1.3)

В описании задаются перечисленные выше статические характеристики переменной.

Кроме того, в алгоритмах-функциях используется простая переменная **знач**, ее тип определяется типом функции, (см. 1.2). Явного описания переменная **знач** не имеет. Ее область действия — тело соответствующего алгоритма-функции.

Команда описания простой величины состоит из ключевого слова нужного типа (**цел**, **вещ**, **сим**, **лит**, **лог**), за которым следует список имен величин.

Пример.

цел j, k, n

вещ длина, ширина
лит мой текст

Для описания таблиц после описания типа нужно указать ключевое слово **таб** (слитно или раздельно с ключевым словом типа). Размерность таблицы и границы изменения индексов указываются после имени каждой величины.

Примеры.
цел таб k[-5:5]
вещтаб tab[1:4, 1:12]

Здесь k — линейная таблица, состоящая из 11 элементов целого типа. Индексы элементов принимают значения от -5 до 5. Таблица tab — прямоугольная. В ней 48 элементов — 4 строки и 12 столбцов.

2.4.4 Область действия описаний

В зависимости от способа описания и места описания в программе, где описана величина, определена ее *область действия описания* — та часть текста программы, где допустимо использование этой величины.

Если величина описана во вступлении к программе, ее можно использовать в любом алгоритме этой программы (но не в исполнителях!).

Если величина описана во вступлении к исполнителю, то ее можно использовать в любом алгоритме этого исполнителя.

Если величина описана в заголовке алгоритма, то ее можно использовать в теле этого алгоритма, а также в заголовке — после этого описания.

Пример:
алг цел сумма элементов таблицы (**цел** длина, **целтаб** таблица[1:длина])

Если переменная описана в теле алгоритма, то ее можно использовать только в теле этого алгоритма *после* места описания. Такие величины называются *промежуточными.*

Пример:
алг
нач
· п := 1 | Так нельзя!
· **цел** п
· п := 1 | Так можно
кон

2.5 Выражения

2.5.1 Общие сведения

Выражение в языке КуМир описывает новое значение, полученное из уже известных значений с помощью предусмотренных в языке КуМир *операций.*

Примеры:

- (a+b)*(a-b)
- да **или** нет
- (sin(альфа))**2+(cos(альфа))**2

В КуМир-программе выражения могут появляться в:

- правой части оператора присваивания

- в индексе таблицы
- в аргументе (типа **арг**) вызова функции
- в качестве подвыражения другого выражения
- в команде **вывод**

2.5.2 Операции в языке КуМир

Операции в языке КуМир — это:

- *базовые операции* (арифметические, логические, текстовые)
- *вырезка из строки*
- *операции, задаваемые алгоритмами-функциями*

Для каждой операции известны:

- количество значений-аргументов
- их типы
- тип результата

2.5.3 Базовые операции

В зависимости от типов аргументов и результата, базовые операции делятся на следующие классы:

- арифметические операции (аргументы и результат — числового типа)
- сравнение арифметическое (аргументы — числового типа, результат — **лог**)
- сравнение текстовое (аргументы — текстового типа, результат — **лог**)
- логические операции (аргументы — **лог**, результат — **лог**)
- текстовые операции (аргументы и результат — текстового типа)

Каждой базовой операции соответствует свой символ. В некоторых случаях приходится применять составной символ, состоящий из двух обычных символов:

- ** — возведение в степень;
- <=, >= — нестрогие неравенства.

Полный список базовых операций и их описания приведены в 9.9.

2.5.4 Тип выражения. Согласованность типов

Типом выражения называется тип результата операции, которая выполняется последней при вычислении этого выражения.

Типы всех подвыражений должны быть согласованы с типами аргументов выполняемых операций.

Пример.

Рассмотрим выражение

гамма(x) - дельта(2y+1, z),

где гамма и дельта — описанные в программе алгоритмы-функции. Это должны быть функции числового типа. Если обе они имеют тип **цел**, то и все выражение имеет тип **цел**. В противном случае выражение имеет тип **вещ**.

2.5.5 Вырезка из строки

Операция *вырезки из строки* имеет 3 аргумента: **лит** строка, **цел** старт, **цел** финиш и результат: **лит** вырезка. В отличие от базовых операций, аргументы вырезки из строки имеют разные типы. Поэтому способ записи вырезки из строки отличается от способа, принятого для базовых операций.

Пример:

```
лит строка, вырезка
строка = "строка"
вырезка := строка[3:5]
утв вырезка = "рок"
```

2.5.6 Функции

В выражениях языка КУМИР можно использовать:

- встроенные алгоритмы-функции КУМИРа, например: sin(x), длин("ХВОСТ")
- алгоритмы-функции встроенных исполнителей, например: температура
- алгоритмы-функции программы пользователя (в том числе — алгоритмы-функции исполнителей пользователя)

. У каждой функции есть имя, для нее фиксировано количество параметров, параметры перенумерованы. Для каждого параметра функции и ее результата фиксированы их типы; тип результата называется типом функции.

Вызов функции с именем имя_функции и аргументами, заданными выражениями $X_1, \ldots, X_n$ записывается так: имя_функции($X_1, \ldots, X_n$).

2.5.7 Примеры записи выражений

$-\frac{1}{x^2}$	-1/x**2
$\frac{a}{bc}$	a/(b*c)
$\frac{a}{b}c$	a/b*c или (a/b)*c
$2^{2^{2^n}}$	2**(2**(2**n))
x^{y^z}	x**(y**z)
$\frac{-b+\sqrt{b^2-4ac}}{2a}$	(-b+sqrt(b**2-4*a*c))/(2*a)
$\sqrt{p(p-a)(p-b)(p-c)}$	sqrt(p*(p-a)*(p-b)*(p-c))
$\frac{a+b+c}{2}$	(a+b+c)/2
$\sqrt{a^2+b^2-2ab\cos\gamma}$	sqrt(a**2+b**2-2*a*b*cos(gamma))
$\frac{ad+bc}{bd}$	(a*d+b*c)/(b*d)
$\sin\alpha\cos\beta+\cos\alpha\sin\beta$	sin(alfa)*cos(beta)+cos(alfa)*sin(beta)

3 Простые команды

В этом разделе описаны 5 видов простых команд (из 6 допустимых в языке КуМир):

- команды присваивания
- команды контроля
- команды ввода-вывода
- команда **выход**
- команда вызова алгоритма-процедуры

Еще один вид простых команд — команды описания величин — представлен в 2.4.3.

3.1 Присваивание

Команда присваивания предназначена для изменения значения простых переменных и элементов таблиц и имеет общий вид <ВЕЛИЧИНА> := <ВЫРАЖЕНИЕ>, где

- *ВЕЛИЧИНА* — это имя простой величины или описание элемента таблицы
- *ВЫРАЖЕНИЕ* — это выражение, составленное из величин, констант, вызовов алгоритмов-функций и знаков операций

Тип выражения должен быть согласован с типом величины.

Примеры:

```
n := 0
m := n
m := m+1
m := 2*длин(t)+div(n,2)
c := (x+y)/2
площадь:=a*b*sin(C)/2
d:=b**2-4*a*c
x[1]:=(-b+sqrt(d))/(2*a)
a[i]:=2*a[i-2]+a[i-1]
b[i,j]:=-b[j,i]
```

Все переменные должны быть описаны, а их типы — согласованы с типами операций и их аргументов.

3.2 Контроль выполнения

В языке КуМир существует три команды контроля выполнения: **утв**, **дано**, **надо**.

Формат вызова:

утв <ЛОГ ВЫРАЖЕНИЕ>
дано <ЛОГ ВЫРАЖЕНИЕ>
надо <ЛОГ ВЫРАЖЕНИЕ>

Все три команды выполняются так. Проверяется условие. Если условие не соблюдается (равно **нет**), то КуМир прекращает выполнение алгоритма и сообщает, что возник отказ. Если же условие соблюдается, то выполнение алгоритма нормально продолжается так, как если бы команды контроля не было вовсе.

Команда **дано** проверяет условие в начале выполнения алгоритма, команда **надо** — в конце выполнения алгоритма, а командой **утв** можно проверить условие в процессе выполнения алгоритма.

Пример 1:

```
алг абс (рез вещ x)
дано x<=0
надо x>=0
нач
· x := -x
кон
```

Пример 2:

```
алг вещ кв (вещ x)
нач
· вещ к
· к := x*x
· утв к>=0
· знач := к
кон
```

3.3 Ввод-вывод

3.3.1 Вывод

Формат вызова:

вывод выражение-1, . . . , выражение-N

Каждое выражение может быть либо арифметическим, логическим или текстовым выражением, либо командой перехода на новую строку (ключевое слово **нс**). Значения выражений выводятся последовательно в строку области ввода-вывода и разделяются пробелом. Когда строка полностью заполнена, автоматически происходит переход к началу новой строки.

Когда окно ввод-вывода полностью заполнено, последующие команды вывода будут сдвигать содержимое окна вверх, вытесняя верхние строки окна.

Пример:

```
алг
нач
· нц 5 раз
· · вывод "Hello!", нс
· кц
кон
```

3.3.2 Ввод

Формат вызова:

ввод имя-1, . . . , имя-N

При выполнении этой команды КуМир выводит курсор в окно ввода-вывода и ждет, пока пользователь введет соответствующие значения. По окончании введенные значения присваиваются указанным величинам. В качестве имени величины можно указать имя простой величины или имя элемента таблицы с указанием значений индексов. Признаком конца ввода служит нажатие на клавишу Enter. При вводе нескольких чисел они отделяются друг от друга запятой или пробелом.

Пример:

```
алг
нач
· целтаб т[1:10]
· цел ц, а
· ввод ц
· нц для а от 1 до ц
· · ввод т[а]
· кц
· нц для а от 1 до ц
· · вывод т[а], нс
· кц
кон
```

3.4 Вызов алгоритма

Вызов алгоритма-процедуры является отдельной командой алгоритмического языка и имеет вид:

- имя_алгоритма-процедуры или
- имя_алгоритма-процедуры (список_параметров_вызова)

Пример 1:

```
алг
нач
· подпр
кон

алг подпр
нач
· вывод "Мы в подпрограмме", нс
кон
```

Пример 2:

```
нач
· сумма(2.4, 7.6)
кон

алг сумма(вещ а, вещ б)
нач
· вывод "Сумма = ", а+б, нс
кон
```

3.5 выход

Команда **выход** используется для выхода из цикла или для окончания работы текущего алгоритма. Если команда **выход** выполняется внутри цикла, то выполнение продолжается с первой команды после тела этого цикла. Если команда **выход** используется во вложенных циклах, то завершается самый внутренний цикл. Если команда **выход** выполняется вне циклов, то она приводит к завершению выполнения текущего алгоритма.

Пример:

```
алг
```

```
нач
· нц
· · нц
· · · вывод "–2–", нс
· · · выход
· · кц
· · вывод "–1–", нс
· · выход
· кц
· вывод "–0–", нс
· выход
· вывод "–F–", нс
кон
```

При выполнении этой программы будет напечатано:

```
–2–
–1–
–0–
–F–
```

4 Составные команды

4.1 Команды ветвления

4.1.1 если-то-иначе-все

Общий вид команды:

```
если условие
· то серия1
· иначе серия2
все
```

Серия2 вместе со служебным словом **иначе** может отсутствовать. В этом случае команда имеет вид:

```
если условие
то серия1
все
```

При выполнении команды **если** КуМир сначала проверяет условие, записанное между **если** и **то**. При соблюдении этого условия выполняется *серия1*, в противном случае — *серия2* (если она есть), после чего КуМир переходит к выполнению команд, записанных после слова **все**.

Если условие не соблюдается, а *серия2* вместе с **иначе** отсутствует, то КуМир сразу переходит к выполнению команд, записанных после слова **все**.

Пример 1:

```
если a<b
· то b:=b-a; p:=p+q
· иначе a:=a-b; q:=q+p
все
```

Пример 2:

```
если x>m
· то
· · m:=x
```

· · n:=n+1
все

4.1.2 выбор-при-иначе-все

Общий вид команды:
выбор
· **при** условие 1 : серия 1
· **при** условие 2 : серия 2
· ...
· **при** условие n : серия n
· **иначе** серия n+1
все

Ключевое слово **иначе** вместе с соответствующей серией команд может отсутствовать:
выбор
· **при** условие_1 : серия_1
· **при** условие_2 : серия_2
· ...
· **при** условие_n : серия_n
все

КуМир сначала проверяет условие 1. Если оно соблюдается, то КуМир выполняет команды из серии 1, после чего переходит к выполнению команд, записанных после слова **все**. В противном случае КуМир делает то же самое с условием 2 и командами из серии 2 и т.д.

Команды, записанные после слова **иначе**, выполняются в том случае, когда не соблюдено ни одно из условий.

В команде **выбор** всегда выполняется не более одной серии команд, даже если несколько условий окажутся истинными. Выполнение команды **выбор** заканчивается после того, как найдено первое (по порядку следования) условие со значением **да** (и выполнена соответствующая серия команд).

Пример 1:
выбор
· **при** a>1: i:=i+1
· **при** a<0: j:=j-1
· **иначе** t:=i; i:=j; j:=t
все

Пример 2:
выбор
· **при** a[i]>1000 : b[i]:=3; c[i]:=3.141
· **при** a[i]>100 :
· · b[i]:=2; c[i]:=3.14
· **при** a[i]>10 :
· · b[i]:=1
· · c[i]:=3.14
все

В примере 2 при a[i]=1812 будут выполнены присваивания: b[i]:=3; c[i]:=3.141.

4.2 Команды цикла

4.2.1 Цикл «для»

Общий вид цикла *для*:
нц для i **от** i1 **до** i2
· тело_цикла
кц

Здесь i — величина типа **цел** (она называется параметром цикла), а $i1$ и $i2$ — целые выражения, т. е. выражения типа **цел**. При выполнении цикла **для** тело цикла выполняется последовательно для $i = i1$, $i = i1 + 1$, ..., $i = i2$. Если $i1 = i2$, то тело цикла выполнится один раз для $i = i1$. Если же $i1 > i2$, то тело цикла не выполнится ни разу.

Общий вид цикла *для с шагом*:
нц для i **от** i1 **до** i2 **шаг** i3
· тело_цикла
кц

Если шаг $i3$ (который также должен быть целым выражением) равен положительному числу d, то тело цикла будет выполняться последовательно для $i = i1$, $i = i1 + d$, $i = i1 + 2d$, ... до тех пор, пока значение i удовлетворяет условию $i \le i2$.

Если же шаг $i3$ отрицателен и равен $-d$, то тело цикла будет выполняется последовательно для $i = i1$, $i = i1 - d$, $i = i1 - 2d$, ... до тех пор, пока значение i удовлетворяет условию $i \ge i1$.

Пример 1:
нц для j **от** 1 **до** длин(t)
· t1[j]:=t[длин(t)+1-j]
кц

Пример 2:
нц для i **от** 1 **до** 100 **шаг** 2
· a[i+1]:=a[i]
кц

Пример 3:
нц для i **от** 100 **до** 1 **шаг** -2
· a[i]:=a[i-1]
кц

В теле любого из циклов может быть использована команда **выход** (см. 3.5). При выполнении этой команды содержащий ее цикл будет завершен.

4.2.2 Цикл «пока»

Общий вид цикла *пока*:
нц пока условие
· тело_цикла
кц

При выполнении цикла **пока** КУМИР циклически повторяет следующие действия:

- Проверяет записанное после служебного слова **пока** условие.
- Если условие не соблюдается, то выполнение цикла завершается и КуМир начинает выполнять команды, записанные после **кц**.

  Если же условие соблюдается, то КуМир выполняет тело цикла, снова проверяет условие и т.д.

Пример:

```
нц пока a<10
· a := a + 1
кц
```

В теле цикла может быть использована команда **выход** (см. 3.5). При выполнении этой команды содержащий ее цикл будет завершен.

4.2.3 Цикл «до тех пор»

Общий вид цикла *до тех пор*:

```
нц
· тело_цикла
кц_при условие
```

При выполнении цикла *до тех пор* КуМир циклически повторяет следующие действия:

- Выполняет тело цикла.
- Проверяет записанное после служебного слова **кц_при** условие.
- Если условие соблюдается, то выполнение цикла завершается и КуМир начинает выполнять команды, записанные после **кц_при**. Если же условие не соблюдается, то КуМир выполняет тело цикла, снова проверяет условие и т.д.

Пример:

```
нц
· x := 2*x
кц_при x>100
```

Условие окончания цикла может быть добавлено в любую команду повторения, например, в *цикл N раз*.

Пример:

```
нц 5 раз
· ввод x, y, z
· вывод нс, "Координаты:", x, y, z
кц_при x+y+z>100
```

В теле любого из циклов может быть использована команда **выход** (см. 3.5). При выполнении этой команды содержащий ее цикл будет завершен.

4.2.4 Цикл «N раз»

Общий вид цикла *N раз*:

```
нц N раз
· тело_цикла
кц
```

Здесь N — целое выражение, задающее число повторений. При выполнении алгоритма последовательность команд циклически повторяется указанное число раз.

Пример:

```
нц 4 раз
· ввод x, y, z
· вывод нс, "Координаты:", x, y, z
кц
```

В теле цикла может быть использована команда **выход** (см. 3.5). При выполнении этой команды содержащий ее цикл будет завершен.

4.2.5 Цикл «нц-кц»

Общий вид цикла:
нц
· тело_цикла
кц
Пример:
нц
· а := а + 1
· **если** а>100 **то выход все**
кц

КуМир не проверяет, встречается ли в теле цикла команда **выход**. Если такой команды нет, то цикл *нц-кц* будет выполняться до бесконечности.

5 Встроенные алгоритмы

5.1 Текстовое представление чисел

5.1.1 цел_в_лит

Синтаксис: **алг лит** цел_в_лит(**цел** x)
Возвращает строковое представление x.
Пример:
алг
нач
цел а
· **лит** б
· а := 5
· б := **цел_в_лит**(а)
· **вывод** б
кон

5.1.2 вещ_в_лит

Синтаксис: **алг лит** вещ_в_лит(**вещ** x)
Возвращает строковое представление x.
Пример:
алг
нач
· **вещ** а
· **лит** б
· а := 5.9999
· б := **вещ_в_лит**(а)
· **вывод** б
кон

5.1.3 лит_в_вещ

Синтаксис: **алг вещ** лит_в_вещ(**лит** *СТРОКА*, **рез лог** *УСПЕХ*)

Переводит строку *СТРОКА* в вещественное представление. Если *СТРОКА* содержит только вещественное число, то в *УСПЕХ* записывается **Да** и алгоритм возвращает вещественное значение, иначе в *УСПЕХ* записывается **Нет** и алгоритм возвращает значение **0**.

Пример:

```
алг
нач
· лит а
· вещ б
· лог усп
· а := "5.9999"
· б := лит_в_вещ(а, усп)
· вывод б, " ", усп
кон
```

5.1.4 лит_в_цел

Синтаксис: **алг цел** лит_в_цел(**лит** *СТРОКА*, **рез лог** *УСПЕХ*)

Переводит строку *СТРОКА* в целочисленное представление. Если *СТРОКА* содержит только целое число, то в *УСПЕХ* записывается **Да** и алгоритм возвращает целосисленное значение, иначе в *УСПЕХ* записывается **Нет** и алгоритм возвращает значение **0**.

Пример:

```
алг
нач
· лит а
· цел б
· лог усп
· а := "5"
· б := лит_в_цел(а, усп)
· вывод б, " ", усп
кон
```

5.2 Математика

5.2.1 sqrt

Синтаксис: **алг вещ** sqrt(**вещ** x)

$\sqrt{x}$ — квадратный корень из x ($x \geq 0$).

Пример:

```
вещ x
алг
нач
· ввод x
· x := sqrt (x)
· вывод "корень квадратный из числа x равен ", x
кон
```

5.2.2 abs

Синтаксис: **алг вещ** abs(**вещ** x)

Абсолютная величина вещественного числа x ($|x|$).

Пример:

```
вещ а, б
алг
нач
```

```
· ввод а,б
· а := а+б
· а := abs(а)
· вывод"Модуль суммы чисел равен ", а
кон
```

5.2.3 iabs

Синтаксис: **алг цел** iabs(**цел** x)
Абсолютная величина целого числа x ($|x|$)
Пример:

```
цел а, б
алг
нач
· ввод а,б
· а := iabs(а)
· б := iabs(б)
· вывод а+б
кон
```

5.2.4 sign

Синтаксис: **алг цел** sign(**вещ** x)
Знак числа x (-1, 0 или 1):

- -1, если $x < 0$
- 0, если $x = 0$
- 1, если $x > 0$

Пример:

```
цел а, б
алг
нач
· ввод а
· б := sign(а)
· если б=-1
· · то вывод а, "<=0"
· · иначе
· · · если б=0
· · · · то вывод а, "=0"
· · · · иначе вывод а, ">=0"
· · · все
· все
кон
```

5.2.5 sin

Синтаксис: **алг вещ** sin(**вещ** x)
Синус x
Пример:

```
вещ x
```

```
алг
нач
· ввод x
· x := sin (x)
· вывод "синус угла x равен ", x
кон

вещ x, y
алг
нач
· вывод "угол x="
· ввод x
· y := 2*sin(x)*cos(x)
· вывод "sin2x = ", y
кон
```

5.2.6 cos

Синтаксис: **алг вещ** cos(**вещ** x)
Косинус x
Пример:

```
вещ x
алг
нач
· ввод x
· x := cos (x)
· вывод "косинус угла x равен ", x
кон

вещ x, y
алг
нач
· вывод "угол x="
· ввод x
· y := 2*sin(x)*cos(x)
· вывод "sin2x = ", y
кон
```

5.2.7 tg

Синтаксис: **алг вещ** tg(**вещ** x)
Тангенс x
Пример:

```
вещ x
алг
нач
· ввод x
· x := tg (x)
· вывод "тангенс угла x равен ", x
кон
```

5.2.8 ctg

Синтаксис: **алг вещ** ctg(**вещ** x)
Котангенс x
Пример:

```
вещ x
алг
нач
· ввод x
· x := ctg (x)
· вывод "котангенс угла x равен ", x
кон
```

5.2.9 arcsin

Синтаксис: **алг вещ** arcsin(**вещ** x)
Арксинус x
Пример:

```
вещ x
алг
нач
· ввод x
· x := arcsin (x)
· вывод "арксинус числа x равен ", x
кон
```

5.2.10 arccos

Синтаксис: **алг вещ** arccos(**вещ** x)
Арккосинус x
Пример:

```
вещ x
алг
нач
· ввод x
· x := arccos (x)
· вывод "арккосинус числа x равен ", x
кон
```

5.2.11 arctg

Синтаксис: **алг вещ** arctg(**вещ** x)
Арктангенс x
Пример:

```
вещ x
алг
нач
· ввод x
· x := arctg (x)
· вывод "арктангенс числа x равен ", x
кон
```

5.2.12 arcctg

Синтаксис: **алг вещ** arcctg(**вещ** x)
Арккотангенс x
Пример:

```
вещ x
алг
нач
· ввод x
· x := arcctg (x)
· вывод "арккотангенс числа x равен ", x
кон
```

5.2.13 ln

Синтаксис: **алг вещ** ln(**вещ** x)
Натуральный логарифм x
Пример:

```
вещ a,б,c
алг
нач
· ввод a, б
· c := a+б
· c := ln(c)
· вывод "Натуральный логарифм от суммы чисел ",a," и ",б," равен ",c
кон
```

5.2.14 lg

Синтаксис: **алг вещ** lg(**вещ** x)
Десятичный логарифм x
Пример:

```
вещ a,б,c
алг
нач
· ввод a, б
· c := a+б
· c := lg(c)
· вывод "десятичный логарифм от суммы чисел ",a," и ",б," равен ",c
кон
```

5.2.15 exp

Синтаксис: **алг вещ** exp(**вещ** x)
e в степени числа x ($e \approx 2.718281828459045\ldots$)
Пример:

```
вещ x
цел a
алг
нач
· ввод a
· x := exp(a)
```

```
· вывод "число е в степени ", а, " равно ", х
кон
```

5.2.16 min

Синтаксис: **алг вещ** min(**вещ** x, **вещ** y)
Минимум из чисел x и y
Пример:

```
вещ а,б,с1, с2
алг
нач
· ввод а, б
· с1 := max(а,б)
· с2 := min(а,б)
· вывод с1, нс
· вывод с2, нс
кон
```

5.2.17 max

Синтаксис: **алг вещ** max(**вещ** x, **вещ** y)
Максимум из чисел x и y
Пример:

```
вещ а,б,с1, с2
алг
нач
· ввод а, б
· с1 := max(а,б)
· с2 := min(а,б)
· вывод с1, нс
· вывод с2, нс
кон
```

5.2.18 mod

Синтаксис: **алг цел** mod(**цел** x, **цел** y)
Остаток от деления x на y (x,y - целые, $y > 0$))
Пример:

```
цел а, б, х, у
алг
нач
· ввод а, б
· х := div(а,б)
· у := mod(а,б)
· вывод "а/б=", х, " с остатком ", у
кон
```

5.2.19 div

Синтаксис: **алг цел** div(**цел** x, **цел** y)
Частное от деления x на y (x, y - целые, $y > 0$)

Пример:

```
цел а, б, x, y
алг
нач
· ввод а, б
· x := div(а,б)
· y := mod(а,б)
· вывод "а/б=", x, " с остатком ", y
кон
```

5.2.20 int

Синтаксис: **алг цел** int(**вещ** x)

Целая часть x: максимальное целое число, не превосходящее x

Пример:

```
вещ а,б
алг
нач
· ввод а
· б := int(а)
· вывод "Целая часть ", а, " равна ", б
кон
```

5.2.21 rnd

Синтаксис: **алг вещ** rnd(**вещ** x)

Случайное число от 0 до x: при последовательных вызовах этой функции получается последовательность случайных чисел, равномерно распределенных на $[0,x]$.

Пример:

```
алг Построение последовательности случайных вещественных чисел
нач
· вещ таб а [1:10]
· цел л
· вещ б
· ввод б
· нц для л от 1 до 10
· · а[л] := rnd(б)
· кц
· нц для л от 1 до 10
· · вывод а[л], " "
· кц
кон
```

5.3 Обработка строк

5.3.1 длин

Синтаксис: **алг цел** длин(**лит** S)

Возвращает количество символов в строке S.

Пример:

```
алг
```

```
нач
· лит а
· цел ц
· вывод "введите строку"
· ввод а
· ц := длин(а)
· вывод ц
кон
```

5.3.2 код

Синтаксис: **алг цел** код(**сим** *с*)

Возвращает номер символа *с* в таблице КОИ-8r. (стандарт RFC 1489).

Пример:

```
алг
нач
· сим а
· цел ц
· вывод "введите символ"
· ввод а
· ц := код(а)
· вывод ц
кон
```

5.3.3 символ

Синтаксис: **алг сим** символ(**цел** *N*)

Возвращает символ, соответствующий номеру *N* в таблице КОИ-8r (стандарт RFC 1489).

Пример:

```
алг
нач
· цел а
· сим б
· ввод а
· б := символ(а)
· вывод б
кон
```

5.3.4 юникод

Синтаксис: **алг цел** юникод(**сим** *с*)

Возвращает номер символа *с* в таблице Юникода.

5.3.5 символ2

Синтаксис: **алг сим** символ2(**цел** *N*)

Возвращает символ, соответствующий номеру *N* в таблице Юникода.

Пример:

```
алг
нач
```

· **цел** а
· **сим** б
· **ввод** а
· б := **символ2**(а)
· **вывод** б
кон

5.4 Система

5.4.1 пауза

Синтаксис: **алг** пауза

Приостанавливает выполнение программы. Переводит Кумир в режим паузы.

Пример:

алг
нач
· **вещ** а, б, с
· а := 1
· б := 2
· **пауза**
· с := а+б
· **вывод** с
кон

5.4.2 стоп

Синтаксис: **алг** стоп

Останавливает выполнение программы.

Пример:

алг
нач
· **вещ** а, б, с
· а := 1
· б := 2
· **вывод** "Остановка перед вычислением значения С"
· **стоп**
· с := а+б
· **вывод** с
кон

5.4.3 время

Синтаксис: **алг цел** время

Возвращает текущее время (местное) в сотых долях секунды, прошедшее с начала суток.

Пример:

алг
нач
· **цел** мс
· мс := **время**

```
цел с, ч, м
· с := div(мс, 100)
· м := div(с, 60)
· ч := div(м, 60)
· с := с - м*60
· м := м - ч*60
· вывод "Текущее время: ", ч, " часов, ", м, " минут ", с, " секунд"
кон
```

5.4.4 клав

Синтаксис: **алг цел** клав

Ожидает нажатия на клавишу и возвращает её код.

Пример:

```
алг
нач
· цел а
· вывод "Нажмите клавишу...", нс
· а := клав
· вывод "Код нажатой клавиши равен ", а, нс
кон
```

Коды клавиш, имеющих символьное представление, совпадают с Юникодами соответствующих клавиш. Коды клавиш, не имеющих символьное представление, приведены в таблице:

Клавиша	Код
Tab	16777217
Backspace	16777219
Enter	16777220
Enter на цифровом блоке клавиатуры	16777221
Insert	16777222
Delete	16777223
Pause	16777224
Print Screen	16777225
Home	16777232
End	16777233
Стрелка влево	16777234
Стрелка вверх	16777235
Стрелка вправо	16777236
Стрелка вниз	16777237
Page Up	16777238
Page Down	16777239
Shift	16777248
Ctrl (на Macintosh - Command)	16777249
Meta — логотип Windows (на Macintosh - Control)	16777250
Alt	16777251
Caps Lock	16777252
Num Lock	16777253
Scroll Lock	16777254
F1	16777264
F2	16777265
F3	16777266
F4	16777267
F5	16777268
F6	16777269
F7	16777270
F8	16777271
F9	16777272
F10	16777273
F11	16777274
F12	16777275
F13	16777276
F14	16777277
F15	16777278

6 Исполнитель Робот

6.1 Общие сведения

Система команд исполнителя «Робот» включает:

- 5 команд, вызывающих действия Робота (**влево**, **вправо**, **вверх**, **вниз**, **закрасить**)
- 10 команд проверки условий:
  - 8 команд вида **[слева/справа/снизу/сверху] [стена/свободно]**

– 2 команды вида **клетка [закрашена/чистая]**

- 2 команды измерения (**температура**, **радиация**)

Командам **влево**, **вправо**, **вверх**, **вниз**, **закрасить** соответствуют алгоритмы-процедуры языка КуМир. Остальным командам соответствуют алгоритмы-функции, тип этих функций указан ниже.

6.2 Команды-действия

Команда	Описание
влево	Перемещает робота на одну клетку влево. Если слева стена, выдает отказ.
вправо	Перемещает робота на одну клетку вправо. Если справа стена, выдает отказ.
вверх	Перемещает робота на одну клетку вверх. Если сверху стена, выдает отказ.
вниз	Перемещает робота на одну клетку вниз. Если снизу стена, выдает отказ.
закрасить	Делает клетку, в которой находится робот, закрашенной.

Пример:

```
алг
нач
· вправо
· вниз
· влево
· вверх
· закрасить
кон
```

6.3 Команды-проверки

Команда	Описание
лог слева свободно	Возвращает **да**, если робот может перейти влево, иначе — **нет**.
лог справа свободно	Возвращает **да**, если робот может перейти вправо, иначе — **нет**.
лог сверху свободно	Возвращает **да**, если робот может перейти вверх, иначе — **нет**.
лог снизу свободно	Возвращает **да**, если робот может перейти вниз, иначе — **нет**.
лог слева стена	Возвращает **да**, если слева от робота находится стена, иначе — **нет**.
лог справа стена	Возвращает **да**, если справа от робота находится стена, иначе — **нет**.
лог сверху стена	Возвращает **да**, если сверху от робота находится стена, иначе — **нет**.
лог снизу стена	Возвращает **да**, если снизу от робота находится стена, иначе — **нет**.
лог клетка закрашена	Возвращает **да**, если клетка закрашена, и **нет**, если клетка не закрашена.
лог клетка чистая	Возвращает **нет**, если клетка закрашена, и **да**, если клетка не закрашена.

6.4 Команды-измерения

Команда	Описание
вещ радиация	Возвращает значение радиации в клетке, где находится робот.
вещ температура	Возвращает значение температуры в клетке, где находится робот.

7 Исполнитель Чертежник

7.1 Общие сведения

Система команд исполнителя «Чертежник» включает 6 команд:

- опустить перо
- поднять перо
- сместиться на вектор (**вещ** dX, dY)
- сместиться в точку (**вещ** x, y)
- установить цвет (**лит** цвет)
- надпись (**вещ** ширина, **лит** текст)

7.2 Описания команд

Команда	Описание
опустить перо	Переводит чертежника в режим перемещения с рисованием.
поднять перо	Переводит чертежника в режим перемещения без рисования.
сместиться на вектор (**вещ** dX, dY)	Перемещает перо на dX вправо и dY вверх.
сместиться в точку (**вещ** x, y)	Перемещает перо в точку с координатами (x, y).
установить цвет (**лит** цвет)	Устанавливает цвет пера. Допустимые цвета: "черный", "белый", "красный", "оранжевый", "желтый", "зеленый", "голубой", "синий", "фиолетовый".
надпись (**вещ** ширина, **лит** текст)	Выводит на чертеж текст, начиная от текущей позиции пера. В конце выполнения команды перо находится на правой нижней границе текста (включая отступ после последнего символа). Ширина знакоместа измеряется в условных единицах чертежника. Это ширина буквы вместе с отступом после нее.

Примечание 1. Поднять (опустить) перо — сокращение от полной формы «сделать так, чтобы перо оказалось поднятым (опущенным)». Если перо, например, поднято, то после выполнения команды поднять перо, оно просто останется поднятым.

Примечание 2. Если в момент вызова функции установить цвет значение ее аргумента не совпадает ни с одним из перечисленных 9 допустимых цветов, то выдается отказ и выполнение программы прерывается.

Пример:

```
алг
нач
· установить цвет("красный")
· опустить перо
```

· сместиться на вектор(1,1)
· надпись(0.5, "Рис. 1")
кон

8 Исполнитель Файлы

8.1 Общие сведения

Система команд исполнителя «Файлы» включает:

- 4 команды-приказа (создание файлов, открытие/закрытие файлов)
- 2 команды проверки условий (существования файла и достижения конца файла)
- 2 оператора ввода-вывода

8.2 Описания команд

- создать файл (**арг лит** имяФайла)

  Создает новый пустой файл текущем каталоге ввода-вывода.

- открыть на чтение (**арг лит** имяФайла, **арг цел** ключ)

  Открывает файл на чтение и присваивает ему идентификатор.

- открыть на запись (**арг лит** имяФайла, **арг цел** ключ)

  Открывает файл на запись и присваивает ему идентификатор.

- **ф_ввод** ключ, ...

  Оператор. Ввод данных из файла с идентификатором *ключ*.

- **ф_вывод** ключ, ...

  Оператор. Вывод данных в файл с идентификатором *ключ*.

- **лог** конец файла (**арг цел** ключ)

  Проверяет, достигнут ли конец файла.

- закрыть (**арг цел** ключ)

  Закрывает файл после того, как он был открыт на чтение или на запись.

- **лог** существует файл (**арг лит** имяФайла)

  Проверяет, существует ли файл в текущем каталоге ввода-вывода.

Пример:

```
использовать Файлы
алг
нач
· цел Ключ
· лит имяФайла
· имяФайла := "t1.txt"
· создать файл(имяФайла)
· открыть на чтение(имяФайла, Ключ)
```

```
· цел n
· n := 0
· вещ таб a[1:100]
· вещ buf
· нц пока не конец файла(Ключ)
· · n := n+1
· · ф_ввод Ключ, buf
· · вывод buf, нс
· · a[n] := buf
· кц
кон
```

9 Справочник

9.1 Команды Робота

вверх	вниз
вправо	влево
закрасить	

вещ температура
вещ радиация

лог сверху стена	**лог** сверху свободно
лог снизу стена	**лог** снизу свободно
лог справа стена	**лог** справа свободно
лог слева стена	**лог** слева свободно
лог клетка закрашена	**лог** клетка чистая

9.2 Команды Чертежника

поднять перо
опустить перо
сместиться в точку (**арг вещ** x, y)
сместиться на вектор (**арг вещ** x, y)
установить цвет (**арг стр** ц)
надпись (**арг вещ** ширина, **арг лит** стр)

9.3 Команды для работы с файлами

создать файл (**арг лит** имяФайла)
открыть на чтение (**арг лит** имяФайла, **арг цел** ключ)
открыть на запись (**арг лит** имяФайла, **арг цел** ключ)
ф_ввод
ф_вывод
лог конец файла (**арг цел** ключ)
закрыть (**арг цел** ключ)
лог существует файл (**арг лит** имяФайла)

9.4 Общий вид алгоритма

алг имя (*аргументы и результаты*)
· **дано** *условия применимости алгоритма*

· **надо** *цель выполнения алгоритма*
нач
· *тело алгоритма*
кон

9.5 Команды алгоритмического языка

нц *число повторений* **раз** · *тело цикла (последовательность команд)* **кц**
нц пока *условие* · *тело цикла (последовательность команд)* **кц**
нц для i **от** $i1$ **до** $i2$ · *тело цикла (последовательность команд)* **кц**

если *условие* · **то** *серия 1* · **иначе** *серия 2* **все**	**если** *условие* · **то** *серия 1* **все**
выбор *условие* · **при** *условие 1*: *серия 1* · **при** *условие 2*: *серия 2* · . . . · **при** *условие n*: *серия n* · **иначе** *серия n+1* **все**	**выбор** *условие* · **при** *условие 1*: *серия 1* · **при** *условие 2*: *серия 2* · . . . · **при** *условие n*: *серия n* **все**

	утв *условие*
	ввод *имена величин*
	вывод *тексты, имена величин, выражения,* **нс**
	выход
вызов:	*имя алгоритма (аргументы и имена результатов)*
присваивание:	*имя величины := выражение*

9.6 Типы величин

	Таблицы:
целые **цел**	целые **цел таб**
вещественные **вещ**	вещественные **вещ таб**
логические **лог**	логические **лог таб**
символьные **сим**	символьные **сим таб**
литерные **лит**	литерные **лит таб**
Пример описания: **цел** i, j, **лит** t, **вещ таб** a[1:50]	

9.7 Виды величин

- аргументы (**арг**) — описываются в заголовке алгоритма
- результаты (**рез**) — описываются в заголовке алгоритма

- значения функций (**знач**) — описываются указанием типа перед именем алгоритма-функции
- локальные — описываются в теле алгоритма, между **нач** и **кон**
- общие — описываются после строки **исп** исполнителя, до первой строки **алг**

9.8 Общий вид исполнителя

исп имя
· *вступление исполнителя:*
· *– описание общих величин исполнителя*
· *– команды для задания начальных значений общих величин и т. п.*
· *алгоритмы исполнителя*
кон_исп

9.9 Операции и стандартные функции для работы с числами

Название операции или функции	Форма записи
сложение	x + y
вычитание	x - y
умножение	x * y
деление	x / y
возведение в степень	x ** y
корень квадратный	sqrt(x)
абсолютная величина	abs(x) и iabs(x)
знак числа (-1, 0 или 1)	sign(x)
синус	sin(x)
косинус	cos(x)
тангенс	tg(x)
котангенс	ctg(x)
арксинус	arcsin(x)
арккосинус	arccos(x)
арктангенс	arctg(x)
арккотангенс	arcctg(x)
натуральный логарифм	ln(x)
десятичный логарифм	lg(x)
степень числа e ($e \approx 2.718181$)	exp(x)
минимум из чисел x и y	min(x,y)
максимум из чисел x и y	max(x,y)
остаток от деления x на y (x, y — целые)	mod(x,y)
частное от деления x на y (x, y — целые)	div(x,y)
целая часть числа x	int(x)
случайное число в диапазоне от 0 до x	rnd(x)

9.10 Логические операции

Название операции	Форма записи	Пример
конъюнкция	**и**	а **и** б
дизъюнкция	**или**	а **или** б
отрицание	**не**	**не** а, завтра **не** будет дождь

9.11 Операции для работы со строками

Название операции	Пример
слияние	а+б
вырезка	а[3:5]
взятие символа	а[3]

9.12 Другие встроенные алгоритмы

Функция	Форма вызова
Строковое представление целого числа	цел_в_лит(х)
Строковое представление вещественного числа	вещ_в_лит(х)
Перевод строки в целое число	лит_в_цел(стр, успех)
Перевод строки в вещественное число	лит_в_вещ(стр, успех)
Длина строки	длин(стр)
Код символа в таблице КОИ-8	код(с)
Символ таблицы КОИ-8	символ(х)
Код символа в таблице Юникод	юникод(с)
Символ таблицы Юникод	символ2(х)
Код нажатой клавиши	клав
Текущее время в миллисекундах	время
Приостановка выполнения программы	пауза
Остановка выполнения программы	стоп

Предметный указатель