Глава 1

Язык КуМир

1.1 Структура программы

1.1.1 Программа

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. 1.1.4).

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

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.

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
```

1.1.2 Описание алгоритма

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**)

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. 1.2.5 и 1.3.4.

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**.

1.1.3 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.

1.1.4 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.

1.1.5 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.

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
```

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
```

Включение строк из текстового файла

Часто используемые в различных программах строки можно хранить в отдельном текстовом файле и включать в программу с помощью команды ВКЛЮЧИТЬ.

Пример:

Содержимое файла fill_A_1_N.kum:

```
цел x
нц для x от 1 до N
· A[x]:=x
кц
```

Содержимое файла use01.kum:

```
алг  вывод квадратов
· цел N, n
· N:=10
· целтаб A[1:N]
· ВКЛЮЧИТЬ 'fill_A_1_N.kum' | массив A заполняется числами
· нц для x от 1 до N
· · вывод A[n] * A[n], ' ' | 1, 4, 9, 16, и т. д.
· кц
кон
```

Содержимое файла use02.kum:

```
алг  вывод кубов
· цел N, n
· N:=10
· целтаб A[1:N]
· ВКЛЮЧИТЬ 'fill_A_1_N.kum' | массив A заполняется числами
· нц для x от 1 до N
· · вывод A[n] * A[n] * A[n], ' ' | 1, 8, 27, 64, и т. д.
· кц
кон
```

1.1.6 Комментарии

```
алг
· # Это алгоритм вычисления суммы двух чисел
нач
· цел a, б | объявляем величины
· ввод a, б | вводим значения с клавиатуры
· вывод a+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.

1.2 Имена, величины и выражения

1.2.1 Имена

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. 1.2.1).

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++`.

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**.

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**.

1.2.2 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 -**МЦЕЛ** до **МЦЕЛ**, где $\mathbf{МЦЕЛ} = 2147483647 = 2^{31} - 1$
- **вещ** — принимает вещественные значения между -**МВЕЩ** до **МВЕЩ**, где **МВЕЩ** – это число немного меньшее, чем 2^{1024}; $\mathbf{МВЕЩ} \approx 1.797693 \times 10^{308}$
- **лог** — принимает значения **да** или **нет** (внутреннее представление – **да**=1, **нет**=0)
- **сим** — значением может быть любой литеральный символ (практически любой символ, см. 1.2.1)
- **лит** — значением может быть строка литеральных символов

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

Значения величин **МЦЕЛ** и **МВЕЩ** определяются способом представления чисел в современных компьютерах и примерно одинаковая в большинстве современных языков программирования.

Язык КуМир содержит встроенные функции преобразования числовых типов в текстовые и наоборот (см. 1.5.1). При необходимости значения целого типа автоматически переводятся в вещественные, а символьные — в текстовые. Для преобразования вещественных значений в целые используется встроенная функция int (см. 1.5.2).

1.2.3 Константы

Виды констант

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

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.

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.

Логические константы

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

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.

1.2.4 Величины

Общие сведения

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

Имя величины служит для обозначения величины в алгоритме (см. 1.2.1).

Тип величины показывает, какие значения может принимать величина, и какие операции можно с ней выполнять (см. 1.2.2).

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.

Простые величины и таблицы. Описания величин

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

Характеристики простых величин описаны в 1.2.4. 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.

Описания величин

Каждая величина должна иметь описание. Это может быть сделано:

- с помощью оператора описания
- при задании формальных параметров алгоритма (см. 1.1.3)

В описании задаются перечисленные выше статические характеристики переменной.

Кроме того, в алгоритмах-функциях используется простая переменная **знач**, ее тип определяется типом функции, (см. 1.1.2). 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.

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

1.2.5 Выражения

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**

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

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;
- <= — меньше или равно;
- >= — больше или равно;
- <> — не равно.

Полный список базовых операций и их описания приведены в A.9.

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**.

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"

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($X_1, \ldots, X_n$).

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)

1.3 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 1.2.4.

1.3.1 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.

1.3.2 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
```

1.3.3 Ввод-вывод

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
```

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
```

1.3.4 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

1.3.5 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–

1.4 Составные команды

1.4.1 Команды ветвления

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

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.

1.4.2 Команды цикла

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. 1.3.5). При выполнении этой команды содержащий ее цикл будет завершен.

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. 1.3.5). При выполнении этой команды содержащий ее цикл будет завершен.

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. 1.3.5). При выполнении этой команды содержащий ее цикл будет завершен.

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. 1.3.5). При выполнении этой команды содержащий ее цикл будет завершен.

Цикл «нц-кц»

Общий вид цикла:
нц
· тело_цикла
кц

Пример:
нц
· a := a + 1
· **если** a>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.

1.5 Встроенные алгоритмы

1.5.1 Текстовое представление чисел

цел_в_лит

Синтаксис: **алг лит** цел_в_лит(**цел** x)

Возвращает строковое представление x.

Пример:
алг
нач
цел a
· **лит** б
· a := 5
· б := **цел_в_лит**(a)

```
· вывод б
кон
```

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
```

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
```

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
```

1.5.2 Математика

sqrt

Синтаксис: **алг вещ** sqrt(**вещ** 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
```

abs

Синтаксис: **алг вещ** abs(**вещ** x)
Абсолютная величина вещественного числа x ($|x|$).
Пример:

```
вещ a, б
алг
нач
· ввод a,б
· a := a+б
· a := abs(a)
· вывод"Модуль суммы чисел равен ", a
кон
```

iabs

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

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

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
```

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
```

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
```

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
```

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
```

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
```

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
```

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
```

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
```

ln

Синтаксис: **алг вещ** ln(**вещ** x)

Натуральный логарифм x

Пример:

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

lg

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

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

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 e в степени ", a, " равно ", x
кон
```

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
```

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
```

```
· c1 := max(a,б)
· c2 := min(a,б)
· вывод c1, нс
· вывод c2, нс
кон
```

mod

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

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

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(a,б)
· y := mod(a,б)
· вывод "a/б=", x, " с остатком ", y
кон
```

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(a)
· вывод "Целая часть ", а, " равна ", б
кон
```

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
```

МВЕЩ

Синтаксис: **алг вещ** МВЕЩ

Самое большое вещественное число, которое можно использовать в языке КуМир. **МВЕЩ** $\approx 1.797693 \times 10^{308}$ (это немного меньше, чем 2^{1024}). Величина этого числа определяется способом представления чисел в современных компьютерах и примерно одинаковая в большинстве современных языков программирования.

Пример:

```
алг Самое большое вещественное число в КуМире
нач
· вещ щ
· щ := МВЕЩ
· вывод 'Самое большое вещественное число в КуМире – это число ', щ
кон
```

МЦЕЛ

Синтаксис: **алг цел** МЦЕЛ

Самое большое целое число, которое можно использовать в языке КуМир. **МЦЕЛ** $= 2^{31} - 1$. Величина этого числа определяется способом представления чисел в современных компьютерах и примерно одинаковая в большинстве современных языков программирования.

Пример:

```
алг Самое большое целое число в КуМире
нач
· цел ц
· ц := МЦЕЛ
· вывод 'Самое большое целое число в КуМире – это число ', ц
кон
```

1.5.3 Обработка строк

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
```

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
```

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
```

юникод

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

Возвращает номер символа *с* в таблице Юникода.

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
```

1.5.4 Система

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
```

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
```

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
```

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

1.6 Исполнитель Робот

1.6.1 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.

1.6.2 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
```

1.6.3 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.

1.6.4 Команды-измерения

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

1.7 Исполнитель Чертежник

1.7.1 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)

1.7.2 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
```

1.8 Исполнитель Файлы П

1.8.1 Описания команд

- **алг создать файл (арг лит имяФайла)**

  Создает новый пустой файл в текущем каталоге ввода-вывода.

- **алг цел открыть на чтение (арг лит имяФайла)**

  Открывает файл на чтение и возвращает его ключ.

- **алг цел открыть на запись (арг лит имяФайла)**

  Открывает файл на чтение и возвращает его ключ.

- **алг цел открыть на добавление (арг лит имяФайла)**

  Открывает файл на чтение и возвращает его ключ.

- **алг начать чтение (арг цел ключ)**

  Перемещает текущую позицию чтения в начало файла, позволяя прочитать открытый файл заново.

- **алг закрыть (арг цел ключ)**

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

- **Фввод ключ, ...**

  Оператор. Ввод данных из файла с идентификатором *ключ*. О формате входных данных см. 1.8.2.

- **Фвывод ключ, ...**

  Оператор. Вывод данных в файл с идентификатором *ключ*. Работает аналогично оператору вывод.

- **алг записать байт (арг цел байт, арг цел ключ)**

  Записывает один байт в файл.

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

  Проверяет, достигнут ли конец файла.

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

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

- **алг удалить файл (арг лит имяФайла)**

  Удаляет файл в текущем каталоге ввода-вывода.

Пример:

```
использовать Файлы П
алг
нач
· если не существует файл("a.txt")
· · то создать файл("a.txt")
· все
· цел r, i
· r:=открыть на запись("a.txt")
· Фвывод r, 5
· закрыть(r)
· r:=открыть на чтение("a.txt")
· утв не конец файла(r)
· Фввод r, i
· утв i=5
· начать чтение(r)
· Фввод r, i
· утв i=5
· закрыть(r)
· вывод (i=5), нс
кон
```

1.8.2 Формат входных файлов

Файл, который читается с помощью оператора **Фввод**, должен иметь следующую структуру:

- данные, записанные в файле, отделяются друг от друга запятой;

- символы и строки должны быть заключены в двойные либо одинарные кавычки;
- после символьных и литерных данных допускается использовать в качестве разделителя не запятую, а символ перевода строки.

Пример входного файла

```
1,
2,1.1,2.2,
'c','т"
да,
нет,
"один"
'два'
```

Глава 2

Практикумы

2.1 Введение

2.1.1 Общие сведения

Система КуМир позволяет учителю создавать, а ученику выполнять учебные практикумы по информатике на базе системы КуМир.

2.1.2 Практикумы. Описание практикумов

Практикум – это набор задач, для которых возможна автоматическая проверка решения. *Задание* предлагает ученику написать программу, удовлетворяющую некоторой спецификации. *Описание задания* включает в себя:

1. текстовое описание задания;
2. шаблон («заготовку») программы в виде файла .kum
3. программу тестирования и, если нужно, обстановки, используемые при тестировании (см п 2.1.5);
4. описание условий, при которых задание становится доступным ученику для выполнения.

На множестве заданий может быть задана структура дерева – так же, как в традиционных учебниках (части, главы, разделы и т.п.). Узлам дерева могут быть приписаны поясняющие тексты, см. 2.2 Подготовка курса и его представление подробнее описаны в п 2.3.

2.1.3 Прохождение курса. Текущие оценки. Методика

При прохождении (выполнении) практикума учеником, КуМир для каждого задания помнит текущую *оценку* – целое число от 0 до 10. Оценка показывает, выполнял ли ученик данное задание и, если да, - насколько успешно он справился с этим заданием. Оценка выставляется, когда ученик выдает запрос на тестирование программы (см. п. 2.1.5).

Описание практикума включает в себя методику – указание на то, какие задания доступны для выполнения в зависимости от уже полученных оценок. Простейшие методики – это "свободная"методика, когда, независимо от полученных оценок, доступны все задания, и "игровая», когда задания линейно упорядочены и очередное задание доступно только после выполнения всех предыдущих. В более сложных случаях методика может

выделять в практикуме, например, задачи повышенной трудности, которые становятся доступными только при хороших оценках за задания основного курса, и вспомогательные задачи, которые предлагаются ученику, если он не справляется с некоторыми основными заданиями.

2.1.4 Проверка задания. Заготовка программы

Проверка правильности выполнения задания выполняется с помощью тестирования. Разработчик курса для каждого задания готовит систему тестов. Эта система включает:

1. тестирующую программу – алгоритм без параметров со стандартным именем *@тестирование*;

2. (если нужно) набор тестовых обстановок.

При получении запроса на тестирование программы, КуМир вызывает алгоритм *@тестирование*. Этот алгоритм может, если нужно, вызывать другие алгоритмы. Алгоритм *@тестирование* и его вспомогательные алгоритмы должны быть включены в заготовку программы] как неудаляемые (и, возможно, невидимые для ученика) фрагменты. Эти алгоритмы могут вызывать программу пользователя (ее заголовок фиксируется в заготовке как неудаляемый, но видимый, фрагмент) шаблона. Подробнее о тестировании см. п. 2.2.4, о подготовке соответствующей части шаблона – см. п. 2.3.3.

2.1.5 Сеанс работы с использованием модуля поддержки курсов.

Для начала работы с практикумом, необходимо открыть окно «Практикум», это можно сделать из меню «Инструменты».

Пользователь загружает файл с копией описания нужного практикума. Это может быть стандартная копия или личная копия пользователя (рабочая тетрадь), которая, кроме описания курса (см. выше п. 1.4) включает

1. ранее полученные учеником оценки за выполнение заданий и

2. созданные ранее учеником версии программ.

Личная копия может быть создана пользователем во время предыдущих сеансов работы.

Далее ученик с помощью окна практикумов поочередно выбирает задания и выполняет их, используя систему КуМир. После того, как (по мнению ученика) программа готова, ученик запускает процедуру тестирования. По итогам тестирования выставляется оценка качества выполнения и, соответственно, меняется список доступных для выполнения заданий в окне курсов.

Сеанс работы может быть прекращен по желанию ученика в произвольный момент. При этом он может сохранить текущее состояние работы над практикумом. Подробнее см. п. 2.2

2.2 Сеанс работы

2.2.1 Окно практикумов

Окно практикумов изображено на рис.2.1 (исходное состояние) и рис. 2.2 (состояние во время работы. В исходном состоянии ученику доступно только одно действие – загрузить практикум.

После того, как практикум выбран, в левой части окна показывается список заданий, в правой – описание текущего задания.

Описание практикума имеет древовидную структуру, ученик может «сворачивать» и «разворачивать» отдельные разделы (узлы дерева). Выбор задания производится щелчком мыши. Задания, доступные для выполнения, выделены черным.

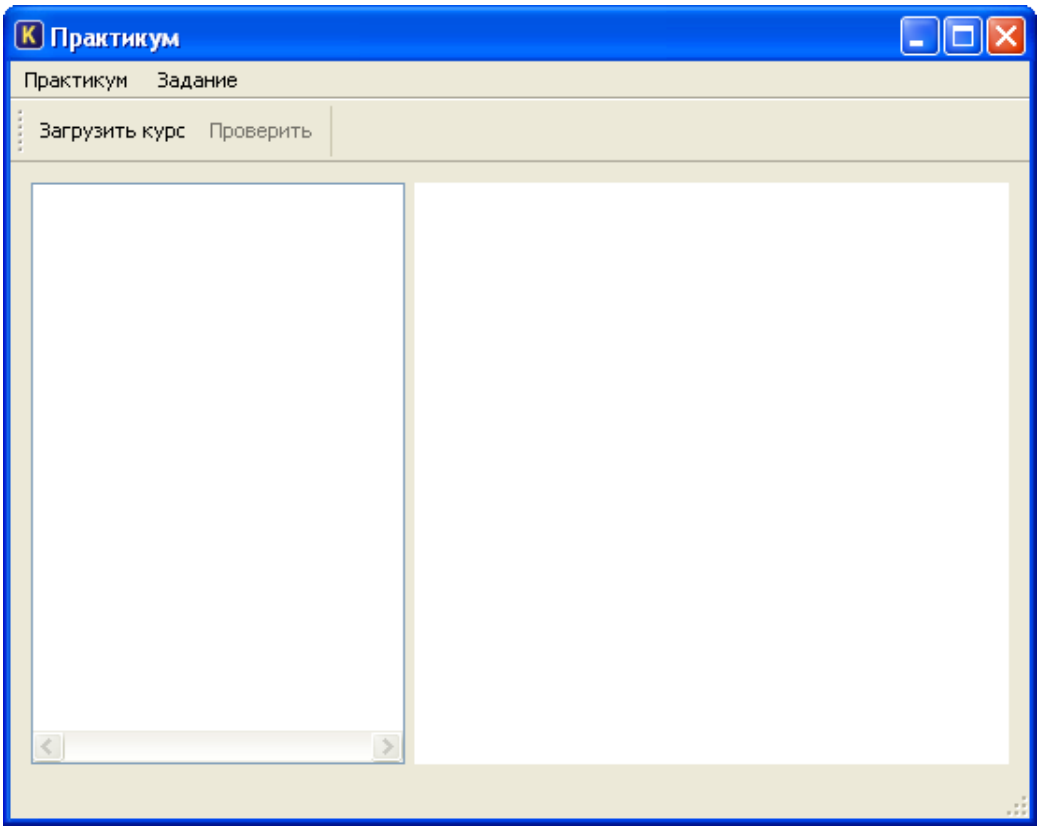

Рис. 2.1: Окно практикумов. Исходное состояние

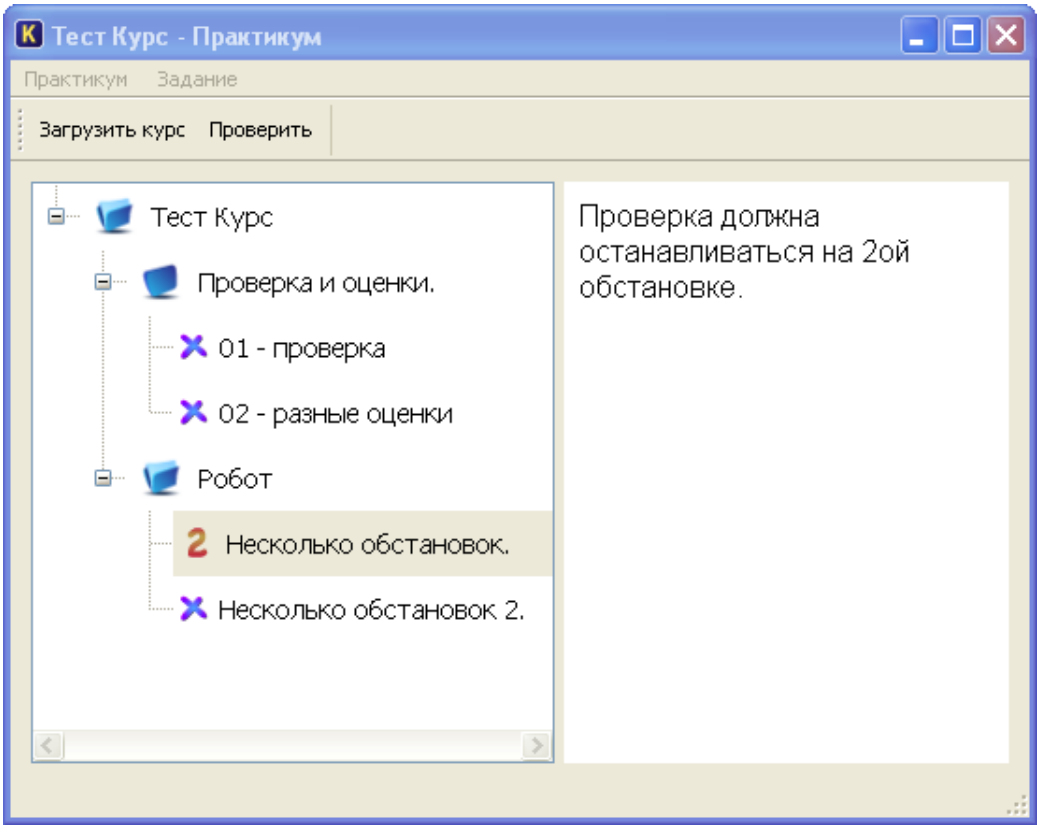

Рис 2.2 Окно практикумов. Тестовый курс

2.2.2 Общее описание сеанса

После того, как появилось окно «Практикум», пользователь выбирает практикум, далее он работает с выбранным практикумом – поочередно выбирает и выполняет задания. В каждый момент пользователю доступен для выполнения определенный набор заданий (см. выше).

В произвольный момент работы пользователь может сохранить состояние своей работы в виде личной копии практикума (рабочей тетради). Впоследствии эта копия может быть загружена и работа будет продолжена с того состояния, в котором она была прервана.

2.2.3 Выполнение задания

Пользователь выбирает задание, используя левое поле окна курса. После этого в правом поле этого окна появляется словесное описание задания, а в поле программы окна системы КуМир устанавливается обстановки исполнителей и начальный текст программы – сохраненный во время предыдущего сеанса текст или (при первом запуске задания) исходный шаблон. Если требуется , загружается обстановка нужного исполнителя. Далее пользователь обычным образом создает нужную программу и, когда, по его мнению, программа готова, вызывает проверку (см. ниже).

По результатам проверки за задание выставляется оценка и, возможно, изменяется множество доступных заданий. Оценки показываются на дереве заданий рядом с соответствующими заданиями.

Пользователь может продолжить выполнение задания (если проверка дала отрицательный результат) или выбрать новое задание.

2.2.4 Проверка задания

Для проверки текущей программы необходимо нажать кнопку «Проверить» главного меню окна курсов, если до этого рабочая тетрадь не была создана то будет выдано предложение создать тетрадь. Задание может содержать несколько обстановок исполнителей. Если на очередной обстановке в работе программы обнаружена ошибка, то тестирование прерывается, в поле ввода-вывода выводится соответствующее сообщение и обстановка, на которой обнаружена ошибка предъявляется ученику. При каждом тестировании проверяется работоспособность программы на всех тестовых обстановках, независимо от результатов предыдущих тестирований. Можно проверить задание только на текущей обстановке (без выставления оценки), нажав Ctrl+t.

Проверка работы программы ученика (при данной обстановке исполнителя) состоит в вызове подготовленного учителем алгоритма-функции без параметров с именем *@тестирование*; этот алгоритм готовит разработчик курса (см. пп.2.1.5 и 2.3.3). Алгоритм *@тестирование* возвращает целое число – оценку за работу программы при данной обстановке. Общая оценка – это минимум из оценок по всем обстановкам.

Как отмечалось выше, тестирующие алгоритмы заданий готовит автор курса. Поэтому критерии оценок полностью находятся в его руках.

Система запоминает тестировавшуюся версию программы. При желании учитель может посмотреть эту версию, внести свои комментарии и т.п. О сохранении версий программ см. ниже.

2.2.5 Сохранение версий программы

Ученик может в любой момент сохранить текущее состояние программы в файл, как это предусмотрено возможностями системы КуМир (или другой используемой управляющей системы).

В то же время в рабочей тетради для каждого задания запоминаются следующие версии программы:

1. исходная заготовка («шаблон»);
2. версия программы, с которой был начат сеанс (см. п. 1.5);
3. последнюю тестировавшуюся версию программы.

Ученик может вернуться к каждой из указанных версий программы с помощью меню «Задание» окна практикумов.

Кроме того, в момент остановки работы над заданием (т.е. при выборе нового задания или проверки) запоминается текущее состояние программы. При повторном обращении к заданию работа будет продолжена с того же состояния, в котором она была прервана.

2.2.6 Сохранение состояния курса

В любой момент сеанса ученик может сохранить текущее состояние работы над практикумом. Состояние сохраняется в т.н. *личном* файле ученика – рабочей тетради. В этом файле хранится ссылка на описание практикума (самого практикума не хранится), оценки по заданиям и созданные учеником для каждого задания версии программ (последняя тестировавшаяся и «текущая», т.е. версия на момент прерывания работы над заданием).

При сохранении состояния ученике, как обычно, выбирает имя личного файла, в которое происходит сохранение. При загрузке курса из личного файла сначала загружается основной практикум, а затем результаты предыдущей работы ученика.

2.3 Подготовка курса

2.3.1 Файл описания курса

Файл описания курса - это XML файл. В нем хранится дерево заданий. Для создания таких файлов в поставке Кумир имеется утилита «Редактор практикумов». Для каждого задания описывается:

1. Номер задания.
2. Имя необходимой УС. Пока только Кумир.
3. Имя задания
4. Текстовое описание задания или ссылка на html файл.
5. Ссылка на файл стартовой программы.
6. Список имен необходимых исполнителей.
7. Ссылки на файлы обстановок исполнителей.
8. Текущая оценка.
9. Минимальные и максимальные оценки по другим заданиям, необходимые для этого задания (указываются только задания, от которых зависит данное задание).

2.3.2 Файлы заданий

Это файлы, на которые имеются ссылки в файле описания курса (файлы обстановок, HTML-файлы описаний, файлы программ и другие файлы необходимые для выполнения задания). Для них может быть указан любой путь. Но на практике удобно, чтобы они лежали в том же каталоге, что и файл описания практикума или в подкаталоге этого каталога.

2.3.3 Подготовка программы-заготовки

Программу-заготовку рекомендуется готовить с помощью системы КуМир, запущенной в учительском режиме (ключ -t). В заготовку должны обязательно входить следующие фрагменты, защищенные от изменения учеником:

1. заголовок требуемого алгоритма;
2. тестирующий алгоритм-функция без параметров **цел** *@тестирование*.

Заголовок должен быть виден ученику, а алгоритм *@тестирование*, как правило (но не обязательно!) скрыт от ученика. Сделать фрагменты программы защищенными от изменения и невидимыми можно в редакторе КуМира, используя правую кнопку мыши. Подробнее о подготовке алгоритма тестирования см. ниже. Кроме указанных необходимых элементов, в заготовку можно включать по желанию разработчика и другие элементы программы – описания величин, управляющие конструкции и т.п. Делать весь текст заготовки защищенным от изменений не обязательно.

Пример заготовки программы – см. DemoKurs.zip

2.3.4 Подготовка тестирующего алгоритма

Тестирующий алгоритм – часть программы-заготовки. Это алгоритм-функция языка КуМир типа **цел**. Возвращаемое значение является оценкой, которую получит ученик. Тестирующий алгоритм должен иметь имя *@тестирование*. Тестирующий алгоритм может вызывать другие алгоритмы. Имена этих алгоритмов должны начинаться с символа , чтобы не создавать конфликтов с именами алгоритмов пользователя. Эти имена не будут показываться ученику при вызове подсказки, которая выдает список доступных алгоритмов.

Тестирующий алгоритм и вызываемые им алгоритмы разработчика должны быть расположены в конце программы-заготовки. Это связано с тем, что в системе КуМир скрыть от ученика можно лишь фрагмент текста от указанной позиции до конца программы.

Поддержка практикумов обеспечивает многократный вызов тестирующего алгоритма, если задан список тестовых обстановок. Нужно иметь Это нужно иметь в виду при создании тестирующего алгоритма. Для проверки заданий, связанных с исполнителем Робот, существуют дополнительные команды см. Список алго-ритмов Робота в системе Кумир . Эти команды нельзя использовать вне тестиру-ющего алгоритма.

2.3.5 Подсказки

Возможность что-то подсказать ученику – важный элемент методики, заложенной в практикум.

1. В заготовку программы можно включать не только заголовок, но и элементы решения.

2. Можно включать в практикум «задания», для которых программа-решение почти полностью включена в заготовку. Такая программа может быть доступна всегда. Другая возможность – задание-подсказка становится доступным только, если ученик попробовал выполнить «основное» задание. О наличии такого вспомогательного задания-подсказки можно сообщить в поясняющем тексте задания.

Глава 3

Исполнитель Робот

3.1 Введение

3.1.1 Обстановки Робота

Исполнитель Робот существует в некоторой *обстановке* — прямоугольном поле, разбитом на клетки, между которыми могут стоять стены. Обстановка, в которой находится Робот, называется *текущей* обстановкой Робота. Кроме того, определена еще одна обстановка Робота — *стартовая* обстановка. Стартовая обстановка используется при управлении Роботом из программы, подробнее см. ниже 3.1.3.

Робот может передвигаться по полю, закрашивать клетки, измерять температуру и радиацию. Робот не может проходить сквозь стены, но может проверять, есть ли рядом с ним стена. Робот не может выйти за пределы прямоугольника (по периметру стоит «забор»). Подробно система команд Робота описана ниже.

Удобно представлять себе, что Робот существует всегда. В частности, когда начинается сеанс работы системы Кумир, Робот уже существует и для него определены и текущая, и стартовая обстановка (они совпадают).

Обстановки Робота могут храниться в файлах специального формата (расширение .fil).

3.1.2 Окно наблюдения за Роботом

В Кумире есть специальное устройство — *Окно наблюдения за Роботом* (иногда для краткости будем говорить *Окно Робота*). В этом окне всегда видна текущая обстановка Робота, включая положение самого Робота. Подробнее об окне Робота см. 3.2.

3.1.3 Управление Роботом из программы

Кумир-программа, управляющая Роботом, должна начинаться со строки `использовать Робот` (подробнее — см. описание языка Кумир). При выполнении этой строки Кумир помещает Робота в некоторую заранее определенную обстановку. Эта обстановка и называется *стартовой* обстановкой Робота.

Таким образом, в каждый момент сеанса работы системы Кумир определены две обстановки Робота — *текущая* и *стартовая*. Текущая обстановка в любой момент показывается в окне наблюдения за Роботом.

3.1.4 Как установить стартовую обстановку

В системе Кумир есть средства, с помощью которых Школьник может задать нужную ему стартовую обстановку. Это можно сделать двумя способами. Один способ — загрузить

стартовую обстановку из указанного Школьником файла. Другой способ — редактировать существующую стартовую обстановку с помощью специального редактора стартовых обстановок.

Редактор стартовых обстановок является частью системы Кумир. Редактирование обстановки происходит в отдельном окне (окно редактирования стартовой обстановки), структура этого окна аналогична структуре окна наблюдения Робота. Редактируемая обстановка может быть сохранена в файл или непосредственно использоваться в качестве стартовой обстановки. Подробнее редактирование стартовых обстановок описано в 3.4.

3.1.5 Ручное управление Роботом

В систему Кумир входит пульт ручного управления Роботом (см. 3.5). Этот пульт позволяет вручную управлять Роботом — выдавать команды, входящие в систему команд Робота. Использовать пульт можно в любое время, кроме тех временных промежутков, когда происходит непрерывное выполнение Кумир-программы. В частности, Роботом можно управлять с пульта в те моменты, когда выполнение Кумир программы приостановлено (система Кумир находится в состоянии «Пауза»).

3.2 Окно наблюдения за Роботом

3.2.1 Видимое и скрытое состояние окна

Окно наблюдения за Роботом создается в момент начала сеанса работы системы Кумир и доступно до окончания сеанса. Во время сеанса работы окно может находиться в одном из двух состояний — видимо или скрыто (с отображением на панели задач или без).

В верхнем правом углу окна Робота (в правом конце заголовка) есть две стандартные кнопки: _ (скрыть с отображением в панели задач) и **X** (скрыть без отображения в панели задач).

В момент запуска Кумира окно наблюдения за Роботом скрыто. Чтобы сделать окно видимым, Школьник должен нажать кнопку «Показать окно Робота» на панели инструментов. Кроме того, окно Робота автоматически становится видимым при запуске на выполнение программы, содержащей строку `использовать Робот` (см. 3.3). Окно Робота становится видимым на том же месте, где оно находилось, когда его последний раз сделали скрытым.

При окончании сеанса Работы система запоминает положение окна наблюдения за Роботом на экране. При следующем запуске окно появится на том же самом месте.

3.2.2 Свойства окна наблюдения за Роботом

На окне наблюдения за Роботом нет ни меню, ни кнопок. Таким образом, в окне Робота есть только полоса заголовка и рабочее поле.

В левой части заголовка есть надпись «Робот», за которой следует имя файла, в котором хранится стартовая обстановка. Если такого файла нет, то вместо имени файла выводится слово «временная». Правила, определяющие привязку стартовой обстановки к определенному файлу, описаны ниже.

Размер окна наблюдения полностью определяется размерами стартовой обстановки. Пользователь не может менять размер окна с помощью стандартных средств, например, манипулируя мышью.

3.2.3 Что входит в описание обстановки Робота

Обстановка Робота представляет собой прямоугольное поле, окруженное забором и разбитое на клетки. Говоря более точно, обстановка описывается следующими величинами:

1. размеры обстановки — количество строк (1–10) и количество столбцов (1–16)
2. для каждой клетки:
   - наличие стен вокруг клетки
   - признак закрашенности
   - величина радиации (измеряется в условных единицах, может принимать любое вещественное значение от 0 до 100)
   - температура (измеряется в градусах Цельсия, может принимать любое вещественное значение от -273 до +233)

Примечание. Нижняя возможная температура — это (приблизительно) абсолютный ноль (0 градусов по шкале Кельвина). Верхняя температура — это температура, при которой горят книги (451 градус по Фаренгейту).

Система команд Робота позволяет ему определить значения всех этих характеристик клетки (см. ниже).

Кроме того, в клетке могут быть пометки, видимые наблюдателю, но не доступные «органам чувств» Робота:

- символы в левом верхнем и левом нижнем углах
- точка в правом нижнем углу

Частью описания обстановки является и положение Робота. Как и для чтения пометок, у Робота нет средств, чтобы определить свои координаты.

3.2.4 Изображение текущей обстановки в окне наблюдения

Изображение текущей обстановка всегда полностью помещается в рабочем поле окна наблюдения за Роботом (см. 3.1.1).

Фон рабочего поля — зеленый. Закрашенные клетки — серые. Между клетками — тонкие черные линии. Стены (в том числе — «забор» по периметру прямоугольника обстановки) изображаются толстыми желтыми линиями.

В клетке рабочего поля окна наблюдения Робот изображается ромбиком. Температура и радиация не показываются, они могут быть только измерены Роботом. Символы в клетках, наоборот, видны человеку, но Робот не умеет их считывать.

3.2.5 Когда и как меняется текущая обстановка Робота

Текущая обстановка изменяется при выполнении команд Робота, подаваемых из программы или с пульта (см. 3.5). Изменения текущей обстановки тут же отображаются в окне наблюдения за Роботом, если оно в этот момент видимо.

Выполнение команд Робота влияет на текущую обстановку следующим образом. Команды перемещения и закрашивания отображаются в текущей обстановке естественно.

Если команда перемещения выдает отказ, то текущая обстановка не изменяется, а на экране (если он виден) соответствующий угол Робота закрашивается красным. Красный цвет снимается:

- при выполнении следующей команды Роботу с пульта
- при выполнении команды использовать Робот
- при принудительном помещении Робота в стартовую обстановку (см. ниже)

Команды проверок и измерения радиации и температуры на текущую обстановку не влияют.

Кроме того, в двух случаях происходит принудительное помещение Робота в стартовую обстановку (текущая обстановка становится равной стартовой). Это происходит:

- при запуске Кумир-программы, которая использует Робот (окно наблюдения при этом становится видимым, даже если оно было скрыто)
- при изменении имени файла стартовой обстановки Робота (в этой ситуации невидимое окно остается невидимым)

3.3 Стартовая обстановка, ее изменение и связь с текущей обстановкой

3.3.1 Вводные сведения

Стартовая обстановка — это обстановка, в которую Робот будет помещен при запуске Кумир-программы, т. е. при выполнении команды использовать Робот (см. 3.1.3).

Со стартовой обстановкой связана специальная настройка Кумира — текстовая строка, в которой записано имя некоторого файла, содержащего описание обстановки Робота. Как правило, в качестве стартовой настройки используется обстановка, хранящаяся в этом файле. Исключения описаны в 3.3.2.

Увидеть имя файла стартовой обстановки (для краткости — *ФСО*) можно с помощью вкладки «Каталоги и файлы» окна настройки. Как можно менять имя ФСО описано в 3.3.3.

3.3.2 Как определяется стартовая обстановка

Как правило, стартовая обстановка — это обстановка, хранящаяся в ФСО.

Из этого правила есть два исключения. Одно из них связано с тем, что файл с указанным именем может не содержать корректного описания обстановки или вообще не существовать. Этот случай рассмотрен ниже в 3.3.3.

Другое исключение связано с возможностью упрощенного внесения временных изменений в стартовую обстановку. А именно, если в данный момент в Кумир-системе происходит редактирование файла обстановки, то текущее промежуточное значение этой редактируемой обстановки считается временной стартовой обстановкой. Говоря неформально, действия пользователя по подготовке обстановки имеют приоритет над содержимым строки с именем ФСО.

Во время редактирования в заголовке окна наблюдения за Роботом вместо имени файла написано «временная». Имя файла восстанавливается в окне Робота в момент окончания редактирования. Редактирование обстановок подробно описано в 3.4.

3.3.3 Изменение имени файла стартовой обстановки

Общие сведения

Школьник может изменить имя файла стартовой обстановки (ФСО) двумя способами:

- с помощью команды «Сменить стартовую обстановку» меню Робота
- с помощью редактора стартовых обстановок

Состояние окна наблюдения (видимо/скрыто) при этих действиях не меняется.

При изменении строки с именем ФСО, стартовая обстановка становится текущей (Робот помещается в новую стартовую обстановку), что отражается в окне наблюдения за Роботом.

Команда «Сменить стартовую обстановку»

Изменение строки с именем ФСО происходит с помощью стандартного диалога выбора файла. При этом в качестве каталога по умолчанию предлагается каталог текущего файла стартовой обстановки. Если выбранный файл существует и содержит корректную обстановку, то эта обстановка считается стартовой. Она же объявляется *текущей*, т. е. Робот помещается в эту обстановку. Это отображается в окне наблюдения Робота. Имя файла стартовой обстановки (без директории) показывается в левой части заголовка окна наблюдения за Роботом.

Если с чтением обстановки из выбранного файла возникают какие-либо проблемы, то стартовой (и одновременно текущей) объявляется стандартная обстановка, хранящаяся в Кумире. В левой части заголовка окна наблюдения за Роботом появляется предупреждающая надпись — «нет файла». Никакой файл с обстановкой при этом не создается, а предыдущее значение имени файла стартовой обстановки не изменяется.

Изменение стартовой обстановки с помощью редактора стартовых обстановок Робота

Редактор стартовых обстановок Робота — составная часть системы Кумир, его работа подробно описана в 3.4. Он позволяет добавлять и удалять стены, менять размеры обстановки и т. п.

В меню этого редактора есть команда «Сохранить как стартовую». По этой команде вызывается стандартный диалог сохранения. По умолчанию предлагается текущее имя файла стартовой обстановки. При корректном выполнении операции сохранения новое имя файла стартовой обстановки запоминается, а сохраненная обстановка объявляется стартовой. При этом новая стартовая обстановка одновременно становится текущей, т. е. показывается в окне наблюдения за Роботом.

Примечание. В заголовке окна наблюдения (до окончания редактирования стартовой обстановки) остается слово «временная», а не имя файла.

3.3.4 Начальная установка имени файла стартовой обстановки

Запуск системы Кумир

Система Кумир при окончании сеанса работы запоминает в своих настройках значение имени файла со стартовой обстановкой. При новом запуске система Кумир проверяет, существует ли файл с запомненным именем. Если файл существует и содержит корректную обстановку, то эта обстановка считается стартовой. Она же объявляется *текущей*, т. е.

Робот помещается в эту обстановку. Это отображается в окне наблюдения Робота. Имя файла стартовой обстановки (без директории) показывается в левой части заголовка окна наблюдения за Роботом.

Если с чтением обстановки из выбранного файла возникают какие-либо проблемы, то стартовой (и одновременно текущей) объявляется стандартная обстановка, хранящаяся в Кумире. В левой части заголовка окна наблюдения за Роботом появляется предупреждающая надпись — «нет файла». Никакой файл с обстановкой при этом не создается, а предыдущее значение имени файла стартовой обстановки не изменяется.

Установка имени файла стартовой обстановки при инсталляции системы Кумир

При инсталляции системы Кумир в качестве имени ФСО записывается полное имя файла со стандартной обстановкой, который входит в поставку Кумира (10x16.fil). Стандартной обстановкой является пустая обстановка максимально допустимого размера 10*16 с Роботом в левом верхнем углу. Слово «пустая» означает, что

- в обстановке нет стен (кроме проходящего по периметру забора)
- в клетках нет точек и символов
- во всех клетках радиация 0 и температура 0

3.4 Редактирование стартовой обстановки

3.4.1 Общие сведения

Редактор стартовых обстановок можно использовать:

- для подготовки новых обстановок
- при отладке Кумир-программ для оперативного внесения небольших изменений в стартовую обстановку

Запуск редактирования производится с помощью команды «Редактировать стартовую обстановку» меню «Инструменты». По этой команде появляется специальное окно — *окно редактирования стартовой обстановки* (для краткости — *окно редактирования*). Выход из состояния редактирования стартовой обстановки производится с помощью соответствующей кнопки на окне редактирования или по команде «Выход» меню «Обстановка» на этом окне.

Свойства окна редактирования аналогичны свойствам окна наблюдения за Роботом. Основных отличий – два:

- фон рабочего поля — синий
- в окне редактирования над рабочим полем есть стандартная полоса, содержащая главное меню окна и инструментальные кнопки

После включения режима редактирования строка «Редактировать стартовую обстановку» становится неактивной, т. е. редактировать одновременно две обстановки нельзя.

3.4.2 Главное меню окна редактирования

Общие сведения

В полосе меню окна редактирования стартовой обстановки имеется выпадающее меню «Обстановка» и команда «Помощь».

По команде «Помощь» появляется окно с текстом:

Редактирование обстановки
Поставить/убрать стену — щелкнуть по границе между клетками.
Закрасить/сделать чистой клетку — щелкнуть по клетке.
Поставить/убрать точку — щелкнуть по клетке при нажатой клавише Ctrl.
Установить температуру, радиацию, метки — щелкнуть по клетке правой кнопкой.
Переместить Робота — тащить мышью.
Изменить размеры обстановки — команда «Новая обстановка» меню «Обстановка».

Меню «Обстановка» содержит следующие команды:

- Новая обстановка
- Открыть
- Недавние обстановки
- Сохранить
- Сохранить как...
- Сохранить как стартовую
- Печать в файл
- Выход

Команды Меню «Обстановка»

Перечислим кратко особенности выполнения каждой команды.

Новая обстановка Открывается стандартная форма. Ввести число, превосходящее максимальный размер нельзя. Допустимые размеры: по высоте (строки) — от 1 до 10, по ширине (столбцы) – от 1 до 16.

Открыть Открывает для редактирования файл с обстановкой, для чего вызывается стандартная форма. О выборе директории по умолчанию — см. ниже 3.4.4.

Недавние обстановки Работает стандартно. В качестве недавних обстановок запоминаются все обстановки, к которым применялись операции сохранения и записи (см. 3.4.4).

Сохранить Работает стандартно. Если обстановка была создана командой «Новая обстановка», то спрашивает имя в стандартной форме. О выборе директории по умолчанию — см. 3.4.4.

Сохранить как... Работает так же, как «Сохранить», но с обязательным запросом имени файла сохранения. Эта команда не влияет на имя файла стартовой обстановки.

Сохранить как стартовую То же, что и «Сохранить как. . . », но с изменением имени файла стартовой обстановки.

Печать в файл Создание PDF-файла с изображением окна (см. 3.6.2).

Выход Синоним — **X** на заголовке окна. Завершает редактирование; если последняя обстановка не сохранена — переспрашивает.

3.4.3 Непосредственное редактирование обстановки

Основное редактирование обстановок Робота ведется в непосредственном режиме. Предусмотрены следующие операции непосредственного редактирования:

- поставить/убрать стену — щелкнуть по границе между клетками
- закрасить/сделать чистой клетку — щелкнуть по клетке
- поставить/убрать точку — щелкнуть по клетке при нажатой клавише Ctrl
- переместить Робота — тащить мышью
- установить температуру, радиацию, метки — щелкнуть по клетке правой кнопкой

В последнем случае появляется форма, с помощью которой можно установить нужные значения.

3.4.4 Операции с файлами обстановок

В заключение этого раздела перечислим все операции с файлами обстановок (расширение .fil), предусмотренные в системе Кумир. Соответствующие им команды входят в меню «Робот» главного окна (команды «Сменить стартовую обстановку», «Сохранить обстановку в файл») и в меню «Обстановка» окна редактирования стартовой обстановки (команды «Открыть», «Сохранить», «Сохранить как. . . », «Сохранить как стартовую»).

Система Кумир запоминает файлы, к которым применялись указанные операции. Список последних таких файлов доступен с помощью команды «Недавние файлы» меню «Обстановка». Во всех перечисленных ниже операциях с файлами обстановок в качестве директории по умолчанию предлагается директория, использовавшаяся в последней по времени выполнения операции с файлами обстановок.

3.5 Пульт Робота

3.5.1 Общие сведения

Пульт Робота располагается в отдельном окне и предназначен для управления Роботом. С помощью пульта Роботу можно передавать команды, Робот непосредственно выполняет эти команды.

Пользоваться пультом можно как при видимом окне наблюдения за Роботом, так и когда это окно скрыто (управление Роботом вслепую).

Пульт влияет только на поведение Робота внутри текущей обстановки и на отображение этого в окне наблюдения за Роботом. В частности, манипуляции с пультом Робота не влияют на свернутость/развернутость окна наблюдения за Роботом и обстановку в окне редактора стартовых обстановок.

Окно Пульта создается при запуске системы Кумир. Его свойства — такие же, как у окна наблюдения за Роботом и окна редактирования стартовой обстановки. В частности: при запуске системы Кумир пульт скрыт; пользователь не может менять размеры окна пульта. Чтобы сделать пульт видимым, нужно воспользоваться командой «Пульт Робота» меню «Робот» на главном окне системы Кумир или соответствующей этой строке инструментальной клавишей.

Связь Пульта с Роботом возможна всегда, кроме тех моментов, когда идет непрерывное выполнение программы, использующей Робот (т. е. программы, которая содержит строку использовать Робот). В частности, не имеет значения, идет редактирование стартовой обстановки или нет. О наличии связи свидетельствуют индикаторы в левом верхнем углу пульта.

По желанию пользователя, система Кумир может «перехватывать» команды, передаваемые с пульта, и вставлять их в текст Кумир-программы (см. 3.5.5).

3.5.2 Общий вид Пульта

На пульте Робота есть:

- лампочки-индикаторы:
  - зеленая лампочка-индикатор «Связь установлена»
  - красная лампочка-индикатор «Нет связи»
- табло для семи последних выполненных команд, разделенное на левую часть (собственно команда) и правую часть (реакция Робота)
- вспомогательные кнопки табло: «Сброс табло» (кнопка очистки, справа от табло); прокрутка табло (две кнопки сверху и снизу)
- кнопочная панель из 9 кнопок, служащая для набора команд Роботу

3.5.3 Кнопочная панель. Передача команд Роботу

На панели есть:

- 4 кнопки-стрелки (они соответствуют командам передвижения Робота)
- кнопка «закрасить» (в центре)
- кнопки «температура» и «радиация» (в левом верхнем и левом нижнем углах)
- две кнопки-префикса «Стена/Закрашено» и «Свободно/Чисто» (в правом верхнем и левом нижнем углах)

Для передачи команды действия (вверх, вниз, вправо, влево, закрасить, радиация, температура) достаточно нажать соответствующую кнопку.

Чтобы передать команду проверки нужно последовательно нажать две кнопки: сначала — префикс («Стена/Закрашено» и «Свободно/Чисто»), а потом — основную (закрасить или одну из стрелок).

Примеры:

1. Чтобы передать Роботу команду проверки «слева свободно?» нужно сначала нажать кнопку-префикс «Свободно/Чисто», а затем кнопку «влево».

2. Чтобы передать Роботу команду проверки «клетка закрашена?» нужно сначала нажать кнопку-префикс «Стена/Закрашено», а затем кнопку «закрасить».

После того, как нажата кнопка-префикс, она меняет цвет. Если после кнопки-префикса нажата кнопка, отличная от стрелок и кнопки «закрасить», то исходная кнопка-префикс просто будет забыта.

3.5.4 Использование табло

На табло выводятся все команды, передаваемые Роботу вместе с реакцией Робота на эти команды. В ответ на команды перемещения в правой части появляется надпись «OK» или «Отказ». Если с Роботом нет связи, на табло выдается «Нет связи».

Вывод на табло при первом включении Пульта в сеансе работы системы Кумир или после нажатия кнопки «Сброс табло» начинается с верхней строки. Когда табло заполнилось, начинается прокрутка и вывод идет в последнюю строку табло. Выше и ниже табло есть две кнопки для прокрутки его содержимого на одну строку вверх/вниз.

3.5.5 Перехват команд пульта системой Кумир

Для того, чтобы установить режим перехвата команд пульта Робота, предназначен пункт «Перехватывать команды пульта» меню «Редактирование» главного окна системы Кумир. Если такой режим установлен (это отображается пометкой в меню), то каждая передаваемая с пульта команда, отображается строкой в тексте Кумир-программы. Место вставки — после строки, в которой находится курсор; после вставки курсор переводится в конец вставленной строки.

При включенном режиме перехвата команд пульта возможно и обычное редактирование, в частности — изменение положения курсора с помощью мыши.

Для команд-действий (вверх, вниз, влево, вправо, закрасить) вставляется вызов соответствующего алгоритма-процедуры. Для других команд Робота, которым соответствуют алгоритмы-функции, вставляются команды вывода, например:

```
вывод "Радиация: ", радиация, нс
вывод "Сверху стена: ", сверху стена, нс
```

3.6 Робот и главное меню

3.6.1 Общие сведения

К Роботу относится меню «Робот» главного окна системы «Кумир» (полностью), а также две команды из других меню — «Редактировать стартовую обстановку» меню «Инструменты» (см. 3.4.1) и «Перехватывать команды пульта» меню «Редактирование» (см. 3.5.5). Меню Робот содержит следующие строки:

- Показать поле Робота
- Напечатать обстановку
- Сохранить обстановку в файл
- Сменить стартовую обстановку
- Вернуться в стартовую обстановку
- Пульт Робота

Для действий «Показать поле Робота» и «Пульт Робота» есть инструментальные кнопки на главной панели.

Ниже отдельно описана каждая из команд меню «Робот».

3.6.2 Команды меню «Робот»

- *Показать поле Робота* — делает видимым окно наблюдения за Роботом.
- *Напечатать обстановку* — создает файл в формате PDF, изображающий текущую обстановку в цветном или в черно-белом варианте.
- *Сохранить обстановку в файл* — создает текстовый файл с описанием обстановки во внутреннем формате .fil. Этот файл в дальнейшем может быть загружен в качестве стартовой обстановки (команда «Сменить стартовую обстановку» ниже) или при редактировании стартовой обстановки (команда «Открыть» окна редактирования стартовой обстановки). Использует стандартный диалог. О выборе директории по умолчанию — см. 3.4.4.
- *Сменить стартовую обстановку* — устанавливает новое имя файла стартовой обстановки (с помощью стандартного диалога) и загружает саму стартовую обстановку. О выборе директории по умолчанию — см. 3.4.4.
- *Вернуться в стартовую обстановку* — делает стартовую обстановку текущей.
- *Пульт Робота* — делает видимым пульт Робота.

3.7 Справочник. Система Кумир+Робот (заключение)

В этом разделе мы перечисляем все объекты и действия, которые относятся к исполнителю Робот.

3.7.1 Обстановки

Робот существует в определенной обстановке. Описания обстановок хранятся в текстовых файлах специального формата (формат .fil).

Обстановка, в которой находится Робот в данный момент (включая информацию о положении Робота), называется *текущей* обстановкой.

Кроме того, есть особая *стартовая* обстановка — обстановка, в которую принудительно помещается Робот в начале выполнения программы, использующей Робот.

Одна из настроек Робота содержит имя файла — файла стартовой обстановки. Как правило, стартовая обстановка — это обстановка, описанная в этом файле.

Из этого правила есть два исключения:

1. если файл с указанным именем не содержит корректного описания обстановки, то в качестве стартовой обстановки берется стандартная обстановка (см. 3.3.3)
2. если включен *режим редактирования стартовой обстановки*, то в качестве стартовой берется временная стартовая обстановка — та, которая в данный момент находится в окне редактирования.

3.7.2 Окна

С Роботом связаны три окна:

- окно наблюдения за Роботом (см. 3.2)
- окно редактирования стартовой обстановки (см. 3.4)
- окно пульта (см. 3.5)

Эти окна создаются в момент начала сеанса работы системы Кумир и могут использоваться одновременно и в любых сочетаниях. Каждое из окна независимо от остальных может быть видимо или скрыто.

Команды, которые делают видимым окно наблюдения за Роботом и пульт, находятся в меню «Робот». Для этих команд также есть инструментальные кнопки. Окно наблюдения за Роботом автоматически становится видимым в момент начала выполнения программы, использующей Робот.

Команда, которая делает видимым окно редактирования стартовой обстановки, находится в меню «Инструменты». Эта команда («Редактировать стартовую обстановку») одновременно включает режим редактирования стартовой обстановки. Выход из режима редактирования стартовой обстановки производится с помощью команды «Выход» меню «Обстановка» окна редактирования или с помощью кнопки этого окна. Во время режима редактирования строка «Редактировать стартовую обстановку» неактивна.

На окне наблюдения за Роботом нет собственного меню (это окно только для наблюдения). На окне редактирования есть меню (см. 3.4.2). Одна из команд этого меню («Сохранить как стартовую...») приводит к изменению текущей обстановки, что находит отражение в окне наблюдения за Роботом.

Команды, передаваемые с пульта, непосредственно влияют на текущую обстановку, а значит, и на окно наблюдения. Команды, передаваемые с пульта, могут отражаться в тексте редактируемой Кумир-программы. На окно редактирования стартовой обстановки команды пульта не влияют.

3.7.3 Правила изменения обстановок

Текущая обстановка изменяется:

- при выполнении Роботом команд (посылаемых из программы или с пульта)
- при принудительном помещении Робота в стартовую обстановку

Робот принудительное помешается в *стартовую обстановку*:

- в начале выполнения программы, содержащей строку использовать Робот
- при выполнении операции, приводящей к изменению имени файла стартовой обстановки

Имя файла стартовой обстановки изменяется при выполнении следующих команд:

- команды «Сменить стартовую обстановку» меню «Робот»
- команды «Сохранить как стартовую...» меню «Обстановка» окна редактора стартовых обстановок

Стартовая обстановка изменяется:

- при изменении имени файла стартовой обстановки (см. выше)
- в процессе редактирования стартовой обстановки («временная стартовая обстановка»: стартовой обстановкой является обстановка, находящаяся в окне редактирования стартовой обстановки)

Глава 4

Исполнитель Чертежник

4.1 Введение

4.1.1 Общие сведения

Исполнитель *Чертежник* предназначен для построения рисунков, чертежей, графиков и т. д. на бесконечном во все стороны листе, ниже этот лист называется *чертежным листом*. На чертежном листе задана прямоугольная система координат, единицу измерения в этой системе координат будем называть *единицей Чертежника* (сокращенно – е. ч.).

Чертежник имеет перо, которое может подниматься, опускаться и перемещаться при выполнении команд КУМИР-программы (см. 4.1.3). При перемещении опущенного пера за ним остается след — отрезок от старого положения пера до нового.

Пользователь может видеть ограниченную часть листа через прямоугольное *окно Чертежника*. Пользователь может задать форму окна («альбомная» или «книжная», см. ниже), какую часть листа показывать и в каком масштабе.

4.1.2 Состояния Чертежника

Поведение Чертежника описывается состоянием его пера:

- координатами (во внутренней системе координат чертежа)
- режимом (поднято — режим перемещения без рисования или опущено — режим перемещения с рисованием)
- цветом чернил

Школьник может работать с чертежным листом (очистить, сохранить, загрузить и т. п.) с помощью меню «Чертеж» на окне Чертежника, и управлять режимом работы самого окна с помощью меню «Вид» и кнопок на окне Чертежника.

4.1.3 Алгоритмы Чертежника

Исполнитель Чертежник может выполнять следующие шесть команд (для каждой команды указан алгоритм языка КуМир).

- **алг** поднять перо

  Переводит чертежника в режим перемещения без рисования.

- **алг** опустить перо

  Переводит чертежника в режим перемещения с рисованием.

  Замечание. 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.

- **алг** сместиться на вектор (**вещ** dX, **вещ** dY)

  Перемещает перо на dX вправо и dY вверх.

- **алг** сместиться в точку (**вещ** x, **вещ** y)

  Перемещает перо в точку с координатами (x,y).

- **алг** установить цвет (**лит** наименование цвета)

  Устанавливает цвет чернил. Допускается 9 цветов: ”черный”, ”белый”, ”красный”, ”оранжевый”, ”желтый”, ”зеленый”, ”голубой”, ”синий”, ”фиолетовый”. Подробнее об использовании цветов в Чертежнике см. 4.1.4.

- **алг** надпись (**вещ** ширина_знакоместа, **лит** текст)

  Каждый символ рисуется шрифтом `Courier New`. Позиция пера в момент начала рисования рассматривается как начальная точка базовой линии рисования. После окончания рисования перо должно оказаться на базовой линии рисования в правом конце *промежутка, отделяющего* нарисованный символ *от следующего символа*. При этом перо находится в том же режиме, что и перед выполнением команды.

  Значение `ширина_знакоместа` задается в единицах чертежника. В соответствии со стандартом шрифта `Courier New`, параметр `ширина_знакоместа` полностью определяет все размеры всех символов, а также ширину расстояния между символами в слове.

4.1.4 Как Чертежник работает с цветами

Общие принципы работы с цветом

В каждый момент работы Чертежника определен используемый *цвет чернил*. Чертежник рисует линии именно этим цветом. При загрузке Кумира устанавливается черный цвет. Пользователь может установить цвет специальной командой. Узнать из программы цвет чернил (как и другие элементы состояния пера Чертежника, см. 4.1.2) из программы нельзя. При рисовании на одном и том же месте разными цветами будет виден последний цвет.

Какие цвета использует Чертежник

Чертежник умеeет рисовать девятью цветами. Эти цвета: ”черный”, ”белый”, ”красный”, ”оранжевый”, ”желтый”, ”зеленый”, ”голубой”, ”синий”, ”фиолетовый”. Никаких других цветов, кроме девяти указанных, в Чертежнике нет, никаких смешиваний цветов сделать нельзя.

Интерфейс изменения цвета в Чертежнике

Изменение цвета производится командой установить цвет (см. выше 4.1.3). У каждого цвета ровно одно допустимое наименование, записываемое строчными русскими буквами: ”черный”, ”белый”, ”красный”, ”оранжевый”, ”желтый”, ”зеленый”, ”голубой”, ”синий”, ”фиолетовый”. Цвета пишутся в кавычках — как литерные константы. Использование заглавных букв не допускается (как и при написании ключевых слов).

Школьник может задать аргумент команды установить цвет не только константой, но и произвольным текстовым выражением. При этом контроль допустимости цвета происходит на этапе выполнения.

4.1.5 Кумир-программы, использующие Чертежник

В начале КУМИР-программы должна быть строка использовать Чертежник.

При выполнении команды использовать Чертежник чертежный лист приводится к *начальному состоянию* — пустому листу. Кроме того, окно Чертежника становится видимым, в нем изображается фрагмент пустого листа, соответствующий текущим настройкам окна. Если нужно, изображаются оси координат и сетка. Если пользователь желает вернуться к настройкам окна по умолчанию, он может сделать это вручную с помощью пункта «Восстановить» меню «Вид».

4.2 Окно Чертежника

4.2.1 Общие сведения

Окно исполнителя чертежник в правом углу поля имеет две стандартные кнопки имеет _ и **X** со стандартной семантикой.

Окно создается в момент старта Кумира, но является невидимым. Окно становится видимым в следующих случаях:

- Пользователь нажал на кнопку «Показать окно чертежника» на панели инструментов или воспользовался соответствующим пунктом в подменю «Чертежник»
- При выполнении строки программы использовать Чертежник

Окно имеет стандартные размеры, зависящие от размеров экрана (см. ниже). Окно может иметь одну из двух стандартных форм, они называются ниже ГОР и ВЕРТ. ГОР — «альбомная форма», ВЕРТ — «книжная». Пользователь может менять форму окна с помощью меню «Чертеж» окна Чертежника (см. ниже). Размеры окна для каждой формы однозначно привязаны к размерам экрана. С помощью стандартных оконных средств (например, мыши) их менять нельзя.

4.2.2 Структура окна

Большую часть окна занимает рабочее поле, в котором изображается выбранный фрагмент листа рисования.

Над рабочим полем расположено раскрывающееся меню «Чертеж» (слева), правее него расположено раскрывающееся меню «Вид» и четыре кнопки, дублирующие некоторые функции меню «Вид» (см. ниже). Меню «Чертеж» работает с содержимым листа рисования; меню «Вид» определяет режим просмотра выбранного фрагмента листа.

При запуске системы Кумир в качестве участка просмотра устанавливается прямоугольник в горизонтальном положении шириной 7 единиц (от -3.5 до 3.5). Высота участка

просмотра в единицах чертежника однозначно определяется указанной шириной и уже известными пропорциями рабочего поля (см. выше). Если перевести окно из горизонтального формата в вертикальный или наоборот, то положение центра окна на чертеже не изменится, а ширина станет равной около 3 единиц.

4.2.3 Просмотр листа рисования

В окне Чертежника виден прямоугольный участок листа рисования. Его размеры в пикселях зависят от размера экрана и выбранной формы окна (ГОР или ВЕРТ). Установка видимого фрагмента чертежа при запуске системы Кумир описана выше.

В дальнейшем размер участка просмотра и его положение на листе устанавливаются с помощью команд меню «Вид» на окне Чертежника, кнопок на окне и манипуляций мышкой.

4.2.4 Сетка

Часть чертежа, попавшая в область просмотра, может быть разлинована сеткой, линии сетки — серые. Рисовать ли сетку, и с какими параметрами пользователь может указывать (в произвольный момент работы Чертежника) с помощью команды «Сетка» меню «Вид». При этом сетка всегда проходит через начало координат, а пользователь задает шаг сетки (при желании – различный по горизонтали и вертикали).

Подчеркнем, что параметры сетки задаются в условных единицах чертежника, т.е. сетка как бы «вморожена» в лист чертежа. Чертеж мысленно разлиновывается сеткой с указанными параметрами, а в окне показывается указанная часть чертежа (вместе с сеткой) и в указанном масштабе.

Примечание. Если сетка становится слишком мелкой, то чертеж «замазывается», а процесс рисования — тормозится. Такое может произойти:

- при изменении параметров сетки (пункт «Сетка» меню «Вид»)
- при изменении масштаба изображения (меню «Вид», колесико мыши)

Чтобы избежать описанных выше явлений, Кумир отслеживает последствия запрашиваемых пользователем «опасных» действий. Если сетка становится слишком мелкой (расстояние между линиями менее трех пикселей), то сетка не рисуется и кнопка «Сетка» на панели окна чертежника (крайняя справа) становится красной.

4.2.5 Оси координат

Оси координат (если они попадают участок просмотра), показываются синими линиями. Оси координат всегда проходят через точку $(0, 0)$.

4.2.6 Показ координат точки

При нажатии правой кнопкой мыши на поле чертежа, на поле показываются координаты точки. Если отпустить кнопку, координаты исчезнут.

4.2.7 Меню «Чертеж»

Меню содержит следующие пункты:

Очистить	Стирает с листа все нарисованные линии и надписи
Сохранить	Сохранить чертеж в .ps файл
Загрузить	Загрузить чертеж из .ps файла (поддерживаются только файлы, сохраненные чертежником)
Печать в файл	Печатает в PDF-файл область просмотра чертежа (то, что видно в рабочем окне)

4.2.8 Меню «Вид»

Общие сведения

Меню «Вид» управляет показом чертежа в рабочем поле окна Чертежника. Оно содержит такие пункты:

- Крупнее
- Мельче
- Весь чертеж
- Восстановить
- Сдвиги
- Сетка
- Информация

Управление масштабом

Управление масштабом показа выполняется с помощью пунктов *Крупнее, Мельче, Весь чертеж, Восстановить*. Кроме того, масштаб изображения можно менять мышью (см. 4.2.10).

Команды *Крупнее* и *Мельче* изменяют масштаб изображения (размер участка просмотра) в $\sqrt{2} \approx 1.4$ раз. Центр участка просмотра на листе рисования при этом не меняется.

Команда *Весь чертеж* выбирает в качестве область показа минимальную прямоугольную область нужной формы, в которой полностью помещается весь нарисованный чертеж. Масштаб изображения выбирается так, чтобы весь этот прямоугольник помещался в рабочем поле (с учетом небольших отступов со всех сторон).

Команда *Восстановить* восстанавливает параметры области просмотра, принятые по умолчанию (см. 4.2.4).

Управление сеткой

По команде *Сетка* возникает форма с двумя кнопками («Показывать сетку» и «Сетка квадратная») и двумя редактируемыми полями («Шаг X» и «Шаг Y»). При этом, если выбрано «Сетка квадратная», то изменение шага по одному из направлений сетки синхронно меняет шаг по другому направлению.

Выдача информации

По команде *Информация* выводится табличка с информацией об области показа и состоянии пера.

Сдвиги

По команде *Сдвиги* выпадает меню четырех направлений сдвигов. Кроме того, в этом меню есть текст, объясняющий, как двигать чертеж мышью и рекомендующий так и поступать.

4.2.9 Инструментальные кнопки

Справа от меню «Вид» расположены 4 кнопки, соответствующие командам *Крупнее, Мельче, Полный чертеж, Сетка* меню «Вид».

4.2.10 Использование мыши

Для управления масштабом изображения можно использовать колесико мыши, а для управлением положением видимой области на листе — «таскать лист» в окне мышью при нажатой левой кнопке. Интерфейс этого — стандартный.

Глава 5

Система Кумир

5.1 Введение

5.1.1 Общие сведения

Система Кумир — позволяет создавать, отлаживать и выполнять программы на универсальном языке программирования Кумир.

Кумир — учебная система. Она сводит к минимуму «накладные расходы» на освоение, имеет развитую систему диагностики ошибок, средства, позволяющие ученику следить за выполнением программы и т. п. Ученик, никогда ранее не программировавший, может начать писать и выполнять алгоритмически относительно сложные программы через 1–2 часа после первого знакомства с Кумиром. В то же время система Кумир позволяет создавать достаточно большие и сложные программы (сотни строк).

Во время редактирования программы система Кумир после каждого перевода курсора на новую строку автоматически производит синтаксический разбор и сообщает о найденных ошибках.

Система Кумир включает графические исполнители Робот и Чертежник, которыми можно управлять из программы (а Роботом кроме того можно управлять вручную).

5.1.2 Основная программа

В каждый момент пользователь работает только с *одной Кумир-программой*, которую мы называем *основной* (в данный момент) программой системы Кумир. Текст основной программы представлен в *рабочем окне* (редакторе) системы Кумир (см. 5.2). Пользователь может прочитать текст основной программы из файла и поместить этот текст в редактор, редактировать текст и сохранить подготовленный текст в файл. Именно к основной программе относятся команды системы Кумир на выполнение.

Во время сеанса работы, кроме текста основной программы пользователю могут потребоваться и другие связанные с ней объекты — вспомогательные программы, написанные на Кумире (т. н. внешние исполнители), файлы с входными данными для основной программы и результатами ее работы, различные описания и т. п. Система Кумир имеет средства, облегчающая работу с этими вспомогательными объектами.

5.1.3 Язык Кумир. Исполнители

Язык Кумир — универсальный язык программирования, его прототипом послужил «школьный язык программирования» разработанный А. П. Ершовым в первой половине 80-х годов XX века. В дополнение к обычным для универсальных языков программирования

возможностям, Кумир имеет средства управления *исполнителями*. Говоря неформально, исполнитель – это устройство, которое может выполнять определенный набор действий. Действие может совершаться над внешними для исполнителя данными (*параметрами действия*) и/или над присущими исполнителю внутренними для него данными (*обстановкой исполнителя*).

Примером исполнителя может служить Робот. Его обстановка — это прямоугольник, разделенный на квадратные клетки. Размер прямоугольника может варьироваться. Каждой клетке приписаны числовые характеристики — «температура» и «радиация». Робот может «измерять» значения этих величин, а также передвигаться по клеткам по горизонтали и вертикали (действия влево, вправо, вперед, назад). Границы между некоторыми клетками непроходимы для Робота (там стоят «стены»), Робот умеет выполнять проверки слева стена, справа стена и т. п. При попытке «пройти через стену» в любом направлении возникает ситуация *отказ*.

Робот — пример встроенного исполнителя. Пользователь также может описать свои исполнители на языке Кумир.

5.1.4 Сеанс работы системы Кумир. Состояния системы

Работа пользователя в системе Кумир состоит в:

- подготовке программы к выполнению (редактирование, загрузка/сохранение программы, настройка параметров системы и т. п.);
- выполнении программы (в обычном или отладочном режиме);
- просмотре (анализе) результатов работы программы (окончательных или промежуточных).

В зависимости от выполняемого действия, система Кумир находится в одном из четырех возможных состояний:

- РЕДАКТИРОВАНИЕ
- ВЫПОЛНЕНИЕ
- АНАЛИЗ РЕЗУЛЬТАТОВ (или просто АНАЛИЗ)
- ПАУЗА

Состояние системы накладывает естественные ограничения на возможность выполнения различных действий. Например, во время выполнения программы нельзя изменять ее текст.

Смысл двух первых состояний ясен из их названия. В состояние АНАЛИЗ система переходит после окончания выполнения программы (нормального или аварийного). В этом состоянии пользователю доступны все рабочие сообщения программы — для просмотра и анализа. Любое действие по изменению текста программы сбрасывает эти рабочие сообщения и переводит систему в состояние РЕДАКТИРОВАНИЕ. В состояние ПАУЗА система переходит в случае остановки во время выполнения (при вызове встроенной функции **пауза** или после очередного шага при выполнении программы «по шагам», см. 5.4.5). Схема возможных переходов между состояниями выглядит так:

- РЕДАКТИРОВАНИЕ → ВЫПОЛНЕНИЕ
- ВЫПОЛНЕНИЕ → {АНАЛИЗ, ПАУЗА}

- ПАУЗА → {ВЫПОЛНЕНИЕ, АНАЛИЗ}
- АНАЛИЗ → {ВЫПОЛНЕНИЕ, РЕДАКТИРОВАНИЕ}

5.1.5 Запуск системы Кумир

Запуск системы Кумир может быть выполнен стандартными средствами операционной системы:

- командной строкой;
- щелчком по пиктограмме системы Кумир;
- щелчком по пиктограмме файла с Кумир-программой (файл с расширением .kum).

Дополнительные ключи запуска

- **-z**
  - Формат вызова: kumir -z *путь*

    Сохранять настройки системы Кумир не в реестре, а в INI-файле. Файл *kumir.ini* должен располагаться в директории *путь*. Если в указанной директории этого файла нет, то он создается, а настройки системы принимаются по умолчанию.
  - Формат вызова: kumir -z *имя_файла*

    Сохранять настройки системы Кумир не в реестре, а в INI-файле *имя_файла*. Если этого файла не существует, то он создается, а настройки системы принимаются по умолчанию.

5.2 Рабочее окно и другие окна системы Кумир

5.2.1 Главное окно системы Кумир

После запуска Кумира на экране появляется *главное окно системы Кумир*. Главное окно открывается при начале сеанса работы системы Кумир и закрывается в момент окончания сеанса работы. Иногда мы будем называть это окно *рабочим*, т. к. основная работа с системой происходит именно в этом окне.

Главное окно системы Кумир имеет стандартный вид. Вверху окна расположены заголовок окна, главное меню и панель инструментов; снизу — строка состояния. Заголовок окна содержит полное имя файла, из которого была загружена основная программа. Строка состояния используется для вывода сообщений, показа положения курсора, состояния системы и т. п.

При работе с окном доступны стандартные возможности управления окнами: окно можно свернуть-развернуть, сжать-растянуть, передвинуть и т. п. При закрытии окна его параметры (например, размеры и положение) запоминаются; при следующем запуске окно открывается с теми же параметрами.

5.2.2 Структура главного окна

Собственно окно разбито на две *основные области*: *рабочую* область (вверху) и область *ввода-вывода* (внизу).

В рабочей области располагается основная программа — программа, с которой в данный момент работает система Кумир. При этом рабочая область делится на две части:

область *программы* (слева) и область *построчных сообщений* (справа). Область построчных сообщений аналогична *«полям»* в ученических тетрадях. В эту область при подготовке программы выводятся сообщения об ошибках, найденных в каждой строке, а при выполнении — сведения о значениях переменных, присваиваемых в строке.

Строки в области программы и области построчных сообщений синхронизированы: при прокрутке текста в области программы синхронно прокручиваются и сообщения. Относительные размеры (по горизонтали) области программы и «полей», а также размер области ввода-вывода по вертикали могут меняться пользователем.

Как по горизонтали, так и по вертикали виртуальный размер всех текстов не ограничены, поддерживается прокрутка.

5.2.3 Меню главного окна

Меню главного окна (главное меню системы Кумир) содержит восемь пунктов:

- Программа
- Редактирование
- Вставка
- Выполнение
- Инструменты
- Робот
- Чертежник
- Инфо

Каждому из этих пунктов соответствует свое раскрывающееся меню. Как правило, действие, для которого есть команда в главном меню, может быть выполнено также с помощью аккорда клавиш и/или инструментальной кнопки. В последнем случае нужный аккорд или пиктограмма приводятся рядом со строкой раскрывающегося меню.

Меню «Программа», «Редактирование» и «Вставка» описаны в 5.3; меню «Выполнение» — в 5.4; меню «Инструменты» — в 5.6. Отметим, что меню «Инструменты» не обязательно при первом знакомстве с системой Кумир.

Меню «Робот» и «Чертежник» посвящены встроенным графическим исполнителям системы Кумир, они описаны в отдельных руководствах.

Меню «Инфо» содержит сведения о версии системы Кумир, а также обеспечивает доступ к руководствам:

- по алгоритмическому языку Кумир;
- по использованию системы Кумир (настоящее руководство);
- по работе с исполнителем Робот;
- по работе с исполнителем Чертежник.

Все эти руководства подготовлены в виде pdf-файлов и могут просматриваться стандартными средствами как в ходе сеанса работы системы Кумир, так и автономно.

5.2.4 Инструментальные кнопки

Инструментальные кнопки (за исключением двух) расположены на панели инструментов, которая расположена под главным меню. Эта панель разбита на области, разделенные двойным пунктиром; порядок областей соответствует пунктам главного меню. Каждая кнопка снабжена всплывающей подсказкой, текст подсказки совпадает с текстом в раскрывающемся меню.

5.2.5 Область ввода-вывода

Область снизу главного окна системы Кумир предназначена для отображения вывода программы и для ввода данных в программу.

После каждого выполнения программы в область ввода-вывода добавляется горизонтальная линия. Таким образом, выводы отдельных запусков программ всегда разделены между собой.

Каждый вывод программ содержит в своем начале и конце специальные строки. Подробнее об этом см. 5.4.6.

Слева снизу области ввода-вывода находятся две кнопки управления ею: «Сохранить область вывода» и «Очистить область вывода». Они также дублируются в командах главного меню «Инструменты». Кнопка очистки просто очищает область вывода, а при нажатии на кнопку сохранения появляется подменю:

- Сохранить все
- Сохранить только последний вывод
- Сохранить выделенный текст

Назначение этих пунктов интуитивно понятно. Вывод сохраняется в текстовый файл.

5.3 Подготовка программы. Меню «Программа», «Редактирование», «Вставка»

5.3.1 Меню «Программа». Файлы, хранящие Кумир-программы

Меню «Программа» обеспечивает работу с файлами, в которых хранятся Кумир-программы, эти файлы имеют расширение .kum. Меню содержит следующие пункты:

- Новая программа
- Недавние программы
- Открыть программу
- Сохранить программу
- Сохранить программу как...
- Передать программу...
- Печать программы
- Выход

Эти пункты имеют стандартный для современных оконных систем смысл. Для пункта «Сохранить» есть инструментальная кнопка.

Пункт «Передать программу. . . » предназначен для передачи текста программы из редактора Кумира в какой-либо дополнительный модуль (например, в конвертор).

5.3.2 Редактирование программы

Редактор системы Кумир обеспечивает стандартные средства редактирования текстов: ввод символов в режиме вставки или замены, удаление символов, выделение / копирование / вставку / удаление фрагмента текста, «откатку» (отмену последних действий) и «накатку» (отмену откатки), поиск по тексту и т. п. Эти действия можно выполнять как в непосредственном режиме, так и с помощью меню «Редактирование». Кроме того, редактор программ системы Кумир предоставляет пользователю дополнительные возможности, ориентированные на специфику языка Кумир.

Редактор программ всегда использует шрифт, в котором все символы имеют одинаковую ширину. Какой именно это будет шрифт и его размер, пользователь может установить на вкладке «Внешний вид» диалогового окна «Настройки». Подробнее об этом см. 5.6.3

5.3.3 Меню «Редактирование»

Меню «Редактирование» содержит следующие строки:

- Отменить
- Отменить отмену
- Вырезать
- Копировать
- Вставить
- Найти и заменить
- Закомментировать
- Раскомментировать
- Перехватывать команды Пульта
- Макросы

Первые шесть строк имеют стандартный смысл и могут быть выполнены с помощью стандартных аккордов, для них (кроме «Найти и заменить») предусмотрены инструментальные кнопки.

Команда «Закомментировать» добавляет знак комментария | в начало каждой выделенной (хотя бы частично) строки. Команда «Раскомментировать» удаляет знак комментария из начала каждой выделенной строки. Если в начале выделенной строки не было знака комментария, то содержимое этой строки не меняется. Для команд «Закомментировать» и «Раскомментировать» предусмотрены инструментальные клавиши.

Команда «Перехватывать команды Пульта» связана с работой встроенного исполнителя Робот и позволяет вставлять в программу команды этого исполнителя. Подробнее — см. руководство по исполнителю Робот.

Строка «Макросы» описана ниже.

5.3.4 Макросы

Строка «Макросы» меню «Редактирование» позволяет запомнить нужную пользователю последовательность действий (*макрос*), а затем выполнить всю эту последовательность, нажав клавишу Esc, а потом — еще одну (выбранную пользователем) клавишу. Каждому макросу, таким образом, соответствуют:

- выполняемая последовательность действий редактирования;
- клавиша, используемая при вызове этой последовательности.

Кроме того, макросу соответствует условное *имя*, используемое при редактировании набора макросов. Определенные таким образом макросы сохраняются от сеанса к сеансу работы системы Кумир, т. е. пользователь может создать свою собственную библиотеку макросов.

Строка «Макросы» приводит к появлению вспомогательного меню, включающего 3 пункта:

- Начать запись макроса
- Завершить запись макроса
- Список макросов

После этих пунктов перечислены определенные в данный момент макросы. Для вызова макроса (вместо использования клавиши Esc и клавиши, определенной пользователем) достаточно щелкнуть левой кнопкой мыши по строке этого макроса.

Из двух первых пунктов вспомогательного меню «Макросы» в каждый момент активен лишь один. Обычно, активен пункт «Начать запись макроса», а после того, как запись макроса начата, активным становится пункт «Завершить запись макроса». После того, как начата запись макроса, система Кумир запоминает все действия пользователя по редактированию программы (одновременно эти действия выполняются). По сигналу об окончании записи макроса, появляется диалоговое окно, предлагающее определить для нового макроса имя и клавишу вызова.

Пункт «Список макросов» приводит к появлению вспомогательного окна, содержащего список определенных пользователем макросов. С помощью этого окна пользователь может удалить любой макрос или изменить его имя и клавишу.

Примечание. По техническим причинам в окне определения макросов клавиши обозначаются заглавными латинскими буквами. При вызове макроса раскладка клавиатуры (рус/лат, большие/маленькие буквы) не важна.

5.3.5 Меню «Вставка»

В этом меню описаны встроенные в систему макросы, обеспечивающие ввод в текст программы стандартных конструкций. Эти макросы не могут быть удалены или отредактированы или отредактированы пользователем. Меню «Вставка» позволяют вводить в текст программы следующие конструкции, для каждой конструкции указана соответствующая клавиша.

5.3.6 Возможности редактора, ориентированные на язык Кумир

Редактор системы Кумир ориентирован именно на работу с программами на языке Кумир. Это проявляется различными способами, которые представлены ниже.

Временное переключение раскладки клавиатуры

В редакторе Кумир есть возможность ввода символов из другой (русской или латинской раскладки), не переключая ее. Для этого необходимо вводить символы с нажатой клавишей "Alt".

Например, при активной русской раскладке клавиатуры, нажимая последовательно клавиши: ц, е, л, пробел, Alt+ш, получим текст: **цел** і.

Кроме того, для некоторых часто используемых последовательностей, зарезервированы сочетания клавиш (в любой раскладке клавиатуры):

- Alt+= – вводит последовательность символов присваивания :=
- Alt+1 – вводит символ комментария |

Синтаксический разбор и сообщения об ошибках

Всякий раз, когда курсор переходит на новую строку, происходит синтаксический разбор программы (всей программы, а не только этой строки). При этом отмечаются ошибки (если они есть), а также происходит разметка текста программы, облегчающая работу с ним:

- различные компоненты программы выделяются различными цветами и видами шрифта;
- в программе расставляются отступы так, чтобы части одной конструкции (**нач-кон**, **нц-кц** и т. п.) оказались друг под другом.

Сообщения об ошибках выводятся в область построчных сообщений (на «поля»), каждое сообщение — напротив той строки, в которой обнаружена ошибка. Место ошибки подчеркивается красным. Если в строке найдено более одной ошибки, то диагностируется только одна (первая слева).

Отступы

Отступы изображаются точками в левой части строки, при синтаксическом разборе эти точки игнорируются. Размер отступа (количество позиций на один отступ) можно установить с помощью меню «Настройки» → «Внешний вид».

Пользователь может поместить курсор в область отступов с помощью мыши или стрелок и передвигать его с помощью стрелок. Однако, для изменений эти области заблокированы. Если нажать какую-нибудь клавишу, отличную от стрелки, то произойдет следующее. Если это символьная клавиша, то курсор будет передвинут в первую «значащую» позицию строки и там будет выведен указанный символ. Клавиша Delete, не меняя текста, переведет курсор в первую значащую позицию строки. Клавиша BackSpace, не меняя текста, переведет курсор в конец предыдущей строки.

Выделение цветом и шрифтом

Различные компоненты текста программы выделяются цветом и видом шрифта (жирность, курсив). К таким компонентам относятся:

- Типы величин
- Значения (числовые и логические)
- Строковые значения

- Имена исполнителей
- Имена алгоритмов
- Строки описаний алгоритмов (начинающиеся со знака #)
- Комментарии

Рекомендуется также выводить курсивом латинские буквы в именах, чтобы не путать их с близкими по начертанию русскими буквами.

Пользователь сам может настроить выбор цветов и свойств шрифта, используя панель «Настройки» → «Подсветка кода». Для настройки цвета нужно щелкнуть правой кнопкой по цветовому квадратику.

Угадывание имен алгоритмов

Система Кумир облегчает пользователю ввод длинных имен алгоритмов. Если набрать несколько первых букв имени алгоритма, а потом аккорд Ctrl-пробел (кроме ОС Mac) или Tab (кроме ОС MS Windows), то:

- если эти буквы однозначно определяют имя алгоритма, то введенное начало будет расширено до этого имени;
- если есть несколько вариантов, то пользователю будет предложено выбрать из них;
- если алгоритма с указанным началом нет, ничего не произойдет.

5.4 Выполнение программ. Меню «Выполнение»

5.4.1 Основные сведения

В этом разделе мы напоминаем необходимые сведения о языке Кумир. Подробнее — см. описание языка Кумир.

Выполнение программы на языке Кумир состоит в том, что последовательно выполняются:

- инструкции использовать... (если они есть, см. 5.5.2);
- вступление к программе (если оно есть);
- основной алгоритм (если он есть).

Выполнение программы происходит по шагам. Один шаг выполнения состоит в:

- выполнении *простой* команды или
- проверке условия в составной команде (например, **если**...) или
- выполнении вспомогательного действия в составной команде (например, обработке ключевого слова **кц**).

При развернутой записи программы каждый шаг выполнения соответствует одной строке программы. Однако, возможна и более компактная запись, при которой одна строка соответствует нескольким шагам выполнения.

Особую роль играют вызовы вспомогательных алгоритмов, представленных в текущей программе. По желанию пользователя, такой вызов может трактоваться как один шаг (крупный «ШАГ»). В то же время, можно и «войти внутрь вызова». Тогда очередной шаг (мелкий «шаг») будет состоять только в выполнении строки-заголовка, а выполнение всего вызова будет рассматриваться как цепочка мелких шагов.

Выполнение программы завершается *нормально*, если выполнена строка **кон** основного алгоритма или выполнены все строки вступления (если основного алгоритма нет). Кроме того, выполнение может завершиться аварийно — если произошла ошибка при выполнении одной из команд или пользователь остановил выполнение командой «Прервать» (см. ниже). В случае аварийного завершения работы программы в поле ввода-вывода выводится сообщение о причинах аварии.

5.4.2 Выполнение программы и состояния системы Кумир

Напомним, что система Кумир может находиться в таких состояниях:

- РЕДАКТИРОВАНИЕ: происходит подготовка программы, выполнения нет.
- ВЫПОЛНЕНИЕ: происходит выполнение программы, редактирование текста программы запрещено.
- ПАУЗА: выполнение программы приостановлено, но может быть продолжено; редактирование текста программы запрещено.
- АНАЛИЗ: выполнение завершено, однако все сообщения программы доступны для наблюдения и анализа; по любому действию в области программы рабочего окна, система переходит в состояние РЕДАКТИРОВАНИЕ.

Переход в состояние ВЫПОЛНЕНИЕ возможен из состояний:

- РЕДАКТИРОВАНИЕ, АНАЛИЗ (с помощью команд «Выполнить непрерывно», «Непрерывно без показа на полях», «ШАГ», «шаг»);
- ПАУЗА (с помощью команд «Выполнить непрерывно», «Непрерывно без показа на полях», «ШАГ», «шаг», «До конца алгоритма»).

Из состояния ВЫПОЛНЕНИЯ система Кумир может перейти в состояния:

- АНАЛИЗ, если выполнение программы завершено (нормально или аварийно);
- ПАУЗА (при выполнении команды **ввод** или встроенных функций **пауза** и **клав** (см. описание языка Кумир), или при выполнении по шагам, см. ниже).

5.4.3 Вывод значений на поля

Как правило, при выполнении Кумир-программы, на поля выводятся значения, присваиваемые величинам и результаты проверок. Вывод значений на поля производится при выполнении следующих команд:

- команда присваивания (пример сообщения: н=5)
- команда ввод (пример сообщения: н=5)

- заголовок цикла «для» (пример сообщения: н=5)
- заголовок цикла «раз» (пример сообщения: 5 раз)
- команды контроля выполнения (**утв**, **дано**, **надо**) (примеры сообщений: да, нет)
- конструкции проверки условий (**пока**, **если**, **при**, **кц_при**) (примеры сообщений: да, нет)

Если в одной строке записано несколько команд, то на поля выводится несколько сообщений, разделенных точкой с запятой.

5.4.4 Выполнение первого алгоритма с параметрами

Первый алгоритм может иметь параметры, типами который являются простые величины (**цел**, **вещ**, **лог**, **сим**, **лит**).

Перед выполнением первого алгоритма, система КуМир запрашивает у пользователя значения всех **арг** и **аргрез** параметров, а после его выполнения - выводит значения **рез** и **аргрез** параметров.

Если алгоритм является функцией (то есть возвращает некоторое значение), то после его выполнения также выводится возвращаемое значение.

Пример.

```
алг лит алгоритм(вещ a, рез цел p)
нач
· p := int(a)
· знач := вещ_в_лит(a)
кон
```

Выполнению этого алгоритма соответствует программа:

```
алг
нач
· вещ a
· цел p
· цел результат_алгоритм
· вывод "Введите a: "
· ввод a
· результат_алгоритм := алгоритм(a, p)
· вывод "Значение функции = ", результат_алгоритм, нс
· вывод "p = ", p, нс
кон

алг лит алгоритм(вещ a, рез цел p)
нач
· p := int(a)
· знач := вещ_в_лит(a)
кон
```

5.4.5 Меню «Выполнение»

Действия, связанные с выполнением основной программы, выполняются с помощью меню «Выполнение». В этом меню шесть пунктов. Для всех этих пунктов предусмотрены аккорды и инструментальные кнопки.

Выполнить непрерывно

Начинает (при состоянии системы РЕДАКТИРОВАНИЕ или АНАЛИЗ) или продолжает (при состоянии системы ПАУЗА) выполнение программы. Программа выполняется «непрерывно», т. е. без остановок между шагами. Выполнение программы может быть завершено (нормально, аварийно или по команде «Прервать») или приостановлено, если в ходе выполнения будет выполняться команда **ввод**, либо одна из встроенных функций **пауза** или **клав**.

Во время выполнения на поля выводятся вычисляемые значения величин и условий (см. 5.4.3).

Непрерывно без показа на полях

Аналогично «Выполнить непрерывно» — но без вывода на поля вычисляемых значений величин и условий.

ШАГ

Выполняет один ШАГ программы и переходит в режим ПАУЗА. Допускается использование в состояниях РЕДАКТИРОВАНИЕ, АНАЛИЗ, ПАУЗА. При запуске в состоянии РЕДАКТИРОВАНИЕ и АНАЛИЗ «проскакивает» строки **использовать**... и останавливается перед выполнением первой строки вступления основной программы (если оно есть) или перед выполнение **алг**-строки основного алгоритма. Выполнение команды вызова алгоритма-процедуры трактует как один ШАГ.

шаг

Аналогично команде «ШАГ». Отличие состоит в трактовке команды алгоритма-процедуры и вычислении значении алгоритма-функции (если они представлены в текущей программе). В этих случаях очередным шагом будет выполнение строки-заголовка вспомогательного алгоритма. В дальнейшем команда «шаг» или «ШАГ» приведет к выполнению очередного шага внутри выполняемого вспомогательного алгоритма.

До конца алгоритма

Допускается использование только в состоянии ПАУЗА. Программа выполняется непрерывно, но останавливается на первой встретившейся строке **кон** (как будто перед ней стоит вызов функции **пауза**).

Прервать

Прерывает выполнение программы. Допускается использование только в состояниях ВЫПОЛНЕНИЕ и ПАУЗА.

5.4.6 Отображение выполнения в области ввода-вывода

В начале выполнения программы в поле ввода вывода выводится строка-заголовок вида:
`> 16:39:48 - Новая программа* - Начало выполнения`

Далее под этой линией появляются все сообщения, выводимые программой (включая эхо ввода, см. ниже). В конце работы программы выводится итоговая строка и линия-разделитель. Предусмотрено 3 вида заключительной строки:

1. при нормальном завершении:
   `> 16:33:33 - Новая программа* - Выполнение завершено`

2. при ошибке выполнения:
   `> 16:32:38 - Новая программа* - ОШИБКА ВЫПОЛНЕНИЯ: утв ложно`

3. при прекращении выполнения командой «Прервать»:
   `> 16:30:51 - Новая программа - Выполнение прервано`

5.4.7 Список доступных алгоритмов

Для просмотра доступных алгоритмов следует выбрать пункт главного меню «Инструменты» → «Алгоритмы пользователя». Появится окно «Алгоритмы пользователя», отображающее древовидный список доступных алгоритмов. На первом уровне дерева находится список исполнителей, слева от каждого исполнителя отображается знак +. После нажатия на + раскрывается список алгоритмов этого исполнителя.

Исполнители разделяются на три вида, соответственно в списке они отображаются по разному:

- курсивом отмечаются исполнители пользователя, описанные в тексте программы;
- прямым шрифтом — подключенные встроенные исполнители;
- прямым серым шрифтом — доступные встроенные исполнители, но для использования их необходимо подключить с помощью команды `использовать` (см. 5.5.2).

При выделении алгоритма в списке отображается информация о нем в правой части окна «Алгоритмы пользователей», а именно: описание алгоритма и формат вызова (синтаксис).

5.4.8 Просмотр текущих значений величин

При выборе пункта главного меню «Инструменты» → «Величины» открывается окно с таблицей величин, состоящей из следующих столбцов:

- Исполнитель
- Алгоритм
- Тип
- Имя
- Значение

Каждой строке таблицы соответствует одна величина (переменная) программы.

Эта таблица содержит все величины, используемые в программе. Таблица является актуальной в каждый момент времени — так, если выполнение программы проходит при открытой таблице величин, то данные в таблице обновляются динамически.

5.5 Исполнители

5.5.1 Основные сведения. Виды исполнителей

Язык Кумир поддерживает работу с исполнителями. Три исполнителя (Робот, Чертежник, Файлы) встроены в систему Кумир. Другие исполнители:

- могут быть представлены в текущей программе;
- могут быть заранее описаны на языке Кумир и сохранены в стандартном формате сохранения Кумир-программ (файлы с расширением .kum) — внешние исполнители;
- могут являться автономными дополнительными модулями Кумира (например, Черепаха, Водолей, Кузнечик) — сетевые исполнители;

5.5.2 Команда «использовать»

Для каждого встроенного, внешнего или сетевого исполнителя, который используется в программе, в начале этой программы должна быть строка вида:
использовать имя_исполнителя

Например:
использовать Робот

Все строки **использовать...** должны располагаться в начале программы. Если используется несколько исполнителей, то порядок следования строк **использовать**... значения не имеет.

При синтаксическом разборе программы строка **использовать**... сообщает о возможности вызова алгоритмов указанного исполнителя. При выполнении программы при обработке строки **использовать**... производятся необходимые для указанного исполнителя подготовительные действия. Например, для исполнителей, написанных на языке Кумир, выполняется вступление (см. описание языка), для исполнителя Робот делается видимым окно наблюдения за Роботом и т. п.

Для исполнителей, представленных в текущей программе, наличие строки **использовать...** не обязательно, наличие такой строки «подразумевается».

5.5.3 Подготовка внешних исполнителей

Напомним, что *внешним* мы называем исполнитель, написанный на языке Кумир, и не представленный в текущей программе. Внешние исполнители хранятся в файлах стандартного формата, используемого для хранения Кумир-программ. Один файл может содержать описания нескольких исполнителей.

Такие файлы могут быть предварительно подготовлены с помощью системы Кумир. Для этого нужно подготовить в редакторе текст, содержащий нужные исполнители, и сохранить его командой «Сохранить» или «Сохранить как...» (см. 5.3.1).

5.5.4 Регистрация внешних и сетевых исполнителей

Общие сведения

Для того, чтобы использовать внешний или сетевой исполнитель, его нужно предварительно зарегистрировать. В строке **использовать**... можно указывать только имя:

- встроенного исполнителя;

- исполнителя, представленного в текущей программе (по желанию);
- зарегистрированного внешнего исполнителя.
- зарегистрированного сетевого исполнителя.

В конце сеанса работы системы Кумир список зарегистрированных исполнителей запоминается. В начале следующего сеанса этот список восстанавливается. Если в промежутке между сеансами файл с исполнителем был испорчен, или сетевой исполнитель стал недоступен, то соответствующий исполнитель не включается в список зарегистрированных. Никакое сообщение при этом не выдается.

Вспомогательное окно «Регистрация исполнителей»

Регистрация исполнителей производится с помощью строки «Исполнители» меню «Инструменты». При нажатии этой строки появляется вспомогательное окно «Регистрация исполнителей», позволяющее:

- *подключить* сетевой исполнитель;
- *зарегистрировать новый* внешний исполнитель;
- *удалить* исполнитель из списка зарегистрированных.

Окно содержит кнопки, позволяющие выполнить указанные действия, а также список уже зарегистрированных исполнителей. Для каждого из них указывается:

- имя;
- адрес и порт, по которому подключен сетевой исполнитель — для сетевых исполнителей;
- файл, в котором хранится текст исполнителя — для внешних исполнителей.

Размеры окна «Регистрация исполнителей», а также ширину каждого из его полей (имя исполнителя, файл) можно изменять с помощью мыши.

Регистрация нового исполнителя

Чтобы зарегистрировать новый исполнитель, нужно нажать соответствующую кнопку на окне «Регистрация исполнителей», после чего в ходе стандартного диалога выбрать имя файла. Система Кумир анализирует выбранный файл и в отдельном окне «Выбор исполнителей» выводит список исполнителей, описанных в этом файле.

Пользователь отмечает в этом окне нужные ему исполнители (не обязательно все). Если в выбранном файле есть исполнитель, имя которого совпадает с именем уже зарегистрированного исполнителя, то об этом выводится предупреждение в поле «Сообщения». Регистрация такого исполнителя невозможна.

Выбранные исполнители регистрируются и показываются в списке зарегистрированных исполнителей окна «Регистрация исполнителей».

Размеры окна «Выбор исполнителей» и ширину его полей можно настраивать с помощью мыши.

Удаление исполнителя из списка зарегистрированных

Отмена регистрации исполнителя производится с помощью окна «Регистрация исполнителей». Для этого нужно нажать на имя нужного исполнителя в списке, а затем нажать кнопку «Удалить».

5.6 Меню «Инструменты»

5.6.1 Общие сведения

Меню «Инструменты» содержит следующие пункты:

- Величины (см. 5.4.8)
- Доступные алгоритмы (см. 5.4.7)
- Новый текст
- Исполнители (см. 5.5.4)
- Редактировать стартовую обстановку
- Сохранить область вывода (см. 5.2.5)
- Очистить область вывода (см. 5.2.5)
- Настройки

Строка «Редактировать стартовую обстановку» относится к работе с исполнителем Робот и описана в соответствующем руководстве.

Ниже мы описываем строки «Новый текст» и «Настройки».

5.6.2 Пункт «Новый текст»

Открывает новое окно простого текстового редактора. Его можно использовать, например, для редактирования и просмотра входных и выходных файлов.

5.6.3 Пункт «Настройки»

Диалоговое окно «Настройки» разделено на 6 закладок:

- Новая программа
- Внешний вид
- Каталоги и файлы
- Подсветка кода (см. 5.3.6)
- Окно ввода-вывода
- Внешние утилиты

На окне имеется кнопка «Восстановить настройки по умолчанию», позволяющая вернуться к первоначальным настройкам.

Новая программа

Здесь задается текст программы, появляющийся в редакторе при выборе пункта меню «Программа» → «Новая программа». Кнопка «Взять из редактора» копирует текст, находящийся в текущий момент в редакторе.

Внешний вид

Параметры внешнего вида редактора. Можно выбрать шрифт для редактора (моноширинный), его размер и начертание, ширину отступа в начале строк программы. Также можно указать, показывать ли номера строк в редакторе.

Каталоги и файлы

На этой закладке:

- задается каталог ввода-вывода, который используется встроенным исполнителем Файлы;
- задается каталог стандартный исполнителей, в котором хранятся внешние исполнители системы Кумир;
- выводится информационная строка, показывающая какой файл хранит стартовую обстановку исполнителя Робот.

Окно ввода-вывода

Здесь задаются параметры окна ввода-вывода системы: цвет области ввода и ширина строки вывода.

Внешние утилиты

Здесь можно выбрать программу, используемую для просмотра документации по системе Кумир в формате PDF.

5.7 Автоматическое тестирование

Файл задания

Ученик получает файл, содержащий заготовки вспомогательного алгоритма (его должен написать ученик), и основного алгоритма, с помощью которого ученик сможет проверить правильность своего алгоритма.

Строки, выделенные темным фоном, защищены от изменения. Это значит, что имя алгоритма и список его параметров заданы заранее, и их нельзя поменять. Для выполнения задания нужно написать тело вспомогательного алгоритма и убедиться в правильности его работы, подготовив и выполнив соответствующий основной алгоритм.

5.7.1 Тестовый алгоритм

В задание включен также алгоритм тестирования вспомогательного алгоритма, подготовленного учащимся. Этот алгоритм по желанию учителя может быть скрыт от ученика (как на показано на предыдущих двух рисунках) или виден ему:

После того, как вспомогательный алгоритм будет готов, полученную программу можно проверить. Для этого необходимо выбрать пункт меню Программа→Запустить тестирование либо нажать комбинацию клавиш Ctrl+T. Эти действия приводят к выполнению алгоритма с именем `@тестирование`.

Тестирование может завершиться с двумя исходами: Если ошибок не обнаружено, то после окончания тестирования в поле ввода-вывода запишется строка “Тестирование завершено успешно”. Если в процессе выполнения тестирующего алгоритма произойдет

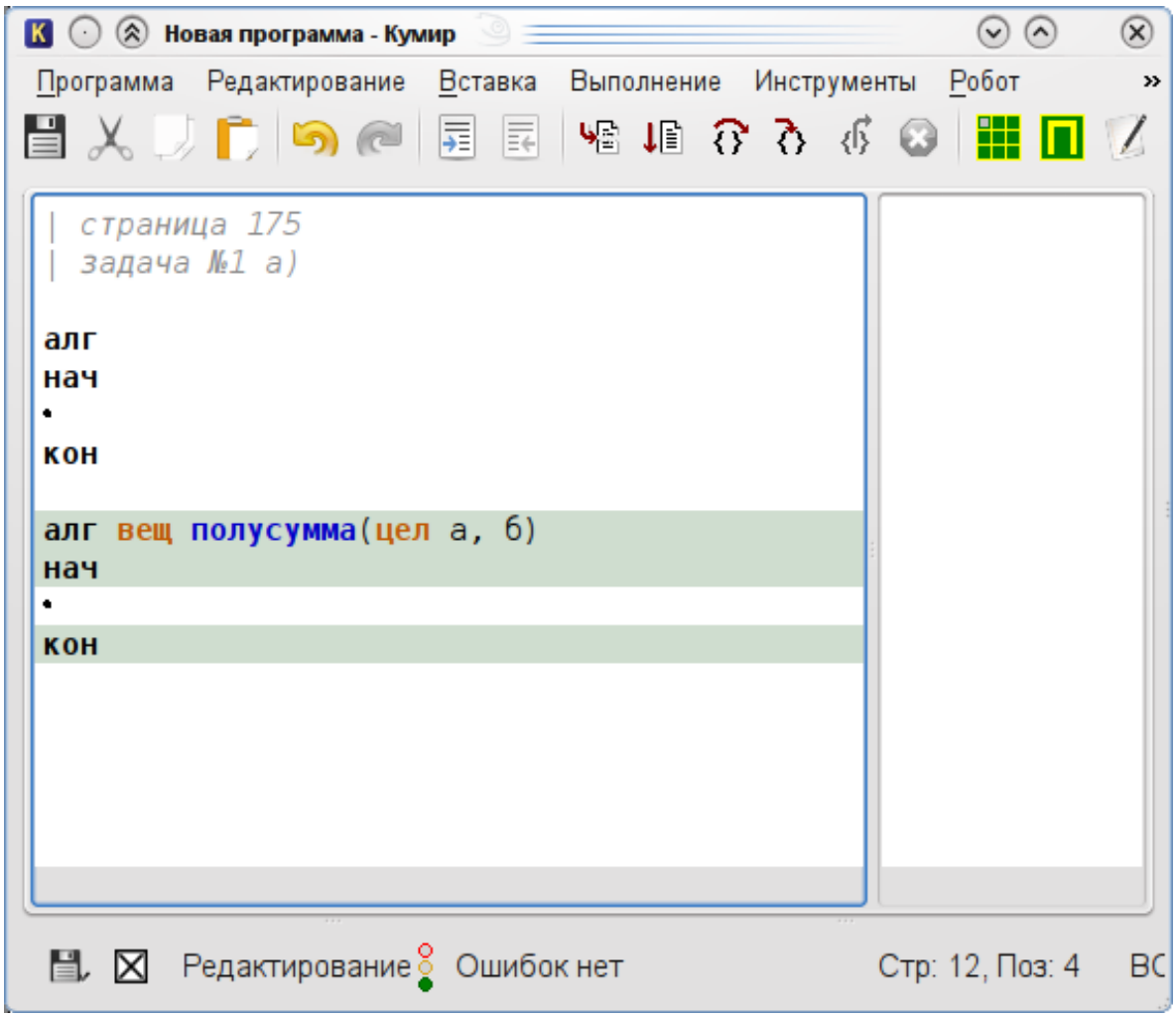

Рис. 5.1: Программа с заготовками вспомогательного алгоритма

ошибка (это означает, что есть ошибка в алгоритме ученика), то тестирование прервется с сообщением “Ошибка выполнения тестирования”.

Возможна ситуация, когда в заготовке программы не предусмотрена возможность тестирования. В этом случает при выборе пункта меню Программа→Запустить тестирование либо при нажатии комбинации клавиш Ctrl+T появится сообщение “Программа не содержит тестирующего алгоритма”.

5.8 Гипертекстовый учебник

В состав поставки КуМир входит гипертекстовый учебник, тесно взаимодействующий с системой. Этот учебник можно использовать с помощью Web-браузера. Вызовы гипертекстового учебника производится из меню Инструменты→Гипертекстовый учебник.

Замечание. В настоящее время текст данного учебника находится в стадии разработки.

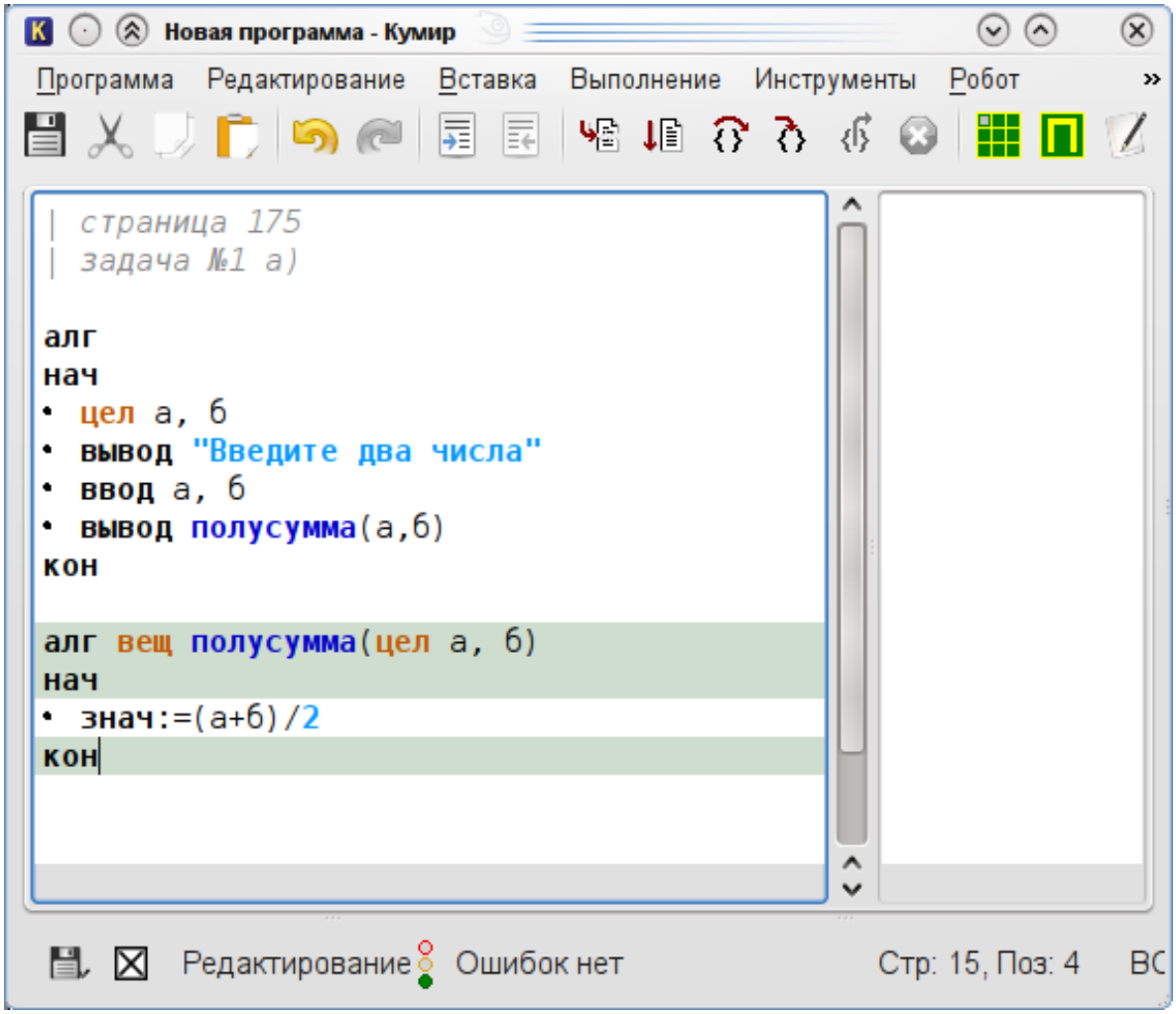

Рис. 5.2: Реализация программы учеником

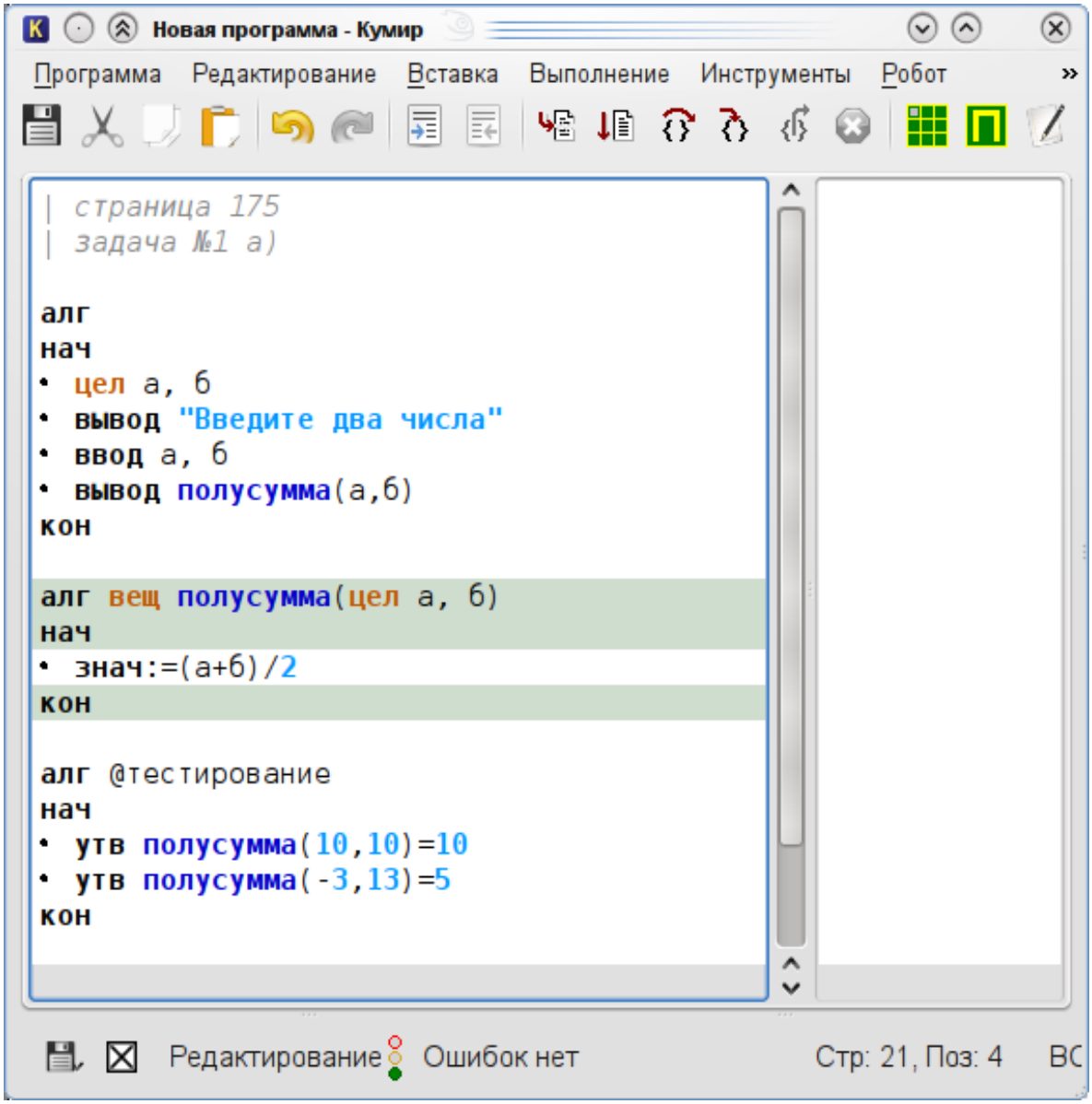

Рис. 5.3: Тестирующий алгоритм `@тестирование`

Приложение А

Справочник

А.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

А.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)

А.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)
начать чтение (**арг цел** ключ)
закрыть (**арг цел** ключ)
ф_ввод ключ, ...
ф_вывод ключ, ...
лог конец файла (**арг цел** ключ)
лог существует файл (**арг лит** имяФайла)
прочесть байт (**рез цел** байт, **арг цел** ключ)
записать байт (**арг цел** байт, **арг цел** ключ)

A.4 Общий вид алгоритма

алг имя (*аргументы и результаты*)
· **дано** *условия применимости алгоритма*
· **надо** *цель выполнения алгоритма*
нач
· *тело алгоритма*
кон

A.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*

A.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]	

A.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**

A.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

A.9 Арифметические операции и стандартные функции для работы с числами

Название операции	Форма записи
сложение	x + y
вычитание	x - y
умножение	x * y
деление	x / y
возведение в степень	x ** y

Название функции	Форма записи
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)

A.10 Операции сравнения чисел

Название операции	Форма записи
равно	x = y
не равно	x <> y
меньше	x<y
больше	x>y
меньше или равно	x<=y
больше или равно	x>=y

A.11 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

A.12 Операции для работы со строками

Название операции	Пример
слияние	а+б
вырезка	а[3:5]
взятие символа	а[3]
равно	а = б
не равно	а <> б

A.13 Другие встроенные алгоритмы

Функция	**Форма вызова**
Строковое представление целого числа	цел_в_лит(x)
Строковое представление вещественного числа	вещ_в_лит(x)
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(x)
Код символа в таблице Юникод	юникод(с)
Символ таблицы Юникод	символ2(x)
Код нажатой клавиши	клав
Текущее время в миллисекундах	время
Приостановка выполнения программы	пауза
Остановка выполнения программы	стоп