МИНИСТЕРСТВО РАДИОПРОМЫШЛЕННОСТИ СССР

АКАДЕМИЯ НАУК СССР
ОРДЕНА ЛЕНИНА СИБИРСКОЕ ОТДЕЛЕНИЕ
ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

УТВЕРЖДЕН
3533847.00042-01 33 01-ЛУ

ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ АВТОМАТИЗАЦИИ ШКОЛЬНОГО УЧЕБНОГО ПРОЦЕССА (ШКОЛЬНИЦА)

РУКОВОДСТВО ПРОГРАММИСТА

3533847.00042-01 33 01

Часть-I

1985

МИНИСТЕРСТВО РАДИОПРОМЫШЛЕННОСТИ СССР

АКАДЕМИЯ НАУК СССР

ОРДЕНА ЛЕНИНА СИБИРСКОЕ ОТДЕЛЕНИЕ

ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

УТВЕРЖДЕН

3533847.00042-01 33 01-ЛУ

ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ АВТОМАТИЗАЦИИ ШКОЛЬНОГО УЧЕБНОГО ПРОЦЕССА (ШКОЛЬНИЦА)

РУКОВОДСТВО ПРОГРАММИСТА

3533847.00042-01 33 01

Часть-I

1985

АННОТАЦИЯ

В документе описывается реализация пакета прикладных программ автоматизации школьного учебного процесса (ШКОЛЬНИЦА) на ПЭВМ "АГАТ" в версии AI.I. Приводится описание работы со всеми составными частями системы, синтаксис входных языков версии, описанные возможностей ПЭВМ и их использования. Построение материала ориентировано на неподготовленного пользователя-программиста, в связи с чем он рекомендуется для самостоятельного изучения системы.

СОДЕРЖАНИЕ

(в скобках приведен номер файла для модуля ИНСТР, см. Приложение 7)

Г.А.Звенигородский, В.А.Цикоза

СИСТЕМА "ШКОЛЬНИЦА"
ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

Предисловие

Эта инструкция предназначена для пользователей, участвующих в опытной эксплуатации системы "ШКОЛЬНИЦА", и содержит описание ее входного языка (РАПИРА) и основных инструментально-сервисных средств. Предполагается, что читатель знаком с каким-либо языком программирования массового применения и имеет в своем распоряжении персональную ЭВМ "АГАТ", оснащенную системой "ШКОЛЬНИЦА". Приведенные ниже сведения не являются каноническим описанием языка и не могут быть использованы для построения самостоятельных реализаций. Для получения более подробных сведений о языке и системе необходимо обратиться к другим источникам.

Семантику отдельных конструкций языка можно уточнить и прямым экспериментом.

Разработчики оставляют за собой право вносить незначительные изменения в синтаксис и семантику языка в соответствии с замечаниями и рекомендациями, появляющимися в процессе опытной эксплуатации.

Описание системы приводится по состоянию на I5 июня I985 года.

РАЗДЕЛ I. ОБЩИЕ СВЕДЕНИЯ О СИСТЕМЕ. КОНФИГУРАЦИЯ ТЕХНИЧЕСКИХ СРЕДСТВ. КОМПЛЕКТ ПОСТАВКИ. ОСОБЕННОСТИ ДАННОЙ ВЕРСИИ

I.I. Назначение и структура системы

Система "ШКОЛЬНИЦА" разработана в Вычислительном Центре СОАН СССР при участии НГУ и МУПК Советского района г.Новосибирска. Система предназначена для оснащения персональных микроЭВМ, используемых в учебном процессе школ и других вычислительных кабинетов.

Входные языки системы были разработаны Г.А.Звенигородским, реализация описываемой версии проведена Е.В.Налимовым, В.А.Цикозой, П.А.Земцовым, Н.Г.Глаголевой.

Основу системы составляет интерпретатор учебно-производственного языка РАПИРА. Это диалоговый бестиповый язык высокого уровня, предназначенный для оперативного решения на ЭВМ небольших задач и для обучения пользователей, в том числе - студентов и школьников. Второй входной язык - РОБИК - предназначен для обучения основам и навыкам программирования младших школьников.

Кроме интерпретаторов РОБИКА и РАПИРЫ в описываемую версию системы входят:

диалоговый экранный редактор программных текстов;

дисковая операционная система, позволяющая работать с текстовыми файлами прямого доступа, хранящимися на гибких магнитных дисках;

графическая система "ШАГА", представленная в рассматриваемой версии несколькими комплектами графических процедур;

программа начальной разметки рабочих дисков пользователя;

автономный отладчик программ в машинных командах, в основном совпадающий по возможностям с монитором ЭВМ "АГАТ";

комплект тригонометрических и некоторых других наиболее употребительных математических функций;

пакет сбора статистики по допущенным в течение сеанса работы ошибкам;

базовый комплект исполнителей интерпретатора РОБИК.

Последние пакеты можно использовать в качестве примеров прикладных программ, написанных на языке РАПИРА.

I.2. Версии системы

Система "Школьница" была первоначально реализована на ЭВМ "Apple 2" с английской лексикой предписаний РАПИРЫ, затем в расширенном виде перенесена на ЭВМ "АГАТ" с переходом на предусмотренную каноническим описанием входных языков русскую лексику.

В этой инструкции описывается версия системы для ЭВМ "АГАТ" (индекс А1.1).

I.3. Конфигурация технических средств

Для работы системы необходим следующий минимальный набор технических средств:

I. ЭВМ "АГАТ" с объемом ОЗУ не менее 64К байт и ПЗУ не менее 32К байт (7 исполнение) и встроенным дисководом для гибких магнитных дисков диаметром 133 мм.

2. Цветной телевизионный монитор.

Для расширения возможностей системы могут быть дополнительно подключены:

I. Игровой блок (комплект потенциометров).

2. Печатающее устройство.

3. Второй дисковод, подключенный к тому же разъему, что и первый.

I.4. Комплект поставки

В комплект поставки системы входят:

I. Гибкий диск Ш1 с базовой частью системы "ШКОЛЬНИЦА" и комплектом поставляемых пакетов.

2. Гибкие диски Ш2 и Ш3, содержащие настоящую инструкцию.

3. Комплект документации, в частности, включающий в себя настоящую инструкцию.

Рекомендуется перед началом работы изготовить рабочую копию диска Ш1 и работать далее с ней, используя исходный диск в качестве контрольной копии.

РАЗДЕЛ 2. ЗАГРУЗКА СИСТЕМЫ. НАЧАЛО РАБОТЫ

2.I. Загрузка системы и запуск интерпретатора РАПИРЫ

Для загрузки системы необходимо вставить диск Ш1 в дисковод включить машину. При этом на экране монитора появляется титульная информация и общее меню системы "Школьница":

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x                                  x
x      ПРОГРАММАЯ СИСТЕМА          x
x                                  x
x         "ШКОЛЬНИЦА"              x
x                                  x
x     (НАЧАЛЬНЫЙ ЗАГРУЗЧИК)        x
x                                  x
x          ВЕРСИЯ I                x
x                                  x
x       НОВОСИБИРСК I985           x
x                                  x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

В СИСТЕМЕ ЕСТЬ:

Р - РАПИРА-ИНТЕРПРЕТАТОР

О - АВТОНОМНЫЙ ОТЛАДЧИК

Ф - РАЗМЕТКА РАБОЧИХ ДИСКОВ

ЧТО?

На запрос необходимо нажать указанную в меню клавишу, обозначающую РАПИРУ. В случае ошибочного ответа запрос повторяется до тех пор, пока не будет получен один из перечисленных в меню вариантов ответа.

Когда нужная клавиша нажата, система загружает с диска интерпретатор. В случае успешной загрузки в верхней, информационной, строке экрана появляется надпись

РАПИРА-АГАТ I.Iвк

в средней части экрана - надпись

ЕСЛИ ВЫ НЕ ЗНАКОМЫ С СИСТЕМОЙ,

НАБЕРИТЕ

ВКЛ МОДУЛЬ ИНСТР;

а в нижнем левом углу - приглашение к диалогу - знак #

Признаком того, что система ожидает ввода с клавиатуры, всегда служит появление на экране курсора. Курсор отмечает на экране монитора позицию, начиная с которой будет отображаться вводимая информация. В системе "ШКОЛЬНИЦА" он имеет вид мигающего подчерка. Если курсор на экране отсутствует, значит ввод по той или иной причине невозможен (идет исполнение программы, машина "зависла" и т.п.).

Выдача приглашения означает, что система готова к работе, можно вводить любые предписания языка РАПИРА.

ВНИМАНИЕ! Предписания языка РАПИРА можно набирать только после появления на экране приглашения (#).

Другие программы и подсистемы "Школьницы" вызываются из главного меню описанным выше способом либо из интерпретатора РАПИРЫ по правилам язнка.

ОСОБЕННОСТИ ЗАГРУЗКИ:

I. Если используются два дисковода, загрузочный диск следует вставлять в дисковод № I.

2. Если машина уже включена и выдает приглашение какой-либо другой системы (например, "ж", " ⌋ "), нужно провести перезагрузку системы с соответствующего дисковода, вставив в него предварительно системный диск "Школьницы".

3. Если вы не знаете, как проводится перезагрузка из другой системы, или если она у вас не получилась, лучше всего выключить машину и включить ее снова. При этом в дисководе должен находиться диск ШI.

4. Чтобы вернуться обратно в системное меню или вызвать другую систему, необходимо:

вставить в дисковод диск с нужной системой (в случае "ШКОЛЬНИЦЫ" - ШI),

после появления приглашения и курсора набрать предписание

ВКЛ МЕНЮ;

нажать клавишу с изображением изогнутой стрелки (перевод строки).

РАЗДЕЛ 3. ВЫЧИСЛЕНИЕ ПРОСТЫХ ВЫРАЖЕНИЙ. ВЫВОД И ПРИСВАИВАНИЕ. ОСНОВНЫЕ ОБЪЕКТЫ РАПИРЫ И ОПЕРАЦИИ НАД НИМИ

3.I. Вычисление простых выражений

Знакомство с возможностями языка РАПИРА удобнее всего начинать с простейших форм предписаний вывода, используемых для вычисления числовых, текстовых и структурных выражений.

Если после появления на экране приглашения набрать на клавиатуре строку

?25;

и выполнить перевод строки (нажать клавишу с изогнутой стрелкой), то на экране появится число 25.

Чтобы вычислить значение выражения

$$5986 \times I928 + 96I^{2} - (5I2 : 256)$$

необходимо набрать

?5986жI928+96Iжж2-5I2/256;

и выполнить перевод строки. На следующей строке сразу появится результат вычисления - число I2464527.

В общем случае для вычисления числового выражения необходимо:

I. Набрать вопросительный знак (?).

2. Набрать нужное выражение.

3. Набрать точку с запятой ";".

4. Перевести строку.

Числовые выражения записываются по правилам, принятым в большинстве процедурных языков программирования: для обозначения операций сложения, вычитания, умножения, деления и возведения в степень используются знаки "+", "-", "ж", "/" и "жж", соответственно.

Приоритеты операций и правила расстановки скобок соответствуют общепринятым, причем допускаются только круглые скобки.

Для отделения целой части от дробной используется точка: число I2,4I (двенадцать целых и сорок одна сотая) записывается как I2.4I. Разрешена запись чисел в формате с плавающей точкой, при этом для обозначения десятичного порядка числа используется буква "E", например, $I0,I5xI0^{24}$ записывается как I0.I5E24, а $56xI0^{-I}$ - как 56E-I8.

Подчеркнем, что точка в записи дробного числа именно разделяет цифры. Поэтому записи 5. и .25, разрешенные в некоторых языках, в РАПИРЕ недопустимы. Следует писать 5.0 и 0.25.

Между числами и знаками операций может стоять любое число пробелов, но внутри числа пробелы не допускаются. Недопустимы они и внутри составных знаков операций, например "**".

Так, предписания

```
?25+I6*48;
```

и

```
? 25 + I6* 48;
```

эквивалентны, но предписание

```
?2 5+I6*48;
```

ошибочно (можете для проверки набрать все три предписания на клавиатуре; заодно это позволит ознакомиться с формой выдачи сообщений об ошибках).

Точка с запятой ";" является признаком конца предписания. Если ее не поставить и перевести строку сразу после записи предписания, то на следующей экранной строке появится приглашение" > ". В общем случае оно показывает, что предыдущее предписание еще не

закончено, и его следует закончить на новой строке (в данной ситуации -- набрать недостающую точку с запятой).

На одной строке можно указывать несколько предписаний. Разделяются они все той же точкой с запятой.

Максимальная длина вводимой строки программы на РАПИРЕ - 255 символов. За 8 позиций до конца строки с каждым набираемым символом выдается звуковое предупреждение. Когда же этот лимит исчерпан, ввод новых символов временно прекращается до нажатия клавиши перевода строки.

При нажатии клавиши "перевод строки" исполняются все предписания, законченные на этой строке. Поэтому эта клавиша в некоторых источниках называется также клавишей исполнения. Например, после набора строки

```
?5*8; ?3+4; ?I5*67;
```

на экране появятся числа:

```
40
 7
I005
```

Обратите внимание, что после выполнения каждого предписания вывода автоматически переводится строка!

Для вычисления нескольких выражений их можно указать в одном предписании вывода, разделяя запятой. В этом случае результаты будут выданы на одной строке. Например, при выполнении предписания

```
?5*8,3+4,I5*67;
```

на экране появится такая строка:

```
40 7 I005
```

Перевод строки эквивалентен пробелу. Это значит, что предписания можно располагать на нескольких строках. Никакие знаки или

признаки переноса при этом недопустимы! Например, предписание

```
?5I2*296;
```

можно набрать и так:

```
?
5I2
* 296:
```

Как уже отмечалось, в этом случае при переходе к новой строке система вместо " # " выдает приглашение " > " . Также недопустимо разбивать переводом строки числа и сложные знаки операций.

Внимание! На одной строке экрана телемонитора помещается сравнительно небольшое число символов. Поэтому в системе предусмотрен автоматический переход на новую строку после заполнения очередной без нажатия клавиши перевода строки и без выдачи приглашения. Такой переход не разрывает строку и может появляться внутри чисел и других лексем РАПИРЫ, описанных ниже.

Таким образом, одна строка РАПИРЫ может занимать несколько экранных строчек. Пробелы и переводы строки в любом количестве не влияют на исполнение программы. Все эти возможности можно использовать при оформлении программ для облегчения их чтения.

Прежде чем читать дальше, рекомендуем поэкспериментировать с вычислением числовых выражений.

3.2. Некоторые ограничения и отличия описываемой версии РАПИРЫ от канонического описания

I. Кроме операций, перечисленных в предыдущем параграфе, над целыми числами предусмотрена операция деления нацело, обозначаемая "//". Например, при выполнении предписания

```
?86//3;
```

будет выдано число 28.

2. В описываемых версиях языка возможна работа с целыми числами из диапазона $|N| < 2*10^{16}$ и дробными числами из диапазона $10^{-128} < |x| < 10^{128}$. Точность представления дробных чисел - I2 значащих цифр. Числа, порядок которых меньше -I27, преобразуются в ноль. Таким образом, диапазон представления целых чисел больше, чем диапазон дробных. Скорость выполнения операций над целыми числами также больше, чем над дробными примерно вдвое. Поэтому при решении прикладных задач, требующих высокой точности и повышенной скорости, рекомендуется использовать целую арифметику (заменяя деление делением нацело и вводя соответствующий масштабный множитель).

Следует отметить, что длина числа, набираемого в составе выражения, не может быть больше длины вводимой строки (255 символов), хотя в результате арифметических операций могут получаться и большие числа. Полезно посмотреть, как выглядит на экране число $2*1000$.

3. В описываемых версиях не реализовано возведение чисел в дробную степень. Для этого следует воспользоваться функциями из стандартного пакета ФУНКЦИИ для вычисления логарифма и экспоненты.

4. В описываемых версиях исключена операция "унарный плюс". Поэтому выражения +I0-5 или 8*(+6) считаются ошибочными.

3.3. Диагностика ошибок и их исправление

Если во вводимой строке содержится несколько предписаний, анализ и выполнение каждого из них начинается только после выполнения предыдущего предписания.

Обнаружив ошибку в записи очередного предписания, система подает звуковой сигнал, дублирует на экране строку, в которой допущена ошибка, выделяя ошибочное место красным цветом, и выдает сообщение об ошибке. Предписание, в котором обнаружена ошибка, не исполняется. Вся оставшаяся часть введенной строки (в том числе все последующие предписания) игнорируется.

Все возможные сообщения об ошибках, пояснения, помогающие понять причины их возникновения, а также способы устранения приведены в Приложении 4.

При обнаружении ошибки во время исполнения предписания выполняются те же действия, но строка с ошибкой не выдается. Попробайте вычислить, например, выражение 5/0.

Если ошибка замечена в процессе набора, до перевода строки, то ее можно исправить, не дожидаясь сообщения. Самый простой способ сделать это - используя клавишу " <-", вернуться к месту ошибки, исправить ее, а затем набрать оставшуюся часть заново или вернуться к концу строки клавишей "->". После этого можно продолжать набор предписания.

Если ошибка своевременно не замечена, и получено диагностическое сообщение, необходимо ввести предписание заново, исправив ошибку. Для этого не обязательно еще раз набирать его: если предыдущий текст еще виден на экране, достаточно подвести курсор к его началу и ввести этот текст в машину с помощью клавиш "-> " и "ПВТ". Разумеется попутно следует исправить ошибку.

Подробнее этот механизм описан в Приложении 2, но при начальном знакомстве с системой его знание необязательно.

3.4. Объекты языка. Имена и их значения. Присваивание

Числа, использованные в приведенных выше примерах, значения выражений являются примерами объектов РАПИРЫ. Объекты - это те данные, которыми оперирует программа.

В описываемой версии РАПИРЫ различаются следующие виды объектов:

целые числа;
дробные (вещественные) числа;
тексты;
множества;
кортежи;
записи;
процедуры;
функции;
файлы;
пусто.

С числами вы уже знакомы, другие виды объектов будут рассмотрены позже.

Объекты, указанные непосредственно в тексте программы, называются константами, а объекты, получающиеся в результате некоторых действий над другими объектами - промежуточными (временными) значениями. После выполнения любой операции над таким значением оно уничтожается. Так, например, уничтожаются все промежуточные результаты, возникающие при вычислении выражения.

Чтобы сохранить в памяти некоторое промежуточное значение, ему необходимо дать имя или, иначе говоря, присвоить его некоторому имени. Такой объект называется значением имени.

Форма имени в РАПИРЕ традиционна для языков программирования: оно состоит из букв и цифр, причем на первом месте должна стоять буква. В связи с отсутствием на клавиатуре ЭВМ "АГАТ" строчных букв в описываемой версии допускается использование только заглавных букв; подстрочные символы (например, широко распространенные в математике индексы) также запрещены. Совпадающие по написанию русские и латинские буквы не различаются (буквы "У" и "Y" различаются!). Символы в имени разрешено для удобства чтения разделять одиночными подчерками.

Примеры имен:

X

У1

WORD_16A2

ИМЯ_УЧЕНИКА

Примеры ошибочных имен:

6НЗ	(имя не может начинаться с цифры)
СКОРОСТЬ	(в начале и в конце имени подчерк недопустим)
ВЫСОТА__ГОРЫ	(два подчерка не могут стоять подряд)
Счетчик	(строчные буквы недопустимы).

Длина имени в РАПИРЕ ограничена только длиной вводимой строки (255 символов). При сравнении имен учитываются все символы, включая подчерки.

Количество различных по написанию имен, используемых в программе, не должно превышать 256 (считая стандартные имена, локальные параметры процедур и функций, имена полей записей, см.3.7, 6.6).

Имена можно использовать в программе без какого-либо предварительного объявления. Всякому имени (за исключением защищенных, см. 3.5) можно присвоить значение любого вида. Допускается присваивание одному и тому же имени значений различных видов в произвольном порядке.

Предписание присваивания в простейшем случае имеет вид:

выражение -> имя;

(обозначения, использованные для описания синтаксиса, поясняются в 4.1). Примеры:

5-> X;

3+X**2+17**X-> СКОРОСТЬ;

X+8-> X;

Для просмотра текущих значений имен удобно использовать простейшую форму вывода, описанную в 3.1. Например, если после трех записанных выше присваиваний набрать:

?X,СКОРОСТЬ;

то на экране будут выданы числа:

13 1420000

Если в выражении или предписании использовано имя, то подразумевается объект, запомненный под этим именем. Пример:

? X**2+СКОРОСТЬ/2;

Имена, которым не были присвоены значения в программе, имеют по умолчанию значение "пусто".

Предусмотрено стандартное имя ПИ, имеющее значение 3.14159265359.

Просмотреть список всех имен (кроме стандартных), использованных в системе с момента загрузки (перезапуска) РАПИРЫ и имеющих непустое значение, можно при помощи предписания

КАТАЛОГ ИМЕН;

Для просмотра списка, содержащего и имена с пустыми значениями, используется предписание

КАТАЛОГ ВСЕХ ИМЕН;

Пример выдачи каталога имен на экран:

—КАТАЛОГ ПРОЦЕДУР —

(Ч) СТАРТ ..... ПРОЦ

—КАТАЛОГ ИМЕН—

СУММА ..... ЦЕЛ

(П) ТАБЛИЦА ..... КОРТ

РАБ_ИМЯ .....ПУСТ

Здесь после имени указывается вид его значения, а перед – – тип защиты (если она есть).

3.5. Защита имен

Начинающие пользователи системы могут при первом чтении пропустить этот параграф.

Присваивать новое значение можно не каждому имени. В РАПИРЕ предусмотрена возможность установить на некоторые имена защиту от присваивания. Различается три вида защиты: частичная, полная и абсолютная (системная).

Если на имя установлена частичная защита, то его значение может быть изменено только после дополнительного подтверждения. При попытке присвоить новое значение такому имени система выдаст на экран запрос:

имя : ПРИСВАИВАТЬ (Д/Н)?

и только в случае положительного ответа присваивание будет выполнено.

Если на имя установлена полная защита, то попытка присваивания ему будет рассматриваться как ошибка. Чтобы изменить значение имени, нужно сначала выключить его защиту.

Абсолютная (системная) защита устанавливается и отменяется системой и не может быть изменена пользователем. Такую защиту имеют, например, имена процедур и функций в момент вызова, имена-параметры циклов перебора, имена важнейших стандартных процедур и функций и т.п. Эта защита служит для предотвращения побочных эффектов, т.е. действий, могущих привести к нарушению правил выполнения предписаний. Подробнее об этом можно прочитать 4.5.

Для включения и выключения полной и частичной защиты используются предписания "ВКЛ" и "ВЫКЛ" (диаграмма 26). Например, предписания

ВКЛ ЧАСТ ЗАЩИТУ ИМЕН: А, Б;

ВКЛ ЗАЩИТУ: В;

устанавливают на имена А и Б частичную защиту, а на имя В – полную;

предписание

ВЫКЛ ЗАЩИТУ : Б,В;

снимает установленные ранее защиты с имен Б и В.

Необходимо отметить, что включение полной защиты имени отменяет частичную и наоборот. Если упомянутые в предписании "ВЫКЛ" имена не имеют никакой защиты, то никаких действий над ними не производится.

В каталоге имен перед защищенными именами в скобках указывается первая буква названия защиты.

Механизм защиты имен позволяет избежать случайной потери особо важной информации из-за неосторожных присваиваний в процессе диалога с ЭВМ, он способствует соблюдению дисциплины программирования и весьма полезен при отладке.

3.6. Тексты и операции над ними

Рассмотрим еще один вид объектов - тексты. Текст - это произвольная последовательность символов. При записи в составе программы тексты заключаются в кавычки, например:

"ТЕКСТ", "SIMPLE TEXT","123+45"

Все символы текста, включая пробелы, занумерованы слева направо, начиная с единицы, например:

"НУМЕРАЦИЯ"
123456789

Кавычки, окаймляющие текст, в его состав не входят.

Тексты могут входить в состав выражений и над ними могут выполняться операции. Предусмотрены следующие операции над текстами:

1. Определение мощности - унарная операция, обозначаемая знаком " # ". Мощностью текста называется количество символов в нем. Например, в результате исполнения предписаний

"КРОКОДИЛ" -> Т; ? #Т;

на экран будет выдано число 8.

2. Конкатенация (слияние) - бинарная операция, обозначаемая знаком "+". Например, после исполнения предписания

"ВЕЛО"+"СИПЕД" -> Т;

имя Т получит значение "ВЕЛОСИПЕД".

3. Выборка - извлечение одного символа по его номеру в тексте (индексу). Обозначается квадратными скобками, например:

Т [7], (Т+"Ы") [9], "ДИНОЗАВР" [N+5].

Результат выборки - литера, т.е. текст единичной длины. В качестве индекса может быть использовано любое выражение с целым положительным значением, не превосходящим длины текста. Примеры:

выражение "ДИНОЗАВР" [3] имеет значение "Н";

в результате исполнения программы

5-> X; "КРОКОДИЛ" -> Т; ?(Т+Т) [3ж X];

будет выдана буква "И".

4. Вырезка - извлечение участка текста, определяемого индексами начального и конечного символов. Обозначается так же, как и выборка, но в скобках указываются два индексных выражения, разделяемые двоеточием, например:

А [3:4], "БЕГЕМОТ" [N-2:N+1].

К условиям на значения индексных выражений добавляется еще одно: второй индекс должен быть не меньше первого. Примеры:

значение выражения "ВЕЛОСИПЕД" [5:7] есть текст "СИП";

в результате исполнения программы

"ПРИМУС" -> Х;

Х [4:4] +Х [3] +Х[6]+Х[#Х]+Х[#Х-1]+Х[2:3]-> РЕКА;

? РЕКА;

на экран будет выдано слово

МИССУРИ.

Обратите внимание, что тексты выводятся без окаймляющих кавычек!

* * *

Операции выборки и вырезки могут стоять как слева, так и справа от знака присваивания. В последнем случае они позволяют изменять отдельные символы или участки текстов, являющихся значениями имен. Например, после исполнения предписаний

"КРОКОДИЛ" -> К; "Я" -> К[3]; "УШКА" ->К[5:#К]

значением имени К станет текст "КРЯКУШКА".

Используя такие присваивания, нужно помнить, что они не меняют длины текста, поэтому длина текста в левой части предписания присваивания должна быть в точности равна длине заменяемого участка.

Если значением имени Д является текст "ТЕКСТ С ОПЕЧТКОЙ", то нельзя исправить ошибку с помощью присваивания

"ЧАТ" -> Д[#Д-4:#Д-3];

Требуемого результата можно достичь, комбинируя вырезку и конкатенацию:

Д[I:#Д-4]+"А"+Д[#Д-3:#Д]-> Д;

Некоторые частные случаи и ограничения.

I. Как слева, так и справа от знака присваивания можно использовать несколько выборок и вырезок подряд, поскольку результат этих операций - тоже текст, и к нему можно снова применять все операции над текстами. Например:

А[5:Io][2:5][4][I] -> Х;

Подробнее эта конструкция описана в 3.8 на примере кортежей, обрабатываемых аналогично.

2. При вводе текста в составе предписания с клавиатуры на него распространяются все правила набора лексем. В частности, текст не может быть разорван переводом строки и продолжен на другой строке, поэтому его длина на вводе не может превышать 253 литер

(плюс две окаймляющие кавычки - всего 255 символов). Однако длина текста, получающегося при выполнении операций над другими текстами, ограничивается только объемом доступной памяти и может достигать нескольких тысяч литер.

3. При вводе текста с клавиатуры в его состав могут входить символы, отсутствующие в алфавите языка, а потому недопустимые в программе. К ним, например, относятся управляющие символы (набираемые функциональными клавишами или на регистре "УПР"). Управляющий символ попадает во входную строку только после нажатия клавиши "УПР-U" и высвечивается желтым цветом (см. также Приложение).

4. Если в состав текста требуется включить кавычку, то ее нужно повторить дважды. Например, после выполнения предписания

"ЛЕДОКОЛ""АЛЬБАТРОС""->КОРАБЛЬ;

значением имени КОРАБЛЬ станет текст

ЛЕДОКОЛ "АЛЬБАТРОС"

5. Допускается использование пустого текста "". Его длина равна нулю.

6. Есть возможность сформировать текст заданной длины, состоящей из одинаковых символов (см. Приложение, функция ФТЕКСТ).

3.7. Составные структуры данных в РАПИРЕ. Формирование структур.

Составными структурами в РАПИРЕ являются множества, кортежи и записи

Множество - это неупорядоченная совокупность произвольных попарно различных элементов.

Кортеж - это упорядоченная совокупность произвольных элементов.

Запись - это совокупность различных именованных полей, значением каждого из которых может быть произвольный элемент.

Элементом составной структуры может быть объект любого вида (см. 3.4). Таким образом, составные структуры могут быть вложенными: элементом множества, например, может в свою очередь оказаться множество, кортеж или запись.

Для образования составных структур из нескольких объектов используется операция формирования. Элементы формируемой структуры разделяются запятыми и заключаются в скобки, соответствующие ее виду: < > - для кортежей, <* *>- для множеств, <¤ ¤>- для записей. При формировании записи перед каждым элементом через двоеточие указывается имя поля. Примеры:

< 5, "МАМА", <I,5> ,5> -> К;

<*I,I+I,К,I,<I,5>,2*> ->М;

<¤ИМЯ:"ИВАН", ФАМИЛИЯ: "ПЕТРОВ"¤>->ЗАП;

После исполнения этих предписаний значением имени К станет кортеж из четырех элементов: числа 5, текста "МАМА", кортежа <I,5> и еще одного числа 5.

Значением М будет множество из четырех элементов: чисел I,2, кортежей К и <I,5>. Обратите внимание, что одинаковые элементы учитываются в множестве только один раз, а в кортеже - столько раз, сколько указано. Например, кортеж <5,5,5> состоит из трех элементов, а множество <*5,5,5*> - из одного. В этом легко убедиться, набрав предписание вывода с использованием этих конструкций:

?<5,5,5>,<*5,5,5*>;

Заодно посмотрите, в каком виде выдаются множества и кортежи.

Значением имени ЗАП станет запись из двух полей: ИМЯ и ФАМИЛИЯ. Названия полей записываются по тем же правилам, что и имена (см. 3.4).

Обратите внимание, что в состав структуры включаются значения выражения или имен, указанных при формировании, а не сами эти имена. Например, после выполнения предписаний

I->A; <*A*>->B; I0->A;

значением В станет множество из единицы. Изменение значения имени А в дальнейшем не приведет к изменению В (проверьте это, выдав значения А и В). Иными словами, элементом множества станет такой же, но не тот же объект.

Операция формирования может входить в состав более сложных выражений с использованием операций над составными структурами, описанными в следующих параграфах.

Особенности и ограничения

I. В отличие от текста составная структура может занимать несколько строк:

<I,2,
3
,4
>->Х;

2. Допускается использование пустых множеств <* *>и кортежей< >, однако пустые записи не допускаются. Не следует путать пустой кортеж (множество) с кортежем (множеством), состоящим из одного элемента "пусто"! Пустые множество и кортеж не совпадают друг с другом и с пустым значением.

3. В записях и множествах элементы не упорядочены (что соответствует математическому содержанию этих понятий). Поэтому при выводе этих структур их элементы могут располагаться в произвольном порядке.

4. Максимальное число элементов множества и записи - 256; максимальная длина кортежа и глубина вложенности всех структур зависит только от объема памяти и размеров системных стеков.

5. Если в программе требуется создать большой кортеж, то быстрее не добавлять по одному элементу к пустому кортежу, а сразу сформировать кортеж требуемой длины и затем изменять его элементы. Такой кортеж формируется стандартной функцией ФКОРТ (см.Приложение 6).

3.8. Операции над кортежами

Над кортежами в РАПИРЕ предусмотрены такие же операции, как и над текстами (3.6).

1. Определение мощности (#).
2. Конкатенация (+).
3. Выборка ([N]).
4. Вырезка ([N1:N2]).

Они выполняются так же, как операции над текстами, если вместо символов рассматривать элементы кортежа.

Примеры:

```
<1,5, "ПРОБА", <"А", "Б", "СЛОВО">> ->Ж;
#К ->МОЩНОСТЬ;
К+<"НОВОЕ СЛОВО"> ->К2;
К [3]->ВЫБОРКА;
К [2:3]->ВЫРЕЗКА;
К[4] [3] [2]->ЦЕПОЧКА;
```

в результате исполнения этих предписаний значения имен станут такими;

```
МОЩНОСТЬ = 4
К2 = <1,5 "ПРОБА",<"А", "Б", "СЛОВО">, "НОВОЕ СЛОВО">
ВЫБОРКА = "ПРОБА"
ВЫРЕЗКА = <5, "ПРОБА">
ЦЕПОЧКА = "Л"
```

Если теперь выполнить присваивания:

```
"СПОРТ" -> К [4] ; <1,2>->К2 [4:5] ;
```

то значения имен К и К2 станут такими:

```
К - <1,5 "ПРОБА", "СПОРТ">
К2 -<1,5 "ПРОБА", 1,2>
```

Ограничения на допустимые значения индексов те же, что и для текстов.

Единственное, но весьма существенное различие между операциями вырезки и выборки для текстов и кортежей заключается в том, что для текстов выборка и вырезка формируют объекты одного вида - тексты. Например, выражения Т[2]и Т[2:2]равноправны и выделяют одну литеру. Для кортежей результатом вырезки всегда будет кортеж, а выборки - элемент любого вида (который, конечно, может быть и кортежем). Поэтому для кортежей равноправными будут выражения К[2:2] и<К [2]>.

Здесь недаром подчеркнуто "выражения", К [2:2]и <К [2]>. К [2:2] справа от знака присваивания нельзя заменить на <К [2]>; так как последнее не является именем.

Если выборка или вырезка использованы справа от знака – то вырезка нужно всегда присваивать кортеж, а выборке – любой элемент. Он может тоже быть кортежем, но это другой вопрос. Если, например, значением имени КРТ является кортеж <I,2,3>, то присваивание

<4> –> КРТ [2] ;

сделает его значением кортеж <I, <4> 3>, а присваивание

<4> –> КРТ [2:2] ;

кортеж <I,4,3>

При работе с вложенными кортежами удобно использовать многократные вырезки и выборки. В этом случае можно использовать сокращенную запись: пару скобок "][" заменять запятой. Например, следующие выражения эквивалентны:

Т [3] [2:5] [3] и Т [3,2:5,3]

Такая конструкция особенно удобна, когда вложенные кортежи используются для моделирования двумерных и многомерных таблиц.

3.9. Операции над множествами

Для множеств в РАПИРЕ предусмотрены четыре основные операции:

1. Определение мощности (#).
2. Объединение (+).
3. Пересечение (*).
4. Разность (–).

Все четыре операции имеют обычный математический смысл.

Примеры:

выражение	значение
#<*I,2,I,I,I,I+I*>	2
<*5,8*>+<*7,8,6*>	<*5,6,7,8*>
<*I,2,3*> *<*8,I,3*>	<*3,I*>
<*8,I0,5;6*> –<*7,I0,8*>	<*6,5*>

С использованием перечисленных операций могут быть построены и более сложные выражения, например: A+B–A*B (симметрическая разность).

При работе с множествами необходимо иметь в виду, что в отличие от математической записи в РАПИРЕ пересечение имеет более высокий приоритет, чем объединение и разность. Во избежание недоразумений рекомендуется использовать скобки.

Отметим также, что в отличие от кортежа, всякое изменение множества-значения имени возможно лишь путем присваивания этому имени нового значения.

3.I0. Операции над записями

Для записей в РАПИРЕ предусмотрена единственная операция - доступ к полю, которая обозначается точкой. Например, значением выражения<Ж(СКОРОСТЬ: 50, ВЫСОТА:200, НАПРАВЛЕНИЕ: "ЮГ")Ж.СКОРОСТЬ является число 50.

Операция доступа к полю также может стоять справа от знака присваивания, позволяя изменять значение отдельных полей записи:

```
250-> ПОЛЕТ. ВЫСОТА;
```

никакие другие операции над записями недопустимы. В частности, невозможно изменить количество полей в записи и их имена.

3.II. Процедуры и функции. Вызов.

Процедура (и функция) - это программа или часть программы, имеющая самостоятельное имя. Это имя указывается при описании процедуры или функции (см.диаграммы и 6.I). Основная операция над объектами этих видов - вызов, т.е. исполнение предусмотренных описанием предписаний. При вызове могут указываться параметры (входные данные). Функции, кроме того, возвращают в качестве результата объект произвольного вида.

Вызов является операцией над процедурным (функциональным) выражением; он обозначается круглыми скобками, в которых через запятую могут указываться значения параметров. примеры:

```
РАКЕТА(I,2, "ВВЕРХ")
SIN (90)
ДСЧ()
```


Основное отличие процедуры от функции определяется наличием у функции результата. Поэтому, вызов процедуры - это предписание, а вызов функции равнозначен выражению. Примеры вызовов процедур:

```
КАДР(I,2); ЛИНИЯ(I0,20,20,30);
```

Примеры использования вызовов функции в выражениях:

```
? SIN (90);
ЦЕЛЧ(I0.89)-> A;
A()+I3-> B;
```

Более подробно порядок описания процедур и функций, способы передачи параметров и особенности вызова описаны в разделе 6.

Примечание. Хотя процедуры и функции в момент описания связываются со своим именем, их можно, как и другие объекты, сделать элементами структур.

Стандартные процедуры и функции РАПИРЫ

В описываемых версиях системы есть ряд встроенных (т.е. не требующих описания и доступных в любой программе) процедур и функций. Они носят в основном сервисный характер. Описание стандартных процедур и функций приводится в Приложении 6.

3.I2. Краткие выводы

В этом разделе были описаны все формы представления данных в виде объектов языка (кроме файлов) и перечислены основные операции над данными различных видов.

Данные (объекты) в языке существуют в трех видах: исходные, задаваемые в программе в явном виде (константы), например

```
I23, I23, 456, "КРОКОДИЛ"
```

промежуточные, получающиеся в результате выполнения операций над другими данными, например:

I2+34, "ТЕКСТ"[2:3] , <I,2,3>

и именованные, т.е. являющиеся значениями имен.

Для составления правильных и эффективных программ необходимо хорошо понять, как с помощью объектов языка смоделировать реальные объекты, взаимодействие которых программируется. Например, кортежи удобны для моделирования списочных структур, тексты - для поддержания диалога с пользователем программы и т.п.

Рассмотрим теперь управляющие конструкции языка, которые, собственно, и составляют основу любой программы на РАПИРЕ.

РАЗДЕЛ IУ. ОСНОВНЫЕ ПРЕДПИСАНИЯ РАПИРЫ

4.I. Форма описания синтаксиса

В этой инструкции используется графический способ описания синтаксиса языка РАПИРА. Это значит, что синтаксис каждой конструкции изображается диаграммой, состоящей из овальных и прямоугольных полей, связанных соединительными стрелками.

Символы и слова, указанные в прямоугольном поле, должны быть указаны в соответствующем месте конструкции в том же порядке. Слова, записанные в овальных полях, обозначают название некоторой другой конструкции, которую следует целиком употребить в данном месте. Синтаксис ее, как правило, тоже описан некоторой диаграммой.

Начало диаграммы обозначается знаком Н, конец - знаком К. Слева от начала может быть указано название диаграммы.

Каждый маршрут от начала к концу диаграммы строго по направлению стрелок соответствует некоторой возможной форме описываемой конструкции.

Прилагаемые к этой инструкции диаграммы описывают пользовательский синтаксис языка РАПИРА в описываемой версии. Для синтаксического разбора используются более подробные диаграммы.

Диаграммы носят иллюстративный характер и не определяют полностью синтаксис рассматриваемых версий. Наиболее очевидные конструкции разъясняются прямо в тексте.

4.2. Общая структура предписания

Всякое предписание РАПИРЫ состоит из отдельных лексем (слов). Лексемы являются минимальными единицами языка, несущими самостоятельную смысловую нагрузку. В РАПИРЕ различаются следующие лексемы:

простые имена;

ключевые слова;

числа;

тексты;

специальные символы (простые и составные).

Длина лексемы во вводимой строке не может превышать 255 символов.

Между любыми лексемами может быть вставлено произвольное число пробелов и переводов строки; сами же лексемы разрывать нельзя. С этими правилами вы уже знакомы из раздела 3 для чисел, имен и текстов.

Если две соседние лексемы при слиянии могут быть приняты за новую лексему (например, два имени или имя и число), то между ними обязателен хотя бы один пробел или перевод строки. Если новая лексема не образуется, то разделитель не требуется (например, имя и знак операции могут быть записаны подряд).

Расположение предписаний на строках произвольно: можно располагать одно предписание на нескольких строках и несколько предписаний на одной строке.

С примерами имен, чисел, текстов и специальных символов вы уже знакомы. Ключевые слова служат для построения предписаний, придавая им вид предложений на языке, близком к естественному. Например:

```
ЕСЛИ A>1  ТО 1->A  ИНАЧЕ 2->A ВСЕ;
```

Здесь ключевые слова выделены.

Внимание! Ключевые слова ничем не выделяются и по правилам записи не отличимы от имен. Поэтому будьте осторожны с именами, которые совпадают с некоторыми ключевыми словами. Например, предписание

```
ВЫВОД->A;
```

проанализируется системой как предписание вывода, и будет выдано сообщение об ошибке в записи этого предписания ("ТРЕБУЕТСЯ ДВОЕТОЧИЕ").

4.3. Общая структура программы. Режимы работы системы. Директивы. Описание имен. Комментарии

Общая структура программы описана диаграммой I3. Как видно, программа состоит из последовательности директив, процедурных блоков и описаний имен, следующих друг за другом в произвольном порядке. После каждой из перечисленных конструкций должна стоять точка с запятой (;).

Директива – это предписание, исполняемое немедленно после ввода. Примерами директив могут служить предписания присваивания (->) и вывода (?), простейшие формы которых были описаны в 3 разделе.

В РАПИРЕ предусмотрено несколько режимов диалога, которые отличаются набором допустимых возможностей. Установить, в каком режиме система находится в данный момент, можно по верхней (информационной) строке экрана (см. Приложение I).

В основном диалоговом режиме на верхней строке высвечивается название текущей версии РАПИРЫ (например: РАПИРА-АГАТ I.I). В этом режиме допустимы следующие директивы:

присваивание (4.4),
вывод (4.8),
ввод (4.9),
вызов (3.II, раздел 6),
включить и выключить (3.5, разделы 7 и 9),
цикл (4.6),
ветвление (4.5),
каталог (3.4, 5.4),

контроль (4.5),

предписания файловой системы (раздел 8).

Эти директивы называются также универсальными предписаниями, т.к. они допустимы и в описании процедуры или функции.

Синтаксис языка допускает также пустое предписание, которое ничем не обозначается и не вызывает никаких действий. Наличие его позволяет, например, записывать несколько точек с запятой подряд и вводить строки, не содержащие предписаний.

Предусмотрены также директивы "РАПИРА" и "РОБИК". Они производят очистку памяти и перевызов системы без перезагрузки с диска. При этом оказывается доступен один из указанных языков.

В режиме редактирования, который используется при описании процедур и функций, в верхней строке экрана нет никакого заголовка. Весь текст в этом режиме выводится не голубым, а зеленым цветом. Предписания языка РАПИРА, набираемые в этом режиме, не исполняются, а только запоминаются в процессе набора. Управление редактированием ведется с помощью функциональных клавиш клавиатуры или в виде выбора из меню. Подробнее этот режим описан в 5.I.

В режиме ПАУЗА на верхней строке высвечивается слово "ПАУЗА". Система переходит в этот режим после прекращения исполнения процедуры, вызванного предписанием "СТОП" или нажатием клавиши останова. В этом режиме возможно исполнение тех же директив, что и в основном режиме. Кроме того, разрешено использование директив "ШАГ", "ВЫХОД" и "ПУСК" (7.2).

В режиме ОШИБКА на верхней строке экрана высвечивается слово "ОШИБКА". Система переходит в этот режим, если в процессе исполнения процедуры или функции возникла ошибка. Кроме директив основного режима здесь допускается использование директивы "ВЫХОД". Режимы "ПАУЗА" и "ОШИБКА" более подробно рассмотрены в 7.2.

Кроме перечисленных, в системе предусмотрено несколько графических режимов, описание которых приведено в разделе X.

Процедурный блок - это описание процедуры или функции. Порядок описания и вызова процедур описан в разделе УI.

Структура описания имен представлена на диаграмме I4: в описываемых версиях РАПИРЫ такое описание состоит из слова "ИМЕНА" и через двоеточие перечня имен, разделяемых запятыми. Примеры:

```
ИМЕНА:A,B,C:
ИМЕНА:X;
```

Обратите внимание, что заголовок этой конструкции всегда пишется во множественном числе, даже если имя одно.

Внутри процедур эта конструкция используется для объявления локальных имен. В канонической РАПИРЕ она позволяет описывать также тип имени, который проверяется затем при каждом присваивании. Однако в описываемых версиях типизация не предусмотрена, и описание имен в основном режиме эквивалентно пустой директиве , как уже упоминалось в 3.4, описывать такие имена в РАПИРЕ необязательно.

Между любыми двумя лексемами РАПИРЫ можно вставить комментарий (диаграмма 5):произвольную последовательность символов, заключенную в скобки (* *), например:

```
(* Эта программа предназначена
   для решения уравнений вида X**2+A*X+B=0 *)
```

Обычно комментарии используют в составе процедур и функций, но их можно набирать и в других режимах. Комментарий не анализируется системой и никак не влияет на исполнение программы.

Внимание! Последовательность символов "*)", встретившаяся в комментарии, считается его концом. Поэтому, в частности, вложенные комментарии недопустимы.

Если комментарий располагается на одной строке, то его длина (число символов, включая скобки) не может быть больше 255 символов; это следует иметь в виду при описании процедурных блоков. В общем случае комментарий может располагаться на нескольких строках, и его длина в этом случае неограничена.

Примечание для новичков: в комментариях обычно записывают пояснения к программе для тех, кто будет читать, использовать и модифицировать ее. Не пренебрегайте комментариями!

4.4. Уточненное имя. Выражение. Приоритеты операций. Присваивание

С простейшими примерами выражений и предписания присваивания вы уже знакомы. В этом параграфе описываются общие правила построения и вычисления выражений на РАПИРЕ и все возможные формы присваивания.

Как уже отмечалось в 3.4, имена используются для хранения в памяти машины некоторых объектов. При повторном присваивании одному и тому же имени его предыдущее значение уничтожается. Однако,

если значением имени является текст, кортеж или запись, то часто требуется менять не все значение, а только его часть (участок текста, элемент кортежа, поле записи).

Для этого в правой части предписания присваивания после имени надо указать изменяемые элементы, используя операции вырезки, выборки и доступа. Конструкция из последовательного применения этих операций называется блоком уточнения (диаграммы 8,9), а имя, после которого он указан – уточненным именем (в отличие от простого имени, введенного в 3.4).

Например, если значением имени ЗООПАРК является кортеж

```
<<¤ЗВЕРЬ:"КРОКОДИЛ", ДЛИНА:200¤>,
 <¤ЗВЕРЬ:"ТИГР",РОДИНА:"ЛИВИЯ"¤>>
```

то ЗООПАРК[1].ЗВЕРЬ – текст "КРОКОДИЛ", а присваивание

```
"ИНД"->ЗООПАРК[2].РОДИНА[1:3];
```

исправит название страны, где водятся тигры.

Правила записи выражений в РАПИРЕ описываются диаграммой II. Как следует из диаграммы, выражение – это последовательность соединенных знаками бинарных операций констант, формирователей структур, простых имен и выражений в скобках, перед каждым из которых может стоять знак унарной операции, а после – произвольная последовательность операций вырезки, выборки, доступа и вызова. Пример:

```
(<1,2>[2:2]+<A[<3+1,4>[K]]>+A)[1]*-2
```

Всякое правильное выражение вырабатывает определенное значение (это следует из того, что каждая операция в выражении вырабатывает результат).

В РАПИРЕ принят следующий порядок вычисления выражений:

I. Выражения в скобках и составляющие выражения в формирователях структур;

2. Формирование структур: <>, <**>, <¤¤>;

3. Вычисление фактических параметров функций и индексных выражений;

4. Выборка, вырезка, доступ, вызов функции: , ,

5. Унарные операции: - , #, @ ;

6. Возведение в степень: ** ;

7. Операции умножения, деления, пересечения: *,/,//;

8. Операции сложения, вычитания, конкатенации, объединения, разности множеств: +, -.

Все операции одного приоритета выполняются слева направо (следовательно, A**B**C означает (A**B)**C). Для операций с различным приоритетом это описание устанавливает лишь относительный порядок исполнения: всякая операция будет выполнена после того, как будут определены ее операнды, вычисляемые с помощью операций более высокого приоритета.

В приведенном выше примере операции будут выполнены в следующем порядке (объясните сами, почему):

I. Формирование кортежа <I,2> ;

2. Вырезка [2:2];

3. Вычисление 3+I;

4. Формирование кортежа из результата (3) и числа 4;

5. Выборка [K] ;

6. Выборка из имени A с полученным в результате (5) индексом;

7. Формирование кортежа из выборки (6);

8. Конкатенация кортежей (2) и (7);

9. Конкатенация кортежей (8) и A;

I0. Выборка [I] из (9);

II. Вычисление -2;

I2. Умножение чисел (I0) и (II).

Полученный результат будет значением выражения.

Примечания:

I. В выражениях вида 5*(-7) скобки можно опускать;

2. Обратите внимание, что диаграмма II рекурсивна, т.е. частью выражения может быть другое выражение! При построении выражений необходимо следить за тем, что результатом каждой операции должен быть объект, над которым может быть выполнена следующая по порядку операция! В противном случае будет выдано сообщение об ошибке. Например, конструкция

A[5] (X) [II] [2:3]

является допустимой, если A - это кортеж, пятым элементом которого является функция с одним параметром; результатом этой функции должен быть кортеж, одиннадцатым элементом которого, в свою очередь является текст или кортеж длиной не менее трех.

Выражения

<I,2,3>[I] [2:3]

и

A[I:5].B

недопустимы: в первом случае предпринимается попытка вырезки из числа (почему?), а во втором - операция доступа для текста или кортежа (а только они могут быть результатом вырезки) не определена.

Теперь становятся совершенно понятны общий вид и назначение предписания присваивания:

выражение -> уточненное имя;

заметим только, что простое имя является частным случаем уточненного.

Примечание для специалиста. При присваивании, формировании структуры, вырезке, выборке и доступе выражение разыменуется. Это значит, что константа и значение имени копируются. Например, после исполнения предписаний

5-> A; <A> ->B; I0->A;

значением B остается кортеж <5>. Здесь еще раз показано, что операции и предписания выполняются над объектами, а имена служат лишь формой их указания.

4.5. Ветвление. Условия. Условные предписания

Одной из самых распространенных ситуаций в программе является выполнение тех или иных предписаний в зависимости от истинности некоторых условий. Предписание, позволяющее сделать этот выбор, называется ветвлением (условным предписанием). В целом формы ветвления в РАПИРЕ традиционны для универсальных языков программирования.

Условием называется некоторое утверждение об объектах, которое может быть истинным или ложным. Например, условие "I2 больше 3" истинно, а условие "число 7 входит в множество <=I,2=>" ложно.

Синтаксис условия описан диаграммой I2. Смысл логических операций "И", "ИЛИ", "НЕ" соответствует принятому в математике. Операции сравнения ">", "<", ">=", "<=" определены только для числовых значений.

Примечание для специалиста. Логических выражений в языке нет, поэтому условие является специфической конструкцией для условных предписаний и цикла "ПОКА".

Условие "A вида B" истинно, если значения выражений A и B одного и того же вида, и ложно в противном случае (см. 3.4).

Условие "A из B" истинно, если литера A входит в текст B или произвольный объект A является элементом кортежа (множества) B.

При сравнении на равенство (=) и неравенство (/=) два значения считаются равными, если оба значения имеют один и тот же вид, причем:

тексты и кортежи совпадают поэлементно;

множества равны в математическом смысле;

записи состоят из одних и тех же полей, и значения соответствующих полей равны;

два пустых значения всегда равны;

процедуры, функции и файлы равны, если они получены из одного и того же значения путем некоторых операций.

Совпадение элементов множеств и кортежей означает, что они равны в описанном выше смысле.

Примечание для специалиста. Фактически для процедур, функций и файлов сравниваются ссылки на значение, т.к. при их присваивании копируются ссылки, а не сами значения.

Порядок исполнения операций в условиях следующий:

1. Вычисление выражений в операциях сравнения (включая "ИЗ" и "ВИДА");

2. Сравнение = , /=, <, > , >= , <= , ВИДА, ИЗ;

3. Отрицание "НЕ";

4. Конъюнкция "И";

5. Дизъюнкция "ИЛИ".

Примеры истинных условий:

```
"КРОКОДИЛ"/="БЕГЕМОТ"
<1,2,3>/=<3,2,1>
<#1,2,3#>=<#3,2,1#>
<¤ИМЯ:"ВАСЯ"¤>/=<¤ИМЯ:"ПЕТЯ"¤>
<¤A:127¤>/=<¤B:127¤>
2#2=4.0
"ВАГОН"[4:5]="СЛОН"[3:4]
12>-34 ИЛИ НЕ 0>=-13.7
НЕ "Я" из "ПРИМЕР" И <1> ИЗ <#2, <1>#>
<>ВИДА <0,1.0>
```

В РАПИРЕ есть два вида ветвления (диаграмма 18).

Предписание "ЕСЛИ" выполняется следующим образом. Вначале проверяется условие, записанное между ключевыми словами "ЕСЛИ" и "ТО". Если оно истинно, выполняются предписания, указанные между "ТО" и "ИНАЧЕ" ("ТО" и "ВСЕ", если вариант "ИНАЧЕ" отсутствует). В противном случае выполняются предписания, расположенные между "ИНАЧЕ" и "ВСЕ" (или не выполняется ничего, если вариант "ИНАЧЕ" не предусмотрен).

Пример:

```
ЕСЛИ A>B  ТО
   5->A; ? B
ИНАЧЕ
   5->B; ? A
ВСЕ;
```

Другое предписание - "ВЫБОР" - позволяет выполнить те или иные предписания в зависимости от нескольких условий. Оно представляет собой лишь более наглядный способ проверки цепочки условий и позволяет не писать сложную последовательность вложенных предписаний "ЕСЛИ" (попробуйте сделать это в случае пяти условий).

Первый вариант этого ветвления позволяет просто проверить набор условий. Как видно из диаграммы, предписание состоит из произвольного числа альтернатив, каждая из которых представлена условием, за которым через двоеточие следует список предписаний. Альтернативы разделяются знаком "!". Допустима конструкция "ИНАЧЕ".

Проверка условий происходит последовательно, в порядке записи альтернатив. Как только одно из них оказывается истинным, выполняются предписания, записанные после него через двоеточие. После этого выполнение предписания заканчивается.

Если ни одно условие не было истинным, выполняются предписания, записанные между "ИНАЧЕ" и "ВСЕ" (если эта часть предусмотрена). Таким образом, в любом случае выполняется не более одной из предусмотренных альтернатив.

Условия в альтернативах могут быть любой степени сложности.

Пример:

```
ВЫБОР ИЗ
   ВЕТЕР<=3: ?"СЛАБО" !
   ВЕТЕР<=7: ?"УМЕРЕННО" !
   ВЕТЕР<=70: ?"ТРЕВОГА!"
ИНАЧЕ
   ?"НЕВЕРНО ЗАДАНА СКОРОСТЬ ВЕТРА"
ВСЕ;
```

Второй вариант выбора исполняется в том же порядке, но вместо проверки условий значение выражения, указанного перед словом "ИЗ" (параметр выбора) сравнивается поочередно со значениями выражений, стоящими перед альтернативами. Таких выражений может быть несколько (через запятую). В случае равенства выполняется соответствующая альтернатива. Значение параметра вычисляется один раз в начале выполнения предписания.

Пример:

```
ВЫБОР ОЦЕНКА ИЗ
        5:?"ОТЛИЧНО" !
        4:?"ХОРОШО" !
        3:?"УДОВЛЕТВОРИТЕЛЬНО" !
        1,2:?"ПЛОХО"
   ИНАЧЕ ?"НЕПОНЯТНО" ВСЕ;
```

Обратите внимание, что ветвления - составные предписания, т.е. содержат другие предписания, которые также могут быть составными. Концом списка вложенных предписаний считаются слова "ВСЕ", "ИНАЧЕ" и знак "!". Если одно из них пропущено, все дальнейшие предписания будут считаться частью все того же ветвления. Поэтому,

при наборе предписания в любом диалоговом режиме следует следить за приглашением ко вводу: если после окончания набора вложенных составных предписаний, точки с запятой и перевода строки вам выдается ">", значит вы пропустили одно из "закрывающих" слов.

Во избежание путаницы советуем оформлять составные предписания так, как в примерах выше: начинать каждое из них с новой строки, а каждое вложенное сдвигать на несколько позиций от начала строки. Так вам будет легче читать и исправлять программу, проще искать ошибки.

К условным относится также предписание "КОНТРОЛЬ" (диаграмма 20). Оно используется для проверки истинности некоторого условия во время исполнения программы. Если условие истинно, это предписание равносильно пустому, в противном случае выдается сообщение об ошибке: "СРАБОТАЛ КОНТРОЛЬ".

4.6. Циклы

Для организации многократного выполнения последовательности предписаний в РАПИРЕ предусмотрены циклические предписания или циклы.

Синтаксис трех основных циклов описывается диаграммой I3.

Цикл "ПОКА" имеет традиционную семантику: вначале проверяется условие, указанное в заголовке цикла. Если оно истинно, выполняются все предписания от знака:: (читается "повторять") до слова "ВСЕ". Затем снова проверяется условие и т.д. Если при очередной проверке условие оказалось ложным, выполнение цикла прекращается. Если оно

было ложным при первой же проверке, тело цикла не выполнится ни разу.

Таким образом, этот цикл позволяет выполнять определенные действия, пока истинно некоторое условие. Пример:

```
I->X;
ПОКА X<=30 :: ?X,2пбX; X+I->X ВСЕ;
```

Введите эту программу в машину и посмотрите на результат.

Цикл "ПОВТОР" позволяет выполнить указанный набор предписаний заданное число раз. Он выполняется следующим образом. Вначале вычисляется значение выражения в заголовке цикла. Оно должно быть целым неотрицательным числом, иначе выдается сообщение об ошибке. Затем тело цикла выполняется указанное число раз. В случае нулевого или отрицательного числа проходов тело цикла не выполняется ни разу.

Пример:

```
I->X;
ПОВТОР 30 РАЗ :: ?X,2пбX; X+I->X ВСЕ;
```

Легко видеть, что этот цикл выполняет те же действия, что и предыдущий, но несмотря на явно заданное число проходов нам все равно пришлось ввести имя-счетчик.

Варианты "РАЗ" и "РАЗА" эквивалентны, служат для удобства чтения программы и могут быть опущены.

Цикл "ДЛЯ" (цикл перебора) существует в двух формах. Форма "ДЛЯ-ОТ" соответствует традиционному для большинства языков циклу типа прогрессии. Имя, указанное после слова "ДЛЯ", называется переменной цикла, выражение после слова "ОТ" определяет начальное значение, выражение! после слова "ДО" - конечное, после слова

"ШАГ" указывается приращение. Если шаг не указан, он по умолчанию принимается равным единице. Значения всех трех выражений должны быть целыми или дробными числами.

Цикл выполняется в таком порядке.

1. Вычисляются все три указанные в заголовке выражения.

2. Переменной цикла присваивается начальное значение (с выдачей запроса, если она имела частичную защиту).

3. Проверяется условие окончания цикла: текущее значение переменной цикла больше конечного значения при положительном шаге или меньше его при отрицательном (или (КЗ-П) X sign (Ш) >= 0, где КЗ,П;Ш - конечное значение, переменная цикла и шаг соответственно; знак нуля считаем единицей).

4. Если оно ложно, выполняется тело цикла, затем к переменной цикла прибавляется значение шага, вновь проверяется условие и т.п.

5. Если при очередной проверке условие оказывается истинным, цикл завершается. При этом переменная цикла получает пустое значение.

Данный цикл наиболее естественным образом задает действия, которые мы специально организовывали в предыдущих примерах:

```
ДЛЯ X ОТ I ДО 30 :: ?X,2пбX ВСЕ;
```

Цикл "ДЛЯ-ИЗ" предназначен для поэлементной обработки текстов, кортежей и множеств. При выполнении этого предписания переменная цикла поочередно принимает значения всех элементов структуры (каждой литеры текста). Для текстов и кортежей порядок перебора определен: от первого элемента к последнему. Элементы множества перебираются в произвольном порядке.

После завершения цикла переменная также получает пустое значение. Значение выражения, указанного в заголовке (перебираемая структура), вычисляется один раз в начале работы цикла.

Пример - подсчет числа букв "А" в тексте с именем КНИГА:

```
0->СЧ;
ДЛЯ БУКВА ИЗ КНИГА ::
    ЕСЛИ БУКВА="А" ТО СЧ+1->СЧ ВСЕ
ВСЕ;
?" В ТЕКСТЕ КНИГА", СЧ, "БУКВ 'А'";
```

Особо отметим, что имя, используемое в качестве переменной в цикле "ДЛЯ" (в обеих формах), при входе в цикл не должно иметь полной или системной защиты.

Цикл "ДЛЯ-ИЗ" позволяет перебрать все элементы структуры или текста, но не позволяет изменять эти элементы. Кроме того, он является единственным предусмотренным в РАПИРЕ средством перебора всех элементов множества.

Если требуется перебрать все элементы кортежа или текста и попутно уметь изменять некоторые из них, следует использовать цикл "ДЛЯ-ОТ" и выборку, например:

```
ДЛЯ Х ОТ 1 ДО #КОРТ ::
    ЕСЛИ КОРТ[Х]<0 ТО
      -КОРТ[Х] -> КОРТ[Х]
    ВСЕ
ВСЕ;
```


Во время исполнения циклов возможно так называемое зацикливание, т.е. невыполнение условия окончания цикла. Причина этого разнообразны, чаще всего они лежат в неверной организации цикла в программе. Приведем наиболее выразительные примеры:

1. ДЛЯ Х ОТ 1 ДО 1 + 1Е-11 ШАГ 1Е-12:: ... ВСЕ;

Хотя цикл должен выполниться всего одиннадцать раз, произойдет зацикливание, потому что

```
1. 00000000000 +
0. 000000000001 =
1. 000000000001
```

, и после отсекания двенадцати значащих цифр (они выделены) очередное значение переменной цикла окажется точно таким же, как до приращения! Приращение переменной цикла фактически оказалось нулем. Поэтому будьте осторожны с дробными числами - описанная ситуация может возникнуть в любом месте.

2. 1->Х;
   ПОКА Х<=30 :: ?Х,Х:=2 ВСЕ;

совершенно очевидно, что условие цикла всегда будет истинным, потому что в цикле Х не получает приращения. Следите за тем, чтобы выражения, входящие в условие цикла "ПОКА", изменялись в теле цикла, иначе зацикливание гарантировано. Кроме того, эти изменения должны в определенный момент приводить к нарушению условия. Например, программа

```
1->Х;
ПОКА Х/=0 :: -Х->Х ВСЕ;
```

тоже зациклится.

3. Цикл "ДЛЯ-ИЗ" зациклиться не может, т.к. перебираемая структура имеет конечную мощность.

4. Цикл "ПОВТОР" зациклиться не может, потому что число проходов - тоже конечное число. Следите, однако, за тем, чтобы оно оказалось слишком большим: даже миллион проходов - это уже слишком долго.

Чтобы прервать исполнение программы, когда зацикливание станет очевидным, надо нажать функциональную клавишу "F1" на функциональной клавиатуре ЭВМ "АГАТ".

При этом на экран выдается сообщение "СТОП", цикл принудительно завершается (переменная цикла "ДЛЯ" сохраняет свое последнее значение), после чего происходит выход в режим "ПАУЗА" (если остановлена процедура или функция).

В заключение заметим, что все циклы также являются составными предписаниями.

4.7. Реализация ограничения побочных эффектов в РАПИРЕ

Начинающие пользователи могут при первом чтении пропустить этот параграф.

Описанные в предыдущих параграфах условные и циклические предписания в сочетании с мощным аппаратом выражений являются плодородной почвой для возникновения всевозможных побочных эффектов, т.е. выполнения некоторого предписания не так, как требует его запись.

Типичным примером является присваивание значений переменной цикла внутри цикла:


```
ДЛЯ X ОТ I ДО II ::
        ?A[Ф(X)] ;
        ЕСЛИ X==2*40 ТО I2->X ВСЕ;
ВСЕ;
```

Здесь приведена попытка сделать досрочный выход из цикла. Не совсем ясно и значение выражения A[Ф(X)] , если функция Ф меняет значение имени A.

В РАПИРЕ принят ряд правил, позволяющих ограничить возникновение побочных эффектов в нежелательных ситуациях, предотвратить появление в программе труднонаходимых ошибок. Знание их облегчает выбор той или иной конструкции при написании программ.

Основным принципом, объединяющим все эти правила, является максимальное соответствие исполнения предписания его записи. Так, в примере выше заголовок цикла строго определяет все значения, принимаемые переменной цикла во время его работы. Любое воздействие на нее с целью нарушить этот порядок считается недопустимым.

Полезно знать основные принципы ограничения побочных эффектов в РАПИРЕ.

Разыменование (извлечение значения) выражения позволяет сделать исполнение составного предписания независимым от возможного изменения в его теле переменных, участвующих в этом выражении. Поясним это на примерах:

```
5->A;
ПОВТОР A РАЗ:: A+I->A; ВСЕ;
```

Число проходов цикла вычислится один раз в начале его работ Изменение переменной А внутри цикла не влияет на уже определенно число проходов. Значение А после выполнения этого цикла станет равным I0.

ДЛЯ А ОТ I ДО В+2 ШАГ С :: А+В->В; С-А->С ВСЕ;

Опять-таки, конечное значение и шаг вычислятся один раз в начале работы цикла и от значений, принимаемых именами В и С в цикле не будут зависеть.

Кроме описанных двух случаев разыменуется параметр предписания выбора по значению (см. 4.5). Здесь неясность возникает, если в составе одного из альтернативных выражений вызывается функция, меняющая значение этого параметра.

Второй прием - абсолютная защита имен - используется там, где невозможно разыменование. Принцип абсолютной защиты уже был вкратце описан в 3.5.

Абсолютная защита устанавливается:

на переменные циклов "ДЛЯ" (обе формы),

на перебираемую структуру в цикле "ДЛЯ-ИЗ" (разыменование здесь не годится, т.к. для разыменования большой структуры может не хватить памяти),

на имя структуры во время вычисления индексов,

на имя процедуры или функции во время вызова.

В общем случае защита устанавливается на имена, участвующие в операциях, на время вычисления других операндов. Например, в выражении А+Ф() функция Ф не может менять значения А (т.к. оно уже считается вычисленным), а в выражении Ф()+А - может, т.к. значение

А к моменту вызова Ф еще не вычислено (вспомните порядок вычисления выражений).

Когда все операнды вычислены, и операция или предписание исполнены, восстанавливается та защита имени, которую оно имело до установки системной защиты.

4.8. Вывод. Формы вывода объектов

Вывод играет ключевую роль в программе, поскольку именно он обеспечивает выдачу промежуточных и итоговых результатов программы, оформление выходных данных, диалог программы с пользователем и т.п.

Предписание "ВЫВОД" в РАПИРЕ предназначено для организации посимвольного вывода объектов РАПИРЫ на экран телемонитора, бумагу и другие устройства (см.диаграмму 25).

Простейшая форма вывода:

ВЫВОД: список элементов вывода;

или (что равносильно)

? список элементов вывода;

вам уже знакома. В списке элементов перечисляются объекты, которые требуется вывести, с указанием формы вывода. Во всех предыдущих примерах этот список состоял из выражений, разделяемых запятыми.

При выводе каждый объект, указанный в списке вывода, преобразуется в последовательность символов, которая и выдается на указанное или взятое по умолчанию устройство.

Как вы уже знаете, объекты, входящие в список вывода, выдаются на одной строке вплотную друг к другу. Переход на следующую строку выполняется только в том случае, если выводимые данные не поместились на одной строке, и после вывода всего списка объектов. Если в заголовке предписания указано слово "БПС", этот последний перевод строки также не выполняется. Вывод пустого списка равнозначен переводу строки. Например:

```
ВЫВОД:;
?;
```

При использовании этих конструкций для перевода строки рекомендуется ставить точку с запятой, иначе в примере

```
ЕСЛИ ДЛИНА  79 ТО
    ВЫВОД:;
ВСЕ;
```

ее отсутствие приведет к ошибке: будет сделана попытка вывода значения имени "ВСЕ" (почему?).

Можно указать в заголовке предписания, на какое устройство следует выдать перечисленные объекты (см. диаграмму), например:

```
ВЫВОД НА БУМАГУ: "2 х 2 = ",2**2;
ВЫВОД НА ЭКРАН БПС: "ВВЕДИТЕ СТРОКУ";
```

Особенности вывода в файл рассмотрены в 8.4.

Более мощный аппарат средств управления выводом рассмотрен в разделе 9.

Формы вывода различных объектов.

I. Самой распространенной формой вывода является естественный формат. В нем по умолчанию выводится любой объект языка. Для этого

объект нужно просто указать в списке вывода. С этой формой мы имели дело во всех предыдущих примерах.

Рассмотрим подробнее правила представления объектов в естественном формате.

а) Тексты выводятся в виде цепочки символов без кавычек. Число занимаемых позиций вывода (длина поля вывода) равна мощности текста. Внутренние кавычки текста не дублируются. Примеры:

```
ВЫВОД: "ЗНАКИ "" НЕ", "ДУБЛИРУЮТСЯ";
```

будет выдана строка:

```
ЗНАКИ " НЕДУБЛИРУЮТСЯ
```

б) Целые числа выводятся как последовательность цифр с одним пробелом впереди (чтобы отделить число от других, возможно стоящих рядом, чисел). Перед отрицательным числом выводится знак "-". Пример:

```
ВЫВОД: 2**10,7,-456789;
```

будет выдана строка

```
1024 7 -456789
```

в) Дробные числа из диапазона [0.1 ; 1.0E13] (по модулю) выдаются в форме с фиксированной точкой, например:

```
75.8736
-12712.0
0.0
```

Дробные числа, порядок которых отрицателен или больше 12, выдаются в нормализованной экспоненциальной форме, т.е. в виде мантиссы, лежащей в диапазоне [0.1;1.0), и десятичного порядка, например:

```
0.2453634E-01
-0.9227E126
```

Незначащие нули в обоих случаях подавляются, если они не примыкают непосредственно к точке.

Г) Кортежи, множества и записи выводятся в соответствующих скобках: < >, <ж ж>, <¤ ¤> ; их элементы разделяются запятыми. Внутри составных структур числа изображаются без предшествующих пробелов, а тексты в кавычках, причем каждая внутренняя кавычка дублируется. В записях, кроме того выдаются имена полей. Пример:

<¤НАЗВАНИЕ: """ЖИГУЛИ""", РАЗМЕРЫ:<I45,234,89>¤>

Таким образом изображение чисел и структур при выводе соответствует их записи в языке.

Д) При попытке вывода объектов, не имеющих символьного представления (пустых значений, процедур, функций, файлов) выдаются названия этих объектов, окаймленные точками, например:

"ПУСТО., .ПРОЦЕДУРА., .ФАЙЛ.

2. После каждого выражения в списке может через двоеточие следовать формат (см. диаграмму 25а), например:

ВЫВОД: A:I0, B:I5, C:20:A-3 ;

Формат определяет еще два способа вывода данных.

Если в формате указано одно выражение, то оно определяет число позиций, отведенное для изображения объекта. Этот формат вывода называется позиционным. В позиционном формате выводятся только тексты и числа. Значение форматного выражения должно быть целое число от 0 до 255.

Правила вывода в позиционном формате таковы.

а) Объект представляется в естественном формате (будем называть длину такого представления объекта просто длиной).

б) Если формат нулевой, происходит вывод в естественном формате.

в) Если формат больше длины объекта, то
тексты дополняются пробелами справа,
числа дополняются пробелами слева

г) Если формат меньше длины объекта, то
тексты обрезаются справа до требуемой длины,
дробные числа сокращаются за счет младших разрядов дробной части (если это невозможно – выводятся в естественном формате),
целые числа выводятся в естественном формате

Формат	Целое	Дробное	Дробное	Текст
4	I2345ж	I2.3ж	0.I234EI2	ШКОЛ*
5	I2345ж	I2.34ж	0.I234EI2	ШКОЛ*
6	I2345ж	I2.345ж	0.IEI2ж	ШКОЛЬ*
7	I2345ж	I2.345ж	0.I2EI2ж	ШКОЛЬН*
8	I2345ж	I2.345ж	0.I3EI2ж	ШКОЛЬНИ*
9	I2345	I2.345ж	0.I234EI2ж	ШКОЛЬНИК*
I0	I2345	I2.345	0.I234EI2	ШКОЛЬНИК **

Здесь звездочкой отмечен конец поля вывода.

Позиционный формат позволяет организовывать таблицы, в которых поле под каждое число или текст строго фиксировано. При выводе целых чисел в колонку по одному и тому же формату соответствующие разряды чисел оказываются друг под другом.

Еще одна полезная возможность: если требуется выдать переменное число пробелов, не обязательно организовывать цикл; можно выдать по соответствующему формату пустой текст.

3. Соблюдения разрядности при выдаче таблиц, которого легко достичь для целых чисел позиционным форматом вывода, можно добиться и для дробных. Для этого служит фиксирующий формат. Он задается указанием в списке вывода двух форматных выражений (через двоеточие после выводимого объекта). Оба формата также должны быть целыми числами от 0 до 255.

Первый формат снова обозначает общую длину поля вывода, второй указывает номер позиции этого поля, в которой должна стоять точка. Форматтирование числа происходит почти так же, как и при позиционном выводе: естественное представление числа "накладывается" на поле вывода так, чтобы точка оказалась в требуемой позиции. Если остались незанятые позиции, они заполняются пробелами. Если дробная часть не помещается в отведенную ей часть поля, она обрезается за счет младших цифр, пока это возможно. В других случаях число выдается в естественном формате (например, если слишком велика целая часть и т.п.).

Если второй формат равен нулю, число выводится в позиционном формате.

Пример: (Пусть ТАБ - кортеж, состоящий из 7 вещественных чисел)
~~ввод~~ ДЛЯ Х ИЗ ТАБ :: ВЫВОД:Х:I2:4,"ж" ВСЕ;

```
  0.I2345E34ж
 -I2.5756   ж
  -0.8EI24  ж
 823.2      ж
  I27I8.28ж      /здесь произошло нарушение формата/
  0.I003E-I2ж
  0.0       ж
```


Другие объекты в фиксирующем формате не выводятся, при попытке сделать это будет выдано сообщение об ошибке.

Внимание! Указывать форматные выражения в элементах выводимых структур нельзя. Поэтому числа и тексты в составе структур выводятся только в естественном формате.

4.9. Ввод текстов и данных с клавиатуры

Основным средством организации диалога из программы является предписание "ВВОД" (диаграмма 24). Рассмотрим основные формы этого предписания и порядок ввода.

Простейшая форма ввода:

ВВОД ТЕКСТОВ: список уточненных имен;

Слово "ТЕКСТОВ" можно опустить. Уточненные (в том числе и простые) имена в списке разделяются запятыми.

Выполнение этого предписания начинается с появления на экране приглашения к вводу "?" и курсора. Система переходит в режим ввода. Последовательность символов, набранная с момента появления приглашения до нажатия клавиши перевода строки, будет воспринята как текст и присвоена очередному имени (или части значения имени) из списка ввода. После этого будет выдано новое приглашение и т.д., до тех пор, пока список ввода не будет исчерпан. После этого продолжается исполнение программы.

Во время ввода можно пользоваться всеми управляющими клавишами клавиатуры, как и во время набора предписания. В частности, можно исправлять ошибки в строке до ввода ее в машину. Текст, вводимый в режиме ввода текстов, не нуждается в обрамляющих кавычках и в дублировании внутренних кавычек.

Пример:

```
ВВОД ТЕКСТОВ: A,B,C [2] ; ВЫВОД: A,B,C [2];
?ПРИМЕР ТЕКСТА
?      /введется пустой текст/
?"НУ,ПОГОДИ!"
 ПРИМЕР ТЕКСТА"НУ,ПОГОДИ!"
```

В этом примере C должно быть либо кортежем, не менее чем из двух элементов, либо текстом, но тогда допустим ввод только одной литеры (почему?).

Вводимый текст не может быть длиннее 255 литер.

Вводить из программы можно не только тексты, но и некоторые объекты РАШИРЫ. Для этого служит такая форма предписания ввода:

ВВОД ДАННЫХ: список уточненных имен ;

Выполнение этого предписания также начинается с выдачи на экран приглашения "?". Объекты записываются в вводимой строке по общим правилам записи их в программе: тексты в кавычках (внутренние кавычки дублируются), кортежи и множества – в своих скобках с соблюдением правил записи элементов. Записи, процедуры, функции, файлы и пустые значения вводить нельзя.

Объекты во вводимой строке разделяются пробелами, переводами строк или запятыми. На них распространяются все правила записи лексем. Это значит, что между двумя любыми лексемами можно вставить произвольное число пробелов, переводов строк и даже комментариев. Можно располагать несколько объектов на одной строке. При переходе на новую строку приглашение автоматически дублируется.

Число вводимых объектов должно соответствовать числу имен в списке. Если их меньше, приглашения будут выдаваться до тех пор, пока не будут набраны недостающие объекты. Если их больше, то лишние игнорируются. По мере ввода объекты присваиваются уточненным именам из списка ввода в порядке их записи. При этом соблюдаются все ограничения, связанные с защитой имен и проверкой размерности структур, как и при обычном присваивании.

При ошибке в записи объекта выдается сообщение "ОШИБКА: ПОВТОРИТЕ ВВОД" и указывается место, начиная с которого надо продолжить ввод.

Пример:

```
ВВОД ДАННЫХ:А,Б,В,Г;
?"ТЕКСТ", <I,2;3,4>
"ТЕКСТ", <I,2
ОШИБКА: ПОВТОРИТЕ ВВОД
 ,3,4> - I2.I2I3E45
? (=ВВОДИМ ПОСЛЕДНИЙ ОБ'ЕКТ=) 0
```

Будут введены объекты: текст "ТЕКСТ", кортеж <I,2,3,4>, числа -I2.I2I3E45 и 0.

Обратите внимание, что после выдачи сообщения об ошибке при вводе кортежа, ввод продолжен с запятой. Чтобы запомнить, с чего надо продолжать ввод, можно считать, что ошибочная лексема во вводимой строке равнозначна переводу строки.

Можно отказаться от ввода текстов или данных и прервать исполнение программы (например, если предыдущие данные были введены неправильно). Для этого вместо очередной строки или лексемы следует вставить во вводимую строку символ останова

"FI" и перевести строку. Это будет воспринято, как ошибка, и выполнение программы прервется.

Примечания:

1. При вводе числа разрешается указывать перед ним знак "-" (формально, он является самостоятельной лексемой и обозначает унарную операцию).

2. Хотя предписание "ВВОД" может быть исполнено и в основном диалоговом режиме, чаще всего оно используется в процедурах для организации диалога с пользователем.

3. Системное приглашение ко вводу "?" можно заменить любым другим символом, используя стандартную процедуру "ПРИГЛ" (см. Приложение 6).

Рекомендуем поэкспериментировать с предписаниями ввода и вывода, проверить еще раз все описанные возможности.

4.10. Основные выводы

В этом разделе были описаны основные конструкции РАПИРЫ, присутствующие во многих алгоритмических языках: присваивание, ветвления, циклы, ввод и вывод. В принципе, они позволяют запрограммировать любой алгоритм, включая организацию ввода исходных данных и вывод результатов.

Сейчас вам уже известны почти все способы обработки данных и минимальные средства построения программ. Эти программы вы можете исполнять в основном диалоговом режиме работы системы.

Однако, этих средств оказывается достаточно только для однократного решения сравнительно небольших задач. Если в программе окажется ошибка, то для ее исправления придется заново набирать всю программу.

Для облегчения работы пользователя в языке и системе предусмотрены дополнительные возможности и отладочные средства, описанию которых посвящены следующие разделы.

РАЗДЕЛ 5. ОБЩИЕ СВЕДЕНИЯ О ПОДГОТОВКЕ И ХРАНЕНИИ ПРОГРАММ

5.1. Подготовка рабочих дисков

В ходе дальнейшего знакомства с системой "ШКОЛЬНИЦА" вам потребуется рабочий диск, на котором вы будете хранить свои программы.

Ознакомьтесь с порядком его подготовки.

1. Вставьте в дисковод диск №1 и произведите запуск системы. Напомним, что для этого достаточно выключить и заново включить машину.

2. На запрос в главном системном меню нажмите клавишу, обозначающую программу разметки рабочих дисков. После загрузки с диска в верхней строке экрана появится надпись

РАЗМЕТКА РАБОЧИХ ДИСКОВ

3. Вставьте в дисковод тот диск, который вы будете использовать в качестве рабочей библиотеки программ. Проследите, чтобы он имел прорезь, иначе разметка будет невозможна.

4. Далее программа запросит у вас заголовок библиотеки – произвольный текст, который будет выдаваться каждый раз при выдаче каталога перед списком программ на диске. Позиция последнего воспринимаемого символа заголовка отмечается на экране красной звездочкой.

5. Если разметка по какой-либо причине не прошла (испорчен диск или неисправен дисковод), выдается сообщение

СБОЙ РАЗМЕТКИ

6. После разметки диска выдается запрос

ХОТИТЕ РАЗМЕТИТЬ ДРУГИЕ ДИСКИ (Д/Н)?

При ответе "Д" весь описанный процесс повторяется заново, при ответе "Н" происходит возврат в общее системное меню.

Рабочий диск отличается от системных дисков Ш1 и Ш2 тем, что почти все пространство на нем отведено для хранения программ и данных пользователя. Если при включении машины в дисководе находится рабочий диск, на экран выдается сообщение:

ЗАПУСК С ЭТОГО ДИСКА НЕВОЗМОЖЕН!

ВСТАВЬТЕ СИСТЕМНЫЙ ДИСК И НАЖМИТЕ КЛАВИШУ "СБРОС"

для того, чтобы произвести запуск системы "ШКОЛЬНИЦА".

Рабочие диски "ШКОЛЬНИЦЫ" полностью совместимы со стандартными дисками, инициализированными штатной дисковой операционной системой DOS 3.3 ЭВМ "АГАТ", поэтому их тоже можно использовать, как рабочие.

5.2. Общие сведения о подготовке и редактировании программных текстов

Для того, чтобы сохранить программу на рабочем диске, её необходимо оформить, как процедуру. Вызовите интерпретатор РАПИРЫ, вставьте в дисковод рабочий диск и наберите на клавиатуре строку

ПРОЦ ПРИМЕР;

Покрутив диск, система очищает экран и выдает сверху зеленными символами:

I ПРОЦ ПРИМЕР;

2

и белый курсор. В нижней части экрана появляется сообщение

ФАЙЛ НЕ НАЙДЕН

показывающее, что программы с именем ПРИМЕР еще нет в библиотеке на диске. Вы находитесь в Редакторе программных текстов системы, а точнее, в режиме редактирования.

Здесь вы можете набирать любой текст, в том числе - программу. Набираемый текст не анализируется, а только запоминается в ОЗУ. Особенности этого режима от основного диалогового режима заключаются в следующем.

I. Экран рассматривается как окно, через которое мы видим некоторую часть редактируемого текста; курсор всегда указывает на текущий редактируемый символ.

2. Текст выдается на экран зеленым цветом, а входящие в его состав управляющие символы - инверсным желтым; голубыми выдаются символы, имеющие коды строчных букв, хотя ввод их в Редакторе не предусмотрен.

3. Редактируемый текст разбит на строки, каждая из которых может занимать несколько экранных строчек (ограничений на их число нет).

4. Первые пять позиций экранной строчки отводятся под номер строки текста.

5. Способ редактирования - экранный, т.е. всякое изменение текста на экране с помощью управляющих клавиш влечет изменение текста в памяти.

6. Движение по тексту (сопровождаемое, согласно I, движением курсора по экрану) осуществляется с помощью четырех клавиш со стрелками. При попытке выхода за границы экрана вверх или вниз окно сдвигается так, чтобы показать требуемую часть текста. При невозможности сдвига подается звуковой сигнал. Если курсор выходит за границы текста (например, выход за конец строки), его цвет меняется на фиолетовый.

7. Для образования новой строки текста после текущей служит клавиша перевода строки.

8. Чтобы заменить текущий символ, надо просто нажать любой другой.

9. Для уничтожения текущего символа следует нажать клавишу I (Здесь и ниже имеются в виду функциональные клавиши, расположенные справа на клавиатуре ЭВМ "АГАТ"). При этом следующая за ним часть текста сдвигается влево, заполняя пустое место.

I0. Для вставки нескольких символов надо нажать клавишу 2. После этого все набираемые символы вставятся между текущим и предыдущим символами. Текст сдвигается вправо, освобождая место для вставки. Чтобы закончить вставку, следует нажать любую редактирующую клавишу (например, ->).

II. Для перехода к началу следующей строки текста, служит клавиша 3.

I2. Для перехода к предыдущей и следующей страницам текста (участкам текста, помещающимся в окне), служат клавиши 4 и 5, соответственно. С их помощью можно быстро просмотреть весь редактируемый текст.

I3. Клавиша 6 сдвигает окно так, чтобы текущая строчка оказалась настолько близко к середине экрана, насколько это возможно. Она позволяет посмотреть ближайшие окрестности редактируемой строки.

I4. Клавиша 7 "склеивает" текущую строку текста со следующей. При этом на месте склейки вставляется пробел.

I5. При нажатии клавиши 8 отсекается часть строки текста, расположенная после текущего символа, включительно; курсор при этом остается на месте.

I6. Чтобы уничтожить всю текущую строку, надо нажать клавишу 9. Курсор при этом оказывается на следующей строке, а если её нет - на предыдущей.

I7. Чтобы ввести в текст один управляющий символ, надо нажать "УПР-*V*", а затем соответствующую ему клавишу. В частности, вставка символа перевода строки позволяет разбить строку текста на две части.

I8. Клавиша *F* 2 переключает режим изображения текста на экране: из 32x32 цветных символов в 64x32 белых символов на черном фоне и наоборот. Последний режим удобнее для работы, если телемонитор - черно-белого изображения.

Таким образом, в режиме редактирования предусмотрены все средства для эффективного редактирования произвольных текстов. Ориентированность его на программные тексты заключается в том, что при вставке новой строки в её начало автоматически вставляется столько пробелов, сколько их было в начале предыдущей строки (но не более одной строчки экрана). Это облегчает запись программы "лесенкой".

Клавиша "РЕД" позволяет перейти из режима редактирования в т.н. режим меню и обратно.

В режиме меню на экране перечислены названия некоторых дополнительных действий (директив). Вид меню приведен в Приложении I. Двигаясь по меню при помощи клавиш-стрелок, можно выбрать одну из предложенных директив (она выделяется на экране инверсом) и исполнить её, нажав клавишу перевода строки.

Нажатие клавиши "РЕД" при выборе директивы вызывает возврат в режим редактирования.

При исполнении директив возможны некоторые дополнительные запросы. Набрав запрошенное число или текст, следует перевести строку. Пустая посылка (т.е. просто нажатие клавиши перевода строки) обозначает выбор по умолчанию (если это номер строки, то он указывается в скобках при запросе).

Например:

НОМЕР СТРОКИ (43) –

Любой запрос можно прервать нажатием F1. Это означает отказ от выполнения директивы.

По директиве "ПЕРЕЙТИ К СТРОКЕ" курсор устанавливается на начало указанной строки.

По директиве "ЗАПОМНИТЬ ТЕКСТ" запрашивается и запоминается произвольная цепочка символов. Если потом, находясь в режиме редактирования, нажать F3, будет выполнен поиск первого после текущего символа вхождения этой цепочки в редактируемый текст. Если такое вхождение не обнаружено, курсор устанавливается в конец текста. Этот прием позволяет найти все вхождения некоторой последовательности символов (например, для замены). Символ "CTRL-A" в цепочке означает, что во время поиска в данной позиции может встретиться произвольный символ.

Директива "ВЫДАТЬ НА БУМАГУ" позволяет выдать участок текста на бумагу. Он определяется номерами первой и последней строк.

Директива "ОЧИСТИТЬ БУФЕР" уничтожает весь находящийся в памяти Редактора текст.

Директива "ВЫЙТИ" позволяет вернуться в основной диалоговый режим без анализа редактируемого текста.

По директиве "КОНЕЦ ОПИСАНИЯ" редактируемый текст анализируется интерпретатором РАПИРЫ как описание процедуры или функции (см. 6.3). Если синтаксических ошибок не обнаружено, происходит выход в основной диалоговый режим. В противном случае, система возвращается в режим редактирования, причем на нижней строке появляется сообщение об ошибке, а курсор устанавливается на начало ошибочной лексемы.

Директивы в правом столбике меню позволяют организовать из Редактора работу с ДЗУ: запомнить редактируемый текст на диск под некоторым именем (т.е. образовать на диске файл с этим именем, содержащий данный текст), считать с диска файл, уничтожить файл на диске, дописать в конец текущего текста содержимое некоторого файла, просмотреть каталог всех файлов, переключиться с одного ДЗУ на другое.

В меню указывается также имя текущего файла, с которым идет работа в настоящий момент. Если на запрос имени файла во всех директивах кроме "СТЕРЕТЬ" дать пустую посылку, указанное действие будет произведено над текущим файлом. Это имя заносится при входе в Редактор, загрузке и запоминании файла.

В Редакторе предусмотрена специальная обработка ошибок чтения с диска: если какой-то участок текста не считывается, на его место в буфере записывается несколько голубых вопросительных знаков, после чего считывание продолжается.

Прежде, чем читать дальше, хорошо освойтесь с описанным выше порядком редактирования, проверьте работу всех клавиш редактора и директив меню, попробуйте набрать и отредактировать несколько программ, запомнить их на диск и считать оттуда.

5.3. Порядок запуска программ

Рассмотрим теперь общее правило оформления программы, предназначенной для записи на диск и последующего исполнения (возможно неоднократного):

I. Набрать директиву описания

ПРОЦ имя;

где имя – это название вашей программы (оно должно быть именем по правилам языка).

2. После входа в Редактор набрать текст программы (первой строкой должен быть тот же текст"ПРОЦ имя;").

3. В конце программы набрать

КНЦ;

4. Выйти из Редактора по директиве "КОНЕЦ ОПИСАНИЯ", на запрос системы

имя : ЗАПОМИНАТЬ (Д/Н)?

ответить "Д". При этом текст вашей программы запишется на диск под указанным именем (оно совпадает с тем именем, которое вы указали после слова "ПРОЦ").

Чтобы выполнить запомненную на диске программу, надо набрать следующие предписания:

ВВОД ИЗ ДЗУ: имя; имя ( );

Попробуйте исполнить таким образом одну из запомненных вами ранее программ.

Если при исполнении программы выдано сообщение об ошибке, или оказалось, что она работает неверно, можно вернуться в режим редактирования и исправить программу,так, как требуется. Это делается так же, как и первоначальное описание:

ПРОЦ имя;

и т.д.

Примечание. Если после выхода из Редактора вы не перевызывали систему, и хотите сразу выполнить только что набранную или исправленную программу, то для её вызова достаточно просто набрать

имя ( );

(конструкция "ВВОД ИЗ ДЗУ" используется для загрузки программы с диска).

5.4. Просмотр каталога диска

Посмотреть, какие программы запомнены на рабочем диске, всегда можно с помощью предписания

КАТАЛОГ ФАЙЛОВ;

или просто

КАТАЛОГ;

При этом выдается следующая информация:

заголовок библиотеки (который вы задавали при начальной разметке диска),

список всех файлов на диске с указанием их длины и типа,

число свободной памяти на диске.

Тип файла определяет способ доступа к нему и средства обработки Описываемая версия системы имеет дело только с текстовыми файлами (они обозначаются в каталоге буквой "Т"), хотя в других языках и системах, имеющихся на "АГАТЕ", используются файлы и других типов.

Длина файла и свободная память измеряются в блоках. Можно считать, что один блок - это 256 символов. Всего на рабочем диске в распоряжении пользователя находится 528 блоков..

Примечание. На рабочем диске, инициализированном штатной ДОС, доступны только 496 блоков.

На самом деле, в каталоге выдается лишь длина запомненной в файле информации. Кроме нее один блок занимают системные сведения о файле.

Каталог выдается порциями. За раз выдается столько строк, сколько помещается на экран, после чего выдача приостанавливается, чтобы дать возможность просмотреть выданную информацию. Для продолжения выдачи надо нажать любую клавишу. Клавиша останова F1 прерывает выдачу каталога.

РАЗДЕЛ 6. ПРОЦЕДУРЫ И ФУНКЦИИ

6.I. Основные черты процедурного блока

С процедурами и функциями, как объектами РАПИРЫ, вы уже познакомились в 3.II. В этом разделе мы рассмотрим аппарат процедур более подробно.

Чтобы не загромождать описания излишними уточнениями, будем везде говорить о процедурах, а при необходимости оговаривать особенности функций.

Основное назначение процедуры – выделение произвольной последовательности предписаний в отдельный блок с указанием имени, по которому этот блок доступен. Как видно из диаграммы 34, в простейшем случае достаточно оформить эти предписания так:

ПРОЦ имя;

тело процедуры – список предписаний через ";"

КНЦ;

Частный случай процедуры был описан в 5.3; там мы в виде процедуры оформляли целую программу, чтобы записать её на диск, а затем вызывать произвольное число раз.

Вызов процедуры в программе означает последовательное исполнение предписаний, образующих ее тело. Фактически, это позволяет заменить несколько предписаний в программе их общим именем и операцией вызова (круглые скобки после имени). Очевидно, что в виде отдельных процедур удобно оформлять повторяющиеся части программы.

Функции, кроме того, позволяют после выполнения некоторых действий выдать результат – произвольный объект РАПИРЫ. Как видно из диаграммы 35, это делается с помощью конструкции

РЕЗ : выражение;

(обратите внимание, что в заголовке функции стоит слово "ФУНК", а не "ПРОЦ"!)

По своему синтаксическому значению вызов функции является операцией, а не предписанием, как вызов процедуры, хотя обозначается так же. Это значит, что вызов функции может стоять везде, где допустимо выражение (мы уже приводили такие примеры в 4.7). Результатом операции вызова является значение выражения, записанного после "РЕЗ:" в описании данной функции.

Синтаксис вызова описывается диаграммой 7з. Как видно, и операция вызова функции, и предписание вызова процедуры выполняются над объектом. Этот объект может быть результатом других операций. Например:

```
<A,B,C> [I] ( );
```

будет вызвана одна из процедур A,B или C, в зависимости от значения имени I.

```
Ф ( ) ( ) -> X;
```

сначала вызовется функция Ф, затем ее результат (он тоже должен быть функцией), и уже его результат присвоится имени X.

6.2. Порядок описания и редактирования процедур

Порядок описания и редактирования процедур уже был описан в 5.3. Отметим лишь некоторые особенности и дадим необходимые пояснения.

I. В тело процедуры могут входить любые предписания РАПИРЫ, кроме системных директив (которые работают только в диалоговом режиме). К таким директивам относится и директива начала описания процедуры "ПРОЦ имя". Из этого следует, что вложенные описания процедур в РАПИРЕ недопустимы.

Примечание. Это, однако, не означает, что из одной процедуры нельзя вызвать другую.

2. По директиве "ПРОЦ" происходит поиск процедуры на диске, загрузка ее текста в память и входв режим редактирования. Средства редактирования были описаны в 5.2. После окончания редактирования и выходе в основной диалоговый режим текст исправленной процедуры остается в ОЗУ. При повторном входе в ту же процедуру по директиве "ПРОЦ" поиск на диске уже не проводится.

3. Для описания функций служит директива "ФУНК", аналогичная "ПРОЦ". Для входа в режим редактирования можно использовать и ее.

4. В режиме редактирования редактируемый текст может состоять только из описаний процедур и функций (возможно, разделенных комментариями). Любая другая информация, а также одиночные предписания при выходе по директиве "КОНЕЦ ОПИСАНИЯ" будет рассматриваться как ошибка.

5. Директива "КОНЕЦ ОПИСАНИЯ" в Редакторе (т.н. "выход с трансляцией") означает конец описания процедурного блока. По ней проводится синтаксический анализ текста; если были замечены ошибки, выдается сообщение об ошибке, и происходит возврат в режим редактирования.

Если текст синтаксически правилен, происходит трансляция всех процедур в редактируемом тексте, т.е. создание объектов языка вида "процедура" ("функция"), соответствующих приведенным описаниям. Каждый объект присваивается имени, указанному в заголовке его описания. Например, при трансляции описания

```
ФУНК МНОГОЧЛЕН;
РЕЗ:Х ** 2 + 10 * Х-8;
КЦ;
```


значением имени МНОГОЧЛЕН станет функция, выдающая в качестве результата значение многочлена x + 10x - 8 в зависимости от x.

6. После завершения трансляции выдается запрос "имя : ЗАПОМИНАТЬ (Д/Н)?". Следует ответить, нужно ли запоминать на диск отредактированный текст. Если текст не редактировался или был запомнен еще в Редакторе, данный запрос не выдается.

7. Имя, под которым описана процедура, по окончании трансляции получает частичную защиту, чтобы предотвратить случайную порчу процедуры неосторожным присваиванием.

8. Если описание процедуры уже хранится в ДЗУ, то вызвать ее можно двумя способами:

сделать "фиктивное" редактирование, т.е. войти в режим редактирования процедуры по директиве "ПРОЦ" и, не внося изменений, выйти по директиве "КОНЕЦ ОПИСАНИЯ";

воспользоваться предписанием "ВВОД ИЗ ДЗУ" (диаграмма 25).

Второй способ удобен тем, что предписание ввода из ДЗУ можно использовать в другой процедуре, организуя тем самым автоматическую загрузку всех необходимых процедур. Если процедур в вашей программе немного, имеет смысл поместить их все в один файл.

6.3. Блочная структура РАПИРЫ. Локализация имен

Одним из важнейших понятий программирования является блок (область действия описаний имен). Во многих языках программирования принято разделение имен на глобальные (известные во всей программе) и локальные (известные только внутри блока, имеющие в нем самостоятельный смысл и маскирующие одноименные глобальные имена на время работы блока).

Структура блоков и механизм локализации имен в РАПИРЕ несколько отличны от традиционных.

Глобальным именем в РАПИРЕ называется имя, известное в основном диалоговом режиме. Оно выдается в каталоге имен по предписанию "КАТАЛОГ ВСЕХ ИМЕН", исполненному в этом режиме. Как уже отмечалось, глобальные имена в описании не нуждаются.

Единственным видом блока в РАПИРЕ является процедура (функция). Все остальные составные предписания (ветвления, циклы) не являются блоками, и внутри них не могут быть локализованы имена (в некоторых языках это возможно).

Имена, описанные в процедуре, являются локальными в ней в общепринятом смысле. При вызове процедуры они имеют пустые значения, при выходе из нее выработанные значения уничтожаются.

Вместе с локальными в процедуре могут использоваться и глобальные имена, а также локальные имена вызвавшей процедуры (если такая есть). Описывать их дополнительно не надо. Если одно из этих имен совпадает по написанию с локальным в вызванной процедуре именем, то его значение в момент вызова становится недоступным на все время работы этой процедуры. При выходе из процедуры доступ к этому значению восстанавливается. Таким образом, имена в РАПИРЕ локализуются по динамической цепочке вызовов, а не по вложенности описания.

Значения глобальных имен в процедуре могут изменяться.

Локальные имена процедуры описываются при помощи конструкции

ИМЕНА: список имен через запятую ;

в заголовке процедуры (диаграммы I4,34,35).

Рассмотрим пример. Опишите процедуру:

```
ПРОЦ СУММА;
    ИМЕНА: СУМ, СЧ;
    0-> СУМ;
```



```
ДЛЯ СЧ ОТ НАЧ ДО КОН
    СУМ+СЧ ** 2-> СУМ;
ВСЕ;
ВЫВОД "СУММА КВАДРАТОВ ЧИСЕЛ ОТ ",НАЧ," ДО ",КОН,"
РАВНА ",СУМ;
КНЦ;
```

Локальными в ней являются имена СЧ и СУМ, глобальными – НАЧ и КОН (хотя они могут быть локальными в некоторой другой процедуре, вызывающей данную).

Наберите теперь следующие предписания:

```
I-> НАЧ; 5-> КОН; 28-> СУМ; СУММА ();
```

На экран будет выдано сообщение

```
СУММА КВАДРАТОВ ЧИСЕЛ ОТ I ДО 5 РАВНА 55
```

Если теперь набрать

```
ВЫВОД СЧ, СУМ;
```

то на экран будет выдано

```
"ПУСТО"  28
```

(объясните, почему). Как видим, локальное имя СУМ маскирует на время работы процедуры такое же глобальное имя.

Заметим в заключение, что ни одно из локальных имен не может совпадать по написанию с именем процедуры.

6.4. Параметры процедур и функций. Способы передачи параметро Рекурсия

В примере из предыдущего параграфа описана процедура СУММА, результат работы которой зависит от значений глобальных имен НАЧ и КОН. Если бы перед вызовом процедуры мы присвоили им другие значения, то получили бы совсем другой результат (попробуйте).

Обратите внимание, что для того, чтобы выполнить процедуру СУММА, мы должны определить значения имен НАЧ и КОН. Существует способ сделать это короче, без предварительного присваивания. Для этого процедуру СУММА следовало описать с параметрами:

ПРОЦ СУММА (НАЧ,КОН);

и т.д.

Тогда для вызова достаточно было бы только указать исходные значения:

СУММА (1,5); СУММА (-2,4);

Такая запись позволяет не заглядывать во внутреннюю структуру процедуры, т.е. полностью абстрагироваться от описания, рассматривать процедуру как "черный ящик", выполняющий некоторые действия в зависимости от входных параметров. Это особенно важно при написании программ, которыми будут пользоваться другие люди.

Итак, параметром процедуры называется ее локальная переменная, предназначенная для организации обмена данными между процедурой и вызвавшей ее программой. Параметры процедуры описываются отдельно от локальных имен: в скобках после имени процедуры перед ";" (диаграммы 34,35). Они также называются формальными параметрами.

При вызове процедуры в круглых скобках следует через запятую указать конкретные имена или значения (фактические параметры), передаваемые в процедуру или принимающие данные из нее.

Использование параметра в процедуре определяется способом его передачи. В РАПИРЕ предусмотрены три способа передачи параметров.

1. Входные параметры служат для передачи объектов в процедуру. При вызове формальному параметру процедуры присваивается такое же (но не то же) значение, какое имеет соответствующий ему фактический параметр.

Далее он рассматривается как самостоятельное имя, ничем не связанное с фактическим параметром. В частности, если входным параметром передано имя, то его значение копируется при входе в процедуру, после чего изменение этого имени в процедуре не влияет на ее исполнение.

2. Выходные параметры служат для передачи объектов из процедуры. При входе в нее они получают пустые значения, а при выходе выработанные значения присваиваются соответствующим фактическим параметрам. Поэтому в скобках вызова на месте выходных параметров должны стоять уточненные имена, способные принять выдаваемое процедурой значение. Внутри процедуры эти имена имеют абсолютную защиту, и изменять их значения нельзя.

3. Возвратные параметры совмещают в себе свойства входных и выходных: при входе их значения присваиваются формальным параметрам, а при выходе - обратно. Это значит, что в списке параметров при вызове вместо них также должны стоять уточненные имена. Возвратные параметры ничем не отличаются от глобальных имен, за тем исключением, что внутри процедуры присваивания этим именам недопустимы.

Способ передачи параметра описывается следующим образом:

перед входным параметром можно указать стрелку " => ";

после выходного параметра ставится стрелка " => ";

перед возвратным параметром ставится стрелка " <=> ".

Если имя указано без стрелки, оно считается входным параметром.

Пример:

```
ПРОЦ ЗАМЕНА ( <=> ТАБЛИЦА, ПОЛЕ, НОМЕР=> );
    1-> НОМЕР;
    ПОКА ТАБЛИЦА [НОМЕР] /=ПОЛЕ
        НОМЕР+1-> НОМЕР;
```

```
        ЕСЛИ НОМЕР > # ТАБЛИЦА ТО
              ТАБЛИЦА+ <ПОЛЕ> -> ТАБЛИЦА
        ВСЕ;
    ВСЕ;
КНЦ;
```

Нетрудно видеть, что эта программа ищет элемент ПОЛЕ в кортеже ТАБЛИЦА и, если его нет, добавляет в конец. В любом случае значением имени НОМЕР будет номер этого элемента в кортеже.

Здесь параметр ПОЛЕ является входным: для поиска и добавления в процедуре нам нужен сам объект. Параметр НОМЕР - выходной: в программе мы должны предусмотреть имя, которое примет выданное процедурой значение. Параметр ТАБЛИЦА - возвратный: обрабатываемый кортеж активно используется в программе как до, так и после вызова процедуры.

Может возникнуть закономерный вопрос, зачем нужны возвратные параметры, если процедура обрабатывает их как глобальные имена. Существенное отличие кроется в переименовании фактического параметра в формальный. Например, описанной выше процедурой можно обрабатывать как таблицу планет Солнечной системы, так и таблицу Менделеева. Для этого достаточно лишь подставить разные имена:

```
ЗАМЕНА (СИСТЕМА,<# ПЛАНЕТА:"ЗЕМЛЯ",РАССТОЯНИЕ:1.5Е8#> ,ИНД1);
ЗАМЕНА (ХИМИЯ,<#ЭЛЕМЕНТ:"УГЛЕВОД",ВЕС:12.011#> ,ИНД2);
```

Как уже указывалось, одна процедура в РАПИРЕ может вызывать другую. Особый интерес представляет случай, когда процедура вызывает сама себя. Этот прием называется рекурсией.

Рекурсия может быть прямой, если процедура вызывает сама себя непосредственно, и косвенной, когда процедура А вызывает процедуру В, а та, в свою очередь, снова вызывает А.

В обоих случаях нужно очень внимательно следить, чтобы последовательность таких вызовов имела условие окончания. В противном случае возможна ситуация, аналогичная зацикливанию (правда, в этом случае произойдет исчерпание системного стека с выдачей сообщения об ошибке "ПЕРЕПОЛНЕН СТЕК").

Рекурсия представляет собой мощное средство программирования, с ее помощью легко программируется вычисление по рекуррентным формулам. Приведем пример использования ее для вычисления факториала:

```
ФУНК ФАКТ (N);
    ИМЕНА:R ;
    ЕСЛИ N =1 ТО
       1-> R ;
    ИНАЧЕ
      N * ФАКТ (N-1)-> R
    ВСЕ;
    РЕЗ:R ;
КНЦ;
```

6.5. Порядок выполнения процедуры

Теперь, когда вам известны все особенности описания процедур и функций, опишем порядок их выполнения.

1. Отыскивается описание процедуры, являющееся значением выражения, записанного слева от скобок вызова.

2. Вычисляются значения всех фактических параметров (в порядке записи).

3. Происходит передача параметров по описанным в 6.4 правилам.

4. Устанавливается полная защита на все входные параметры. Соответствующие выходным и возвратным параметрам уточненные имена получают абсолютную защиту. Абсолютная защита устанавливается и на имя вызванной процедуры.

5. Исполняются все предписания, образующие тело процедуры (в порядке записи).

6. При вызове функции вычисляется значение результата – выражения, записанного после "РЕЗ:".

7. Уточненным именам, соответствующим выходным или возвратным параметрам присваиваются значения, выработанные в процедуре. При этом ведется контроль границ вырезки и защиты. Значения локальных имен и входных параметров уничтожаются.

8. В случае функции значение результата передается в качестве значения выражения (операции вызова).

6.6. Основные выводы

Из всего сказанного в этом разделе следует, что аппарат процедур играет в РАПИРЕ чрезвычайно важную роль. Помимо своих традиционных функций – единицы структурирования программы, локализации имен и действия описаний – процедура в РАПИРЕ является единственным средством хранения программы в ДЗУ.

Таким образом, в языке нет особого понятия программы. Ею является любая последовательность предписаний, исполненная в основном диалоговом режиме. Понятие задания, принятое в пакетных языках, в РАПИРЕ успешно заменяется теми же процедурами, оформленными при необходимости в виде пакета.

Пакет – это совокупность процедур и функций, среди которых выделена головная процедура, выполняющая загрузку остальных процедур пакета и некоторые начальные действия.

Благодаря наличию в языке процедурных и функциональных значений значительно расширены возможности процедурного аппарата языка. Использование этих значений в выражениях позволяет программно управлять вызовом тех или иных подпрограмм (в частности, на РАПИРЕ элементарно реализуются переключатели – выбор одной из процедур по индексу).

Возлагая на процедурный аппарат все нагрузки, связанные с подготовкой, хранением и запуском программ, никак нельзя не предусмотреть средства отладки этих программ. Это требование приобретает особое значение для учебного и диалогового языка программирования.

Отладочным возможностям РАПИРЫ посвящен следующий раздел.

РАЗДЕЛ 7. ПРИНЦИПЫ ОТЛАДКИ ПРОГРАММ В СИСТЕМЕ "ШКОЛЬНИЦА". ОТЛАДОЧНЫЕ СРЕДСТВА ЯЗЫКА РАПИРА

7.1. РАПИРА, как система программирования. Программная среда.

Для работы с вычислительной машиной еще недостаточно знания алгоритмического языка и умения составлять на нем программы. Требуется умение работать с той или иной системой программирования, которой оснащена данная ЭВМ. Системы программирования позволяют вводить, хранить, редактировать, исполнять, отлеживать программы и т.д.; как правило, они имеют свой собственный язык управления – на уровне команд или директив.

На персональных ЭВМ наблюдается тенденция к слиянию языка и системы программирования, когда все управление ЭВМ ведется средствами того же языка, на котором составляются программы. Работа в таких системах более удобна и эффективна, потому что для запуска задачи программисту не требуется знания языков обслуживающих систем.

Кроме того, язык программирования обладает несравненно более мощными средствами управления, чем язык команд. Это мы еще проиллюстрируем позднее.

Такое слияние позволяет говорить о единой программной среде, как совокупности состояния системы и возможностей изменения этого состояния средствами языка. Программная среда в каждый момент времени характеризует состояние и возможности всей ЭВМ.

В системе "ШКОЛЬНИЦА" таким языком является РАПИРА.

Пользователь в системе является активным компонентом программной среды. Он имеет возможность формировать и изменять ее в соответствии с особенностями решаемых задач в режиме непосредственного диалога с ЭВМ, не прибегая к специальному оформлению своих действий в виде отдельной программы и вызову специальных систем.

Помимо чисто языковых средств пользователю предоставлены мощные средства управления программами, право вмешиваться в ход их исполнения, корректировать его при необходимости. Все это в совокупности образует набор отладочных средств РАПИРЫ, которые жизненно необходимы в учебно-производственном языке на этапе формирования навыков и стиля программирования.

7.2. Управление программами. Системные директивы. Режимы диалога

Как вы уже знаете, в системе предусмотрены две формы исполнения программ.

Первая – это ввод предписаний непосредственно в режиме диалога. Каждое предписание исполняется, как только оно поступает в машину.

Пользователь в любой момент, получив приглашение к вводу, имеет право выдать значения имен, произвольных выражений, просмотреть каталог имен и файлов на диске, т.е. выполнить любые предписания языка.

Вторая – это вызов программы, запомненной в ДЗУ как процедура или функция. Из того, что вам известно о системе до сих пор, следует, что управление вернется к пользователю лишь после того, как эта программа выполнится. Для организации отладки этой программы традиционными средствами придется вставлять в нее выдачу отладочной информации (как это делается – другой вопрос).

Однако, в систем предусмотрен мощный аппарат, позволяющий получить исчерпывающую информацию о программе в любой момент ее исполнения. Это режим приостанова (паузы), устанавливаемый по предписанию СТОП при исполнении процедуры или функции.

Предписание

СТОП;

записанное в программе, вызывает ее приостанов и выход в диалоговый режим. При этом состояние программной среды сохраняется, и пользователь получает возможность взглянуть на состояние объектов и имен в момент приостанова. В частности, можно узнать значения локальных имен и параметров процедуры, которые недоступны вне нее.

Кроме того, можно вручную присвоить некоторым именам другие значения, в том числе – локальным именам и параметрам. Помните только, что некоторые имена в момент приостанова могут иметь защиту и что изменение значений имен должно обеспечивать возможность продолжения программы.

Переход в режим приостанова сопровождается появлением в информационной строке экрана слова "ПАУЗА" и выдачей сообщения о том, в какой процедуре и в какой ее строке произошел приостанов, например:

ОСТАНОВ В СТРОКЕ 15 "РАКЕТА"

В этом режиме доступны те же предписания, что и в основном, но есть и три специфичные директивы.

По директиве

ПУСК;

происходит выход из режима приостанова и запуск приостановленной программы, начиная с того места, где произошел приостанов. Дальнейшее выполнение программы ведется уже с учетом возможных изменений, внесенных пользователем в режиме приостанова.

Директива

ШАГ;

срабатывает также, как и "ПУСК", но выполняется только одна строчка программы. После этого снова происходит приостанов по общим правилам. Эта директива позволяет организовать "пошаговое" исполнение программы, удобное, например, при поиске ошибки (согласитесь, что вставлять "СТОП;" после каждого предписания процедуры утомительно).

По директиве

ВЫХОД;

происходит завершение остановленной программы (т.е. последовательный выход из всех вложенных по вызову процедур без продолжения работы) и выход в основной диалоговый режим.

Таким образом, режим приостанова можно назвать "стоп-кадром" программы, позволяющим временно прервать программу, чтобы посмотреть и проанализировать результаты выполненных действий, а затем продолжить ее или отказаться от дальнейшего выполнения.

Преимущества режима приостанова, как средства отладки, заключаются в том, что он позволяет не загромождать экран или бумагу зачастую излишней отладочной информацией, а выбрать лишь ту, которая нужна в данный момент.

При этом сохраняется та же свобода действий, что и в основном режиме.

При ошибке в процедуре происходит переход в режим останова по ошибке. Он сопровождается появлением в информационной строке экрана слова "ОШИБКА" и сообщения о характере ошибки с указанием процедуры и номера строки, где она произошла, например:

НЕДОПУСТИМЫЕ ОПЕРАНДЫ "+" В СТРОКЕ 32 "СЧЕТЧИК"

Этот режим отличается от режима приостанова только тем, что продолжение выполнения программы невозможно, в связи с чем недопустимо использование директив "ПУСК" и "ШАГ". Режим останова по ошибке предусмотрен для того, чтобы дать возможность пользователю ознакомиться с состоянием программной среды в момент ошибки.

В заключение отметим еще ряд особенностей режимов "ПАУЗА" и "ОШИБКА".

1. Директива описания процедуры "ПРОЦ" ("ФУНК"), выполненная из этих режимов, перед входом в режим редактирования осуществляет действия, предусмотренные директивой "ВЫХОД".

2. Нажатие клавиши останова "ВI" во время выполнения программы также вызывает приостанов программы и переход в режим "ПАУЗА".

3. Еще раз обратим внимание, что "ПУСК", "ШАГ" и "ВЫХОД" являются директивами режимов "ПАУЗА" и "ОШИБКА", т.е. не могут встречаться в теле процедуры и быть исполнены в основном режиме.

4. Из режимов "ПАУЗА" и "ОШИБКА" допустим вызов другой программы (например, программы обработки ошибки), которая тоже может быть приостановлена. Не следует, однако злоупотреблять этой возможностью, потому что

а) вернуться в режим приостанова первой программы можно только

после нормального завершения второй; в частности, при ошибке во второй программе, первую продолжить нельзя;

б) по директиве "ВЫХОД" происходит сквозной выход из всех остановленных программ в основной диалоговый режим; директива "СТОП;", выполненная из основного режима - например, внутри составного предписания - вызывает лишь прекращение его выполнения без перехода в режим приостанова;

в) возможны ошибки при попытке присваивания во второй программе тем глобальным именам, которые в момент приостанова первой программы имели защиту.

Наконец, очень трудно не испортить вызовом второй программы рабочее состояние программной среды для первой.

Поэтому перед перевызовом программы следует выполнять выход в основной режим.

7.3. Языковые средства отладки. Прокрутка и след. Предписание "ВКЛ/ВЫКЛ"

Для организации отладки программ в РАПИРЕ предусмотрены и традиционные средства. Это прежде всего прокрутка имен и трассировка вызовов процедур (функций) и выходов из них (след). Они, как и некоторые другие механизмы РАПИРЫ, представлены в виде некоторых режимов работы системы, которые можно включить и выключить. Это делается с помощью предписания "ВКЛ/ВЫКЛ" (диаграмма 26).

Прокрутка имени означает выдачу сообщения о каждом присваивании тому или иному имени, например:

```
ИМЯ = I25
КОРТЕЖ [...] = .ПУСТО.
ТЕКСТ = "БЕГЕМОТ"
```


Слева от знака равенства указывается имя, справа - новое значение. "..." означает, что происходит изменение только некоторой части его значения, например, при присваивании выборке или вырезке.

Можно включить (выключить) прокрутку одного, нескольких или сразу всех имен. Как сделать это, показывают следующие примеры:

```
ВКЛ ПРОКРУТКУ : ОДНО;
ВКЛ ПРОКРУТКУ ИМЕН: ОДНО, ДВА, ТРИ;
ВЫКЛ ПРОКРУТКУ ИМЕН;
```

Слово "ИМЕН" везде можно опускать.

Имена, прокрутка которых включена, в каталоге имен выдаются со звездочкой, например:

```
* ИМЯ.....ЦЕЛ
```

Включение прокрутки сразу всех имен в каталоге не отражается.

Следом процедуры или функции называется сообщение о входе в нее или выходе из нее. При входе сообщается имя вызванной процедуры и информация о параметрах:

для входных параметров выдается только передаваемое в процедуру значение;

для выходных выдается только имя, которое примет значение из процедуры;

для возвратных выдается и то, и другое.

При выходе из процедуры также выдается сообщение, в случае выходе из функции выдается возвращаемый результат. Примеры:

```
СТРЕЛА(I32, ВЫХОДНОЙ_ПАРАМЕТР, ВОЗВРАТНЫЙ_ПАРАМЕТР = I,2)
ЦЕЛЬ (I2)
ЦЕЛЬ = "РЕЗУЛЬТАТ"
КОНЕЦ РАБОТЫ "СТРЕЛА"
```

Если вызываемая процедура является не значением имени, а результатом некоторого выражения, то вместо ее имени выдается .ВЫР.

След всех вызываемых процедур включается и выключается, соответственно, следующими предписаниями:

ВКЛ СЛЕД;

ВЫКЛ СЛЕД;

Более подробную информацию о ходе выполнения процедуры можно получить включением следа строк:

ВКЛ СЛЕД СТРОК;

ВЫКЛ СЛЕД СТРОК;

Если этот режим включен, то при переходе к выполнению очередной строки процедуры выдается ее номер, например:

#12 #13 #25 #26 #13 #25 #26

Отметим сразу, что в некоторых случаях допустимо отклонение на одну строку, например, если предписание занимает несколько строк или после него нет точки с запятой.

Предписание ВЫКЛ СЛЕД выключает и след строк, если он был включен.

Прокрутка и след позволяют проследить за изменением значений имен в программе, за порядком вызова процедур и их взаимодействием с программой.

Как уже неоднократно говорилось, хорошим отладочным средством является защита имен по записи, позволяющая защитить те или иные имена от случайных и нежелательных присваиваний.

Приостановить выполнение программы без перехода в режим приостанова можно нажатием пробела. Это бывает удобно, когда, например, программа выдает на экран непрерывный поток информации, и надо просмотреть ее по частям, временно останавливая работу программы.

Чтобы продолжить исполнение программы, достаточно нажать любую клавишу.

Еще один механизм РАПИРЫ, удобный не только для отладки, но и для программирования на РАПИРЕ в целом, рассмотрен в разделе 9.

7.4. Обработка ошибок

Программируемая обработка ошибок, предусмотренная каноническим описанием в настоящей версии не реализована. Вручную исполнить те или иные действия можно непосредственно из режима останова по ошибке (например, вызвать соответствующую процедуру обработки).

Для сбора статистики по допущенным ошибкам вызовите процедуру СТАТ, которая хранится на диске Ш1. Для ее вызова достаточно исполнить предписания:

ВВОД ИЗ ДЗУ:СТАТ;

СТАТ ();

после чего на экран телемонитора будет выдана информация о том, какие ошибки и сколько раз были допущены с момента последнего вызова (перевызова) системы.

7.5. Краткие выводы

Перечислим еще раз все отладочные средства РАПИРЫ:

1. Режим приостанова процедур и пошаговое исполнение программ (7.2).
2. Защита имен по записи (3.5, 4.7).
3. След вызовов процедур и функций (7.3).
4. Прокрутка имен (7.3).
5. Организация контрольных точек (предписание "КОНТРОЛЬ",4.5).
6. Отладочный вывод пользователя.

РАЗДЕЛ 8. ФАЙЛЫ И СМЕН ПРИНЦИПЫ РАБОТЫ С ВНЕШНЕЙ ПАМЯТЬЮ

8.1. Организация внешней памят . Файлы

С внешней памятью вам уже не раз приходилось иметь дело при записи и чтении программ (см. 5 раздел). Ознакомимся теперь более подробно со структурой и другими способами работы с ней.

Долговременным запоминающим устройством (ДЗУ) в описываемой версии системы служит дисковод (один или два).

Внешняя память ЭВМ располагается на гибких магнитных дисках. Она предназначена для долговременного хранения произвольной информации, в частности, программ. Эта информация не уничтожается при выключении машины или выходе из системы, в отличие от той, которая находится в ОЗУ.

Именная самостоятельный смысл информация, хранящаяся на диске, называется внешним файлом и имеет имя, позволяющее отличить один файл от другого.

Каждый рабочий диск представляет собой библиотеку файлов. Часть места на диске отведена под каталог, содержащий краткие сведения о каждом файле: имя, возможность доступа к нему и возможности обработки. Каталог используется системой для поиска нужного файла на диске. Часть памяти занимают системные программы; все остальное место отведено под файлы.

Вся память на диске разделена на блоки. Блок является единицей выделения и освобождения внешней памяти.

Способ обработки файла определяется его типом. В описываемой версии системы обрабатываются только текстовые файлы.

Файл может быть заперт (закрыт на запись). Информация, хранящаяся в таком файле, может быть только прочитана, но не может быть изменена или уничтожена.

Работу с внешней памятью осуществляет встроенная файловая система. Помимо обращения к ДЗУ из Редактора (см. 5.2) она обеспечивает в языке работу с файлами произвольного доступа, значительно расширяющей его возможности.

8.2. Текстовые файлы произвольного доступа

С точки зрения пользователя файл - это составной объект языка, который находится в ДЗУ , т.е. на диске, и сохраняется при выключении машины. Этим определяются отличия файла от других объектов языка и особые способы его обработки. Напомним еще раз, что здесь и ниже речь идет только о текстовых файлах.

Файл представляет собой упорядоченную последовательность литер, оканчивающуюся специальным символом "конец файла". Эта последовательность может расширяться.

Файлы в РАПИРЕ обрабатываются по принципу прямого доступа. Это означает, что во время обработки у файла существует указатель текущей позиции, указывающий на литеру файла, которая может быть прочитана или заменена в данный момент. Определение позиции и смещение указателя позволяют добраться к любой литере файла.

Две основные формы доступа к файлу - это ввод и вывод. Ввод означает считывание литеры из текущей позиции файла в ОЗУ. Вывод - запись литеры из ОЗУ в текущую позицию. Они возможны в любой позиции файла, причем после обработки текущей литеры указатель смещается к следующей позиции. Если в позиции указателя находится символ "конец файла", то попытка ввода считается ошибкой, а при выводе очередная литера помещается в конец файла, расширяя его.

Символ конца при этом сдвигается в следующую позицию.

Можно заметить, что обработка файла напоминает работу с обычным магнитофоном. Файл здесь играет роль магнитной пленки, на которой записана некоторая информация, указатель – роль головки звукоснимателя, перемещение указателя соответствует обычной перемотке ленты вперед–назад, а ввод–вывод аналогичен записи и воспроизведению.

Возможностями ввода, вывода и перемещения указателя исчерпывается набор средств обработки файлов. Отметим еще некоторые особенности:

I. Файл может только расширяться, причем лишь в одну сторону.

2. Указатель можно перемещать только внутри файла, т.е. номер его позиции должен быть числом от I до N, где N – позиция символа конца файла.

По своей структуре файл очень похож на текст, но в силу описанного выше способа обработки мы получаем возможность работать с очень большими текстовыми данными в ДЗУ (в несколько раз превосходящими размеры ОЗУ), не считывая их целиком в память – ведь зачастую это и не нужно.

Последовательный ввод–вывод позволяет хранить в файле объекты РАПИРЫ, имеющие текстовое представление (числа, тексты, множества, кортежи), и при необходимости вводить их с диска в ОЗУ.

Прочие полезные и удобные возможности файлов будут рассмотрены далее.

8.3. Файлы, как объекты РАПИРЫ. Открытие и закрытие файла

Как уже отмечалось, файлы являются также объектами языка РАПИРЫ.

Рассмотрим в этом параграфе файлы с точки зрения программной среды.

Файл, как носитель информации, существует только на диске и в отличие от других объектов РАПИРЫ является вполне реальным физическим образованием. Для работы с ним в программе требуется предварительно организовать взаимодействие между ЭВМ и диском, иначе говоря, наладить канал связи между ОЗУ и ДЗУ, по которому будет вестись обмен данными. Это процесс называется открытием доступа к файлу или просто открытием файла.

Предписание открытия файла имеет вид:

ОТКРЫТЬ текстовое выражение КАК имя;

Выполнение его заключается в следующем.

I. Проверяется, есть ли свободный канал связи с ПЗУ. Если нет, выдается сообщение об ошибке "СЛИШКОМ МНОГО ОТКРЫТЫХ ФАЙЛОВ". Всего допустимо одновременно обрабатывать 7 файлов.

2. Определяется имя внешнего файла, доступ к которому должен быть открыт. Если текстовое выражение указано, именем файла считается текст, определяемый этим выражением. В противном случае именем файла считается имя.

3. Проверяется защита, которую имеет имя. Если она полная или абсолютная, выдается сообщение об ошибке, и файл не открывается.

4. Происходит поиск указанного файла в текущей библиотеке. Если он не обнаружен, заводится пустой текстовый файл.

5. Указатель помещается в первую позицию файла.

6. Значением имени имя становится объект файловое значение. Оно определяет канал связи с открываемым файлом. В дальнейшем употребление этого значения в операциях и предписаниях файловой системы означает обращение к соответствующему каналу связи, а следовательно – к файлу в ДЗУ.

7. На имя устанавливается частичная защита.

Описанная конструкция позволяет открыть файл под своим именем или под любым другим. Например:

ОТКРЫТЬ АРХИВ КАК АРХИВ;

ОТКРЫТЬ АРХИВ;

ОТКРЫТЬ <"ФАЙЛ1", "ФАЙЛ2", "ФАЙЛ3">[X] КАК ФАЙЛ;

Первые два предписания эквиваленты. Третий пример показывает, как можно открыть один из нескольких файлов в зависимости от некоторых условий.

Внимание! Пока существует хотя бы один канал связи с диском, т.е. пока открыт хотя бы один файл, диск переставлять нельзя! Это будет рассматриваться как ошибка.

После выполнения предусмотренной обработки файла его необходимо закрыть, т.е. освободить канал связи с файлом. Если не сделать этого, возможна частичная потеря информации в файле или даже порча библиотеки.

Для закрытия файлов служит предписание

ЗАКРЫТЬ имя ;

Если указано имя, то закрывается доступ только к связанному с ним файлу; иначе закрываются все открытые в данный момент файлы.

При закрытии файла соответствующее ему файловое значение не уничтожается, но перестает выполнять свои функции доступа к файлу.

Обратите внимание, что доступ к файлу в программе определяется объектом "файловое значение", а не именем, под которым он открыт. Например, после выполнения предписаний:

ОТКРЫТЬ Ф; Ф-> Ф1; <1,2,Ф1> -> Ф2;

Имена Ф, Ф1 и третий элемент кортежа Ф2 будут определять доступ к одному и тому же файлу. Самому имени Ф можно даже присвоить другое значение.

При уничтожении последней ссылки на файловое значение оно уничтожается, а сам файл закрывается. Например, после выполнения предписаний

ОТКРЫТЬ Ф; Г-> Ф;

открытый файл будет сразу же закрыт. Чтобы предотвратить потерю доступа к файлу в результате случайного присваивания имени, на имя, под которым открывается файл, и устанавливается частичная защита.

Примечание. Для облегчения понимания программы рекомендуется указывать в предписании "ЗАКРЫТЬ" то имя, под которым файл был открыт. Именно поэтому в нем допускается только имя, а не произвольное файловое выражение.

Закрытие уже закрытого файла не считается ошибкой; никакие действия при этом не выполняются.

8.4. Языковые средства обработки файлов

1. Текущую позицию файла можно определить с помощью операции

@ файловое выражение

Значением файлового выражения здесь и далее должен быть некоторый открытый файл. Результатом выполнения операции является целое число от 1 до номера позиции символа конца файла. Эта операция является полноправной унарной операцией и может стоять в любом месте, где допустимо выражение.

2. Признак конца файла определяется с помощью стандартной функции

КФ (файловое выражение)

В качестве результата она выдает литеру "Д", если в текущей позиции файла находится символ конца, и литеру "Н" – в противном случае.

Цикл вида

ПОКА КФ (ФАЙЛ)="Н" :: ... ВСЕ;

является типичным примером последовательной обработки файлов.

3. Переместить указатель файла к другой позиции позволяет предписание

ПОЗИЦИЯ файловое выражение = выражение;

где значением выражения является целое число - номер новой позиции указателя - от I до N-I (N- позиция символа конца файла).

Если новая позиция файла задана неверно, выдается сообщение об ошибке, и указатель устанавливается в последнюю позицию на символ конца файла. Примечание: если номер позиции меньше I или заведомо больше размеров библиотеки, указатель не сдвигается.

4. Вывод в файл осуществляется средствами вывода языка РАПИРА (предписания "ВЫВОД", "КАТАЛОГ" и др., рассмотренные в 4 разделе). Для этого, как вы уже могли заметить при знакомстве с диаграммами этих предписаний, в поле "устройство вывода" следует писать

В ФАЙЛ имя файла

(слово "ФАЙЛ" можно опускать). Например:

ВЫВОД В ФАЙЛ Ф: ...

КАТАЛОГ ИМЕН В ФАЙЛ КТЛ;

Вывод в файл всей предусмотренной предписанием информации осуществляется по тем же правилам и в том же виде, что и на экран или бумагу. Порядок вывода в файл очередного символа был описан в 8.2. Вывод возможен и в середину, и в конец файла. В последнем случае файл расширяется.

Перевод строки выводится в файл как литера "УПР-М".

5. Ввод из файла также осуществляется средствами ввода языка РАПИРА. Для этого в заголовке предписания "ВВОД" необходимо писать

ИЗ ФАЙЛА имя файла

(слово "ФАЙЛА" можно опускать).

Ввод из файла осуществляется так же , как с клавиатуры: возможен ввод в режиме приема текстов и в режиме приема данных, записанных в файле в текстовом виде. Тексты и данные должны быть записаны в файле по тем же правилам, по каким они вводятся с клавиатуры с учетом всех ограничений на длину лексем, мощность структур и т.п. (см. 4.9). Концом вводимой строки считается символ перевода строки "УПР-М".

Пример:

ВВОД ИЗ ФАЙЛА Ф1:A,B,C;

ВВОД ИЗ ФАЙЛА Ф2 ДАННЫХ:K [I] ,M;

Таким образом, вывод и ввод в случае файлов совместимы: любая выведенная информация может быть введена с соблюдением разбиения на строки.

Особенности ввода из файла:

попытка чтения символа конца файла считается ошибкой (ситуация исчерпания данных в файле);

при ошибке ввода данных в режиме "ВВОД ДАННЫХ" перезапрос не происходит, и ввод прерывается с выдачей сообщения об ошибке;

записанные в файле управляющие символы "-> ", "<-" и др. при вводе из него редактирующих действий не выполняют.

6. Для удобства полиферной обработки файлов предусмотрена функция чтения нескольких литер из файла:

СИМФ (файловое выражение, выражение)

Значение выражения должно быть целым числом от 0 до 255 – оно показывает, сколько литер надо считать из файла.

Очередные литеры из указанного файла оформляются в виде текста и выдаются результатом функции. Указатель при этом сдвигается к первой непрочитанной литере. Как и при вводе, ведется контроль конца файла.

Приведем теперь полезные приемы использования файлов в программе:

а) Распечатка содержимого файла:

```
ОТКРЫТЬ ИСТОЧНИК;
ПОКА КФ (ИСТОЧНИК)="Н" ::
    ВЫВОД НА БУМАГУ БПС: ЧТФ(ИСТОЧНИК,1);
ВСЕ;
ЗАКРЫТЬ ИСТОЧНИК;
```

б) Относительное смещение на X символов по файлу:

```
ПОЗИЦИЯ Ф=@Ф+X;
```

в) Программная обработка каталога файлов (получение информации о первом файле в каталоге)

```
ОТКРЫТЬ КТЛ;
КАТАЛОГ В КТЛ;              (* ЗАПИШЕМ КАТАЛОГ В ФАЙЛ *)
ПОЗИЦИЯ КТЛ=1;              (* ВЕРНЕМСЯ К НАЧАЛУ *)
ВВОД ИЗ КТЛ :П,П,С;         (* ПРОПУСТИМ ПУСТУЮ СТРОКУ,
ЗАГОЛОВОК *)
                            (* И ВОЗЬМЕМ ДАННЫЕ О ФАЙЛЕ № 1 *)
ЧТФ(КТЛ,1)-> ЗАЩИТА;        (* ОПРЕДЕЛЕНИЕ ПО КАТАЛОГУ *)
ЧТФ(КТЛ,1)-> ТИП;           (* ВСЕЙ ИНФОРМАЦИИ О ФАЙЛЕ *)
@КТЛ-> ПОЗ;
ПОЗИЦИЯ КТЛ=ПОЗ+5;          (* НАСТРОИМСЯ НА ИМЯ *);
```



```
ВВОД ИЗ КТЛ:ИМЯ_ФАЙЛА;
ПОЗИЦИЯ КТЛ=ПОЗ;                (* ВЕРНЕМСЯ К ДЛИНЕ *)
ВВОД ИЗ КТЛ ДАННЫХ:ДЛИНА;       (* ОСТАТОК ИГНОРИРУЕТСЯ *)
(* СЮДА МОЖНО ВСТАВИТЬ ОБРАБОТКУ ПОЛУЧЕННЫХ *)
(* ДАННЫХ ИМЯ_ФАЙЛА, ЗАЩИТА, ТИП И ДЛИНА *)
        ЗАКРЫТЬ КТЛ;
```

Примечание. Текст процедуры или функции хранится в ДЗУ в виде текстового файла. Поэтому в РАПИРЕ можно программно обрабатывать тексты процедур и рассматривать как тексты процедур произвольные файлы, сформированные программно. В частности, их можно редактировать с помощью Редактора программных текстов.

8.5. Работа с библиотеками в программе

Кроме обработки отдельных файлов, в РАПИРЕ предусмотрены минимальные возможности обработки библиотек.

Выдача каталога файлов в библиотеке, как известно, происходит по предписанию "КАТАЛОГ". Форма выдаваемой информации была описана в 5.4. Заметим, только, что защищенные файлы отмечаются звездочкой в первой позиции строки.

Защита файла по записи и снятие с него этой защиты выполняется по предписаниям

```
ЗАПЕРЕТЬ  текстовое выражение;
ОТПЕРЕТЬ  текстовое выражение;
```

соответственно.

Уничтожение файла в библиотеке выполняется по предписанию

```
СТЕРЕТЬ  текстовое выражение;
```

При этом вся занимаемая файлом память на диске освобождается для дальнейшего использования.

Значение текстового выражения в этих предписаниях определяет имя файла на диске. Рассмотрим некоторые общие правила записи имен внешних файлов (они относятся и к именам, стоящим в предписании "ОТКРЫТЬ"):

именем внешнего файла может быть произвольный текст длиной 30 литер,

можно указывать более короткие имена (вплоть до пустого текста); при этом они дополняются пробелами справа.

если длина имени больше 30, то все литеры начиная с 3I-й, игнорируются.

Эти правила записи имен файлов распространяются и на режим работы с ДЗУ в Редакторе.

В заключение подчеркнем, что все описанные предписания могут быть исполнены как в диалоговых режимах, так и в теле процедуры.

8.6. Переключение ДЗУ

В этом параграфе описываются дополнительные возможности системы, если в ее конфигурацию входят два дисковода, подключенные к одному разъему.

Одновременная обработка двух ДЗУ ведется с помощью стандартной процедуры

ДЗУ (выражение);

где значением выражения могут быть числа I или 2.

Данная процедура устанавливает номер т.н. текущего активного ДЗУ. Это значит, что по предписаниям "ОТКРЫТЬ", "ЗАПЕРЕТЬ", "ОТПЕРЕТЬ", "СТЕРЕТЬ", "КАТАЛОГ" и директивам "ПРОЦ" и "ФУНК" обращение произойдет к тому ДЗУ, которое в этот момент объявлено активным.

В каждый момент активным может быть только одно устройство.

Особо отметим, что другое ДЗУ вовсе не является недоступным. При открытии файла система запоминает, на каком устройстве он открыт. В дальнейшем весь обмен с этим файлом ведется, естественно, с того же ДЗУ, независимо от того, активно оно или нет.

В Редакторе переключение ДЗУ происходит по директиве "РАБОТАТЬ С ДЗУ *N* ".

РАЗДЕЛ 9. ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА В РАПИРЕ.

9.I. Потоки информации и их распределение по внешним устройствам.

Сейчас вы уже ознакомились со всеми формами ввода и вывода в РАПИРЕ (как в языке, так и в системе). По характеру и источникам возникновения всю информацию, участвующую в диалоге человека с ЭВМ можно разбить на следующие потоки.

Потоки вывода:

I. Системная информация выдается системой и служит для поддержания системного диалога с пользователем. Сюда относятся:

приглашение к вводу и отображение набираемого на клавиатуре текста,

диагностические сообщения об ошибках и переходе в режим приостанова,

системные запросы "ЗАПОМИНАТЬ (Д/Н)?" и "ПРИСВАИВАТЬ (Д/Н)?".

2. Программный вывод состоит из информации, выдаваемый пользователем программными средствами. К нему относятся:

тексты и данные, выводимые по предписанию "ВЫВОД",

информация, выдаваемая по предписанию "КАТАЛОГ".

3. Отладочный вывод включает в себя выдачу отладочной информации в соответствующих режимах. К ней относятся:

прокрутка имен,

след входов и выходов процедур и функций,

след строк в процедуре.

Все эти три потока в совокупности образуют полный протокол диалога пользователя с ЭВМ.

Потоки ввода:

I. Системный ввод предписаний и директив в диалоговых режимах.

2. Программный ввод, т.е. ввод, организуемый пользователем в программе с помощью предписания "ВВОД".

Примечание. Специальные средства символьного ввода, жестко связанные с определенным внешним устройством (функции ЧТФ – для файлов; КЛАВ, НАЖАТО – для клавиатуры, ЭКЦВ, ОКСИМ – для экрана телемонитора и т.п.), в потоки ввода не объединяются.

В потоки вывода не включаются сообщения, выдаваемые в информационной строке экрана и специально организуемый с помощью стандартных процедур графический вывод на экран.

Перечислим теперь все устройства, которые могут получать или выдавать информацию.

Устройства вывода:

I. Экран телемонитора.

2. Печатающее устройство.

3. Файлы.

Устройства ввода:

I. Клавиатура.

2. Файл.

Как вы уже знаете, в языке предусмотрена возможность сделать разовый вывод на любое из перечисленных устройств вывода или разовый ввод с любого устройства ввода.

Однако, существует возможность на некоторое время полностью связать один из потоков с одним из устройств, например, организовать вывод всей отладочной информации только в файл, чтобы не загромождать экран, сделать копию протокола диалога с ЭВМ на бумагу, переключить

ввод данных из программы с клавиатуры на файл, и т.п. Даже по этим примерам видно, какие мощные средства программирования и отладки предоставляет аппарат переключения потоков.

9.2. Средства переключения потоков

Все переключение потоков в РАПИРЕ осуществляется предписаниями "ВКЛ/ВЫКЛ".

Переключение имеет вид:

ВКЛ вид устройство;

ВЫКЛ вид устройство;

(см. также диаграмму 26).

В случае переключения вывода вид – это название вида выдаваемой информации:

"ВЫВОД" – программного вывода,

"ОТЛАДКУ" – отладочного вывода,

"ПРОТОКОЛ" – всех трех потоков вывода одновременно.

Устройство – это "НА ЭКРАН", " НА БУМАГУ" или "В ФАЙЛ имя". Смысл этих конструкций очевиден.

Включение потока вывода на одно из устройств означает, что вся относящаяся к этому потоку выходная информация будет выводиться в том числе и на указанное устройство (возможно, до этого она выдавалась на другие устройства), т.е. подключение потока к устройству.

Выключение означает, что если данный вид информации выводился на указанное устройство, то теперь этот вывод отключается.

Вспомним теперь основную форму вывода данных: если в заголовках предписаний "ВВОД" или "КАТАЛОГ" указано устройство, то вывод ведется только на него (разовый вывод). Если же устройство вывода

не указано, то выводная информация через поток программного вывода поступает на все устройства, к которым он подключен.

Польза такой возможности иллюстрируется следующим примером. Если вы хотите получить на бумаге результаты выполнения вашей программы, то вместо утомительной замены каждого слова "ВЫВОД" на "ВЫВОД НА БУМАГУ" достаточно перед вызовом программы набрать предписание

ВКЛ ВЫВОД НА БУМАГУ;

Переключение ввода в связи с небольшим количеством вариантов реализуется проще:

ВКЛ ВВОД ИЗ ФАЙЛА имя;

ВЫКЛ ВВОД ИЗ ФАЙЛА имя;

Первая конструкция означает переключение программного ввода на файл с указанным именем, вторая – переключение его снова на клавиатуру.

Переключать системный ввод нельзя!

Аналогично выводу, все предписания ввода с явно указанным устройством выполняют ввод с этого устройства, а все предписания, в которых оно не указано, – с того, к которому подключен поток ввода.

Опять-таки, используя эту возможность, вы можете записать в файл все данные, вводимые вашей программой, и перед ее вызовом настроить ввод на этот файл.

Примечание. При переключении ввода-вывода на файл, а также при разовом вводе-выводе, необходимо следить за тем, чтобы файл был открыт. В противном случае будет выдано сообщение об ошибке "ФАЙЛ НЕ ОТКРЫТ", и переключение не сработает.

Ограничения и особенности аппарата потоков.

I. Максимальное наполнение потока вывода таково:
два файла + экран + бумага,
три файла.

2. При ошибке обмена с файлом, все потоки ввода-вывода, которые настроены на него, автоматически отключаются.

3. При выдаче на экран управляющие символы выполняют возложенную на них дополнительную функциональную нагрузку. Это значит, что вставкой в выдаваемый текст этих символов можно управлять цветом, миганием и т.п. При выводе на бумагу или в файл все управляющие символы рассматриваются как обычные литеры.

РАЗДЕЛ IO. ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ И ОРГАНИЗАЦИЯ ДИАЛОГА

IO.I. Графические возможности ЭВМ "АГАТ"

Все описанные в предыдущих разделах средства программирования языка РАФИРА помогают использовать основные возможности вычислительной машины: хранение и обработку данных, в частности, арифметические вычисления, работу с текстами. В этом разделе вы познакомитесь еще с одной формой использования ЭВМ - обработкой графических изображений.

Экран телемонитора ЭВМ "АГАТ" можно рассматривать как прямоугольное поле, разлинованное наподобие тетрадного листа на клетки. Рисовать можно только закрашивая клетку целиком в тот или иной цвет, т.е. любое изображение можно представить на экране лишь приближенно (попробуйте нарисовать на клетчатой бумаге какой-нибудь рисунок и закрасьте его внутренние клетки).

Размеры отдельной клетки и возможные цвета раскраски зависят от графического режима работы "АГАТА". Всего существует 5 таких режимов:

I. Графика высокого разрешения (ГВР). Экран делится на 256x256 клеток (фактически они выглядят точками). Каждая точка может быть только черной или белой.

2. Графика среднего разрешения (ГСР). Экран состоит из I28xI28 клеток 8 различных цветов.

3. Графика низкого разрешения (ГНР). 64x64 клетки 8 цветов.

3. Графика цветных символов (ГЦС). Экран состоит из 32x32 клеток, однако, в каждой клетке может быть изображен символ (буква, цифра и т.д.). При этом возможны три вида изображения:
нормальный - цветной символ на черном фоне,
инверсный - черный символ на цветном фоне,
мигающий - попеременная смена нормального и инверсного символов.

4. Графика нецветных символов (ГНС): 64x32 черных символа на белом экране или столько же белых символов на черном экране. Изображение в последнем режиме хорошо выглядит только на черно-белом мониторе.

Далее для удобства будем называть элементарную клетку каждого графического режима просто точкой.

Содержимое экрана в том или ином режиме постоянно хранится в ОЗУ машины в специальной кодировке. Это позволяет хранить, изменять, читать изображение программой и, в принципе, запоминать их на диск и загружать с него. Возможность ЭВМ показывать на экране содержимое различных участков памяти в различных режимах позволяет одновременно обрабатывать несколько изображений. В символьном ре-

Л-Т-15227 Подписано в печать 13/VIII-85
Формат бумаги 60x84/16. Бумага писчая. Ротапринт.
Объем 1,45 п.л. Тираж 250 экз. Заказ №153 Бесплатно.
Ротапринт НИИШОТСО АПН СССР. 103062, Москва, Лялин п., д. 3а.

МИНИСТЕРСТВО РАДИОПРОМЫШЛЕННОСТИ СССР

АКАДЕМИЯ НАУК СССР
ОРДЕНА ЛЕНИНА СИБИРСКОЕ ОТДЕЛЕНИЕ
ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

УТВЕРЖДЕН
3533847.00042-01 33 01-ЛУ

ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ АВТОМАТИЗАЦИИ ШКОЛЬНОГО УЧЕБНОГО ПРОЦЕССА (ШКОЛЬНИЦА)

РУКОВОДСТВО ПРОГРАММИСТА

3533847.00042-01 33 01

Часть-2

1985

3533847.00042-01 33 01

жиме кроме рисования можно организовать диалог с пользователем, отведя под него весь экран или некоторую его часть.

Графические возможности, предоставляемые ЭВМ, значительно расширяют класс решаемых с ее помощью задач, позволяют повысить наглядность при программировании моделей реального мира. В этом легко убедиться, познакомившись с некоторыми из стандартных пакетов системы "Школьница".

10.2. Управление графикой и диалогом в Рапире. Процедура R 1.

Структура графической памяти в Рапира-интерпретаторе такова:

большая графическая страница для работы в режимах ГСР и ГВР,

5 малых графических страниц для работы в режимах ГНР, ГЦС и ГНС; причем 2,3,4 и 5 страницы располагаются в области памяти большой страницы – при переходе, например, в режим высокой графики их содержимое может испортиться.

Содержимое одной из этих страниц постоянно отображается на экране в некотором режиме.

Одна из 5 малых страниц является также текстовой страницей – носителем протокола диалога пользователя с ЭВМ. Диалог может вестись в одном из двух режимов: ГЦС или ГНС.

Одна из 6 графических страниц является текущей: с ней работают все процедуры и фукции графического комплекта "ШПАГА".

Особо отметим, что все три страницы – показываемая, текстовая и текущая графическая – могут быть различными! Это значит, что можно не видеть на экране ни протокола своего диалога с ЭВМ, ни того, что рисует программа.

Такая разветвленная структура страниц и их использования нуждается в простом и достуном управлении. Для этого в Рапире предусмот-

рена стандартная процедура РЖМ, позволяющая переключать графику и диалог с одной страницы на другую, показывать страницы в любом режиме и др.

Процедура РЭМ имеет собственный язык команд. Ее единственный параметр - текст произвольной длины, в котором через пробелы или запятые перечислены графические команды. Каждая графическая команда имеет такой вид:

<режим-и-адресат><действие>

После <режим-и-адресат> задает графическую страницу и режим исполнения указанных далее действий. Здесь возможны следующие варианты:

Т*i* - одна из 5 малых страниц в режиме ГЦС (*i*=I,2,3,4,5)

У*i* - одна из 5 малых страниц в режиме ГНС (*i*=I,2,3,4,5)

Н*i* - одна из 5 малых страниц в режиме ГНР (*i*=I,2,3,4,5)

С - большая страница в режиме ГСР,

В - большая страница в режиме ГВР,

* - текущая графическая страница в текущем графическом режиме. Поле <действия> определяет, что надо выполнить с указанным адресатом:

О - очистить черными точками указанного режима,

П - показать на экране,

Г - переключить графику (объявить текущей графической страницей),

Д - переключить диалог (объявить текстовой страницей) и показать на экране,

К - комбинировать с текстовой страницей.

Примечания и пояснения.

I. Черной точкой ГНС считается пробел, а ГЦС - черный пробел. Поэтому, если в ГЦС текстовая страница совпала с очищаемой, то при перемещении стрелками по очищенному полю курсор не виден, т.к. он имеет цвет того символа, на который указывает.

2. При переключении диалога текущий цвет вывода и содержимое новой текстовой страницы сохраняются, но устанавливается стандартное окно (см. I0.3).

3. При переключении графики устанавливается белый цвет отрисовки, кроме того, в ГЦС символом отрисовки устанавливается инверсный пробел, а в ГНС - " * ".

4. Из-за особенностей ЭВМ в режиме ГНС I,3,5 страницы выдаются только в белыми символами на черном фоне, а 2 и 4 страницы - наоборот.

5. Комбинированный режим означает, что в нижней части экрана вместо изображения указанной страницы выдаются 6 нижних строк текстовой страницы в том режиме, в котором идет диалог. В силу особенностей реализации этого режима звук и пауза в нем выполняются дольше, чем положено, сам же режим сбрасывается при обращении к дисководу. Нормальный сброс комбинированного режима - показ любой страницы по команде процедуры РЖМ.

6. Внимание! Комбинация "*Д" понимается как " переключить диалог на текущую текстовую страницу" (а не на графическую, как следует из описания), и может быть использована для показа текстовой страницы из программ.

Для облегчения работы с графикой введен показ попеременно текстовой и текущей графической страниц при каждом нажатии функциональной клавиши "8" во время ожидания системой ввода символа.

Примеры использования процедуры РЕЖ:

РЕЖ ("ТЦД,СТОП"); - переключить диалог на первую страницу в режиме цветных символов, переключить графику на большую страницу в режиме ГСР, очистить ее и показать на экране;

РЕЖ ("*О,Н2К"); - очистить текущую графическую страницу и показать на экране вторую страницу в режиме низкой графики, комбинированную с текстовой страницей.

I0.3. Графический комплект "ШАГА".

Для отрисовки изображений в графическом режиме в Рапире используется подмножество графического комплекта "ШАГА" (Школьный Пакет Адаптированных Графических средств), разработанного с ориентацией на обучение школьников работе с машинной графикой.

Координаты

Экран в графическом режиме рассматривается как часть непрерывной координатной плоскости. При установке режима начало координат автоматически помещается в левый нижний угол экрана, координатные оси направляются вправо и вверх, единицей масштаба по осям считается одна точка текущего режима.

В связи с тем, что физическая ширина экрана больше его высоты, изображение получается не изометрическим. Чтобы установить изометрический режим, достаточно в процедуре РЕЖ при установке режима указать не "Т", а "ТМ".

Начало координат и масштабы по осям могут быть изменены в программе. Для этого служат процедуры

ОТСЧЕТ (Нх,Ну); и МТБ (Мх,Му);

Координаты Нх и Ну - абсолютные, в точках текущего режима, Мх и Му - целые числа от 0 до 255. 0 означает отсутствие масштабирования по указанной оси; другое значение - это число 256-х долей нового масштаба по указанной оси. Таким образом, координаты точки (Х,У), указанной в процедурах отрисовки графических элементов преобразуются по формулам:

Х' =Нх + Х * Мх / 256

У' =Ну + У * Му / 256 (если масштаб ненулевой)

причем допускаются и отрицательные координаты.

Ограничение: независимо от масштабов абсолютное значение координаты не должно превосходить 5II.

Примечание: здесь и далее в процедурах "ШАГИ" дробные числовые параметры обрезаются до целых.

Процедуры установки.

В процедурах "ШАГИ" используются такие данные, как текущие цвет, символ и точка. Цвет и символ определяют раскраску каждой точки графического элемента, который рисует та или иная процедура.

I. Текущий цвет имеет смысл для всех режимов, кроме ГНС и задается процедурой

ЦВЕТ (N);

Параметр N должен быть целым числом от 0 до 255 (здесь и далее в процедурах "ШАГИ" дробные числовые параметры округляются до целых). в режимах ГНР, ГСР и ГВР его смысл таков:

0 - черный	4 - синий
I - красный	5 - фиолетовый
2 - зеленый	6 - голубой
3 - желтый	7 - белый

и далее через 8 цвета повторяются.

В режиме ГЦС цвет задается так:

0-7 – инверсный символ,

8-I5 – мигающий символ,

I6-23 – инверсный символ,

24-3I – нормальный символ.

Внутри каждой 8-ки цвета располагаются так же, через 32 раскраски повторяются.

При установке нового режима автоматически устанавливается белый цвет (в ГЦС – белый нормальный).

2. Текущий символ имеет смысл только в символьных режимах. Он задается процедурой

СИМРИС (Л);

Л– это литера, содержащая символ, которым будут рисоваться графические элементы (в сочетании с текущим цветом).

При установке нового режима текущим символом автоматически устанавливается пробел для ГЦС и "х" для ГНС.

3. Определить цвет точки на экране можно с помощью функции ЭКЦВ(Х,У), которая выдает номер цвета в описанном выше виде, причем все одинаковые цвета сводятся к цвету с наименьшим номером.

4. Текущая точка – это точка, в которой закончилась отрисовка последнего графического элемента. Она устанавливается каждой процедурой "ШАГИ" и используется в процедуре ЛУЧ для определения точки, которой надо провести линию. При установке нового режима текущей точкой является начало координат.

Графические процедуры

I. ТЧК (Х,У)

Раскрашивает точку с указанными координатами в текущий цвет и символ, если надо).

2. ЛИН (XI,YI,X2,Y2)

Рисует линию из точки (XI,YI) в точку (X2,Y2). Последняя становится текущей.

3. ЛУЧ (Х,У);

Рисует линию из текущей точки в точку с указанными координатами. Последняя становится текущей.

4. ПРЯМ (XI,YI,X2,Y2);

Рисует прямоугольник, диагональ которого определяется координатами

РАЗДЕЛ II. МОДУЛИ И ИСПОЛНИТЕЛИ

II.I. Об одном недостатке аппарата процедур в РАПИРЕ.

В разделе 6 при знакомстве с аппаратом процедур в РАПИРЕ вы узнали о возможности локализовать имена, т.е. ограничивать время и область существования их самих и связанных с ними объектов. Иными словами, каждая процедура (будем и в дальнейшем для краткости объединять процедуры и функции) обладает своей локальной программной средой, доступ к которой вне этой процедуры невозможен.

Наличие локальной среды – очень удобное свойство процедуры. Она позволяет скрыть особенности работы процедуры от программы – пользователя.

Но при создании больших программ одной процедуры бывает недостаточно: требуется несколько взаимозависимых и информационно связанных процедур. Описывая пакет процедур, работающих с общими данными, мы вынуждены представлять эти данные глобальными именами, хотя может возникнуть необходимость и их спрятать от пользователя или программы, вызывающей данный пакет.

Например, в пакете процедур, имитирующем движение лунохода,

может

могут оказаться процедуры ВПЕРЕД, НАЗАД, ВЛЕВО, ВПРАВО. Общими данными для них могут быть модель лунного ландшафта, текущие координаты лунохода, направление его движения и т.д. Понятно, что некоторые из этих данных используются исключительно для организации взаимодействия процедур пакета, о которых его пользователю знать не обязательно.

Отсюда возникает дополнительное требование к аппарату процедур: уметь создавать локальную среду для нескольких процедур так же, как мы создавали ее для одной, т.е. объявлять имена, доступные сразу из нескольких процедур и только из них. Это позволит спрятать от пользователя излишние "тонкости" работы целого пакета процедур.

Такую возможность в РАПИРЕ предоставляет аппарат модулей.

Примечание. Аппарат модулей описывается в настоящей инструкции чисто с практической точки зрения, чтобы научить использовать их для описания исполнителей в РОБИКЕ, которые как раз и действуют в некоторой замкнутой модели реального мира (см. II.7). Поэтому здесь приводятся упрощенное определение модуля и только самые основные его свойства. Более полное и точное описание модуля, его место в логике языка, свойства и возможности можно найти в каноническом описании.

II.2. Модуль

Модуль в РАПИРЕ – это пакет процедур и функций со скрытой внутренней структурой, в котором выделены имена, доступные во всех процедурах пакета и недоступные (или с ограниченным доступом) вне его.

Поясним это определение описанием свойств модуля и указанием их отличий от свойств процедуры.

1. Расширение локальных свойств. Помимо локальных имен каждой из процедур существуют их общие имена, локальные во всем пакете. Этим выполняется требование, выдвинутое в предыдущем параграфе.

2. Статичность. Локальные имена модуля известны только внутри описанных в нем процедур. Это значит, что имена в модуле локализованы по статической вложенности описаний (а не по динамической цепочке вызовов, как в процедурах). Например, если процедура из модуля вызывает процедуру, описанную вне его, то в последней локальные имена модуля недоступны. В остальном общепринятые правила локализации остаются в силе и для процедур модуля.

3. Независимость. Модуль не является объектом языка и не связан ни с каким объектом (процедура существует в программе как объект языка). Поэтому модуль не может быть присвоен имени, вызван, введен, выведен и т.д. Единственные действия, предусмотренные для работы с модулем в целом – это его включение и выключение, описанные ниже.

4. Способы доступа. Некоторые локальные имена модуля, в том числе имена входящих в него процедур, могут быть открыты для внешнего доступа (доступ к локальным именам процедуры невозможен). Такие имена называются открытыми.

5. Постоянство существования. Модуль может быть включен или выключен программно и все время от включения до выключения находится в памяти со всеми локальными именами и их значениями. (локальные имена процедуры существуют от вызова до выхода из нее).

6. Глобальность. Модуль – это глобальное образование. Он не может быть локализован в какой-либо процедуре независимо от места его

включения. Доступ к открытым переменным модуля возможен с любого уровня вложенности вызовов.

7. Управление. Пользователю может быть предоставлено полное или частичное управление модулем (порядок работы процедуры строго определяется описанием процедуры; пользователь может вмешаться в ход исполнения только в режиме приостанова).

Это значит, что возможны две крайности при написании пакетов-модулей. Программа типа "вопрос-ответ" сама ведет диалог и не передает управления пользователю. Полное управление предоставляет простой набор процедур, которые пользователь может вызывать в программе по своему усмотрению. Возможны и промежуточные варианты.

8. Скрытая структура. Основное отличие модуля от пакета процедур – его целостность. С точки зрения пользователя модуль так же неделим, как любая операция. Его внутренняя структура, в т.ч. локальные имена, абсолютно недоступны даже в режиме приостанова (локальные имена процедуры в режиме ее приостанова доступны наравне с глобальными). Подробнее об этом см. II.6.

II.3. Описание модуля

Описание модуля представляет собой текстовый файл в ДЗУ, содержащий специальным образом оформленную информацию о данном модуле. Названием модуля называется имя этого файла.

Подготовить описание можно, например, с помощью автономного Редактора текстов или Редактора программных текстов, встроенного в интерпретатор РАПИРЫ.

Внимание! Описание модуля – это не процедурный блок. Поэтому его нельзя транслировать из Редактора по директиве "КОНЕЦ ОПИСАНИЯ".

Рассмотрим структуру описания модуля.

Модуль описывается с соблюдением всех правил записи лексем и предписаний, принятых в РАПИРЕ (см. 3.4, 4.I). Синтаксис описания модуля представлен диаграммой 4I.

Как видно из диаграммы, описание отличается от пакета процедур только наличием заголовка модуля.

Заголовок состоит из двух основных частей.

I. Список имен в конструкции ИМЕНА определяет локальные имена модуля, общие для всех описанных в нем процедур (как минимум, в нем должны быть указаны имена самих этих процедур и функций).

Каждое из имен, указанных в списке, доступно только в пределах модуля (модуль является областью действия описаний этих имен).

Имена, после которых стоит слово "ДОСТУПНО", становятся открытыми. От глобальных они отличаются тем, что их значения являются объектами модуля, как локальной среды, т.е. возникают и исчезают вместе с ним. Кроме того, возможность их использования несколько ограничена (см. II.5).

2. Конструкции

СТАРТ имя;

и

ФИНИШ имя;

определяют стартовую и финишную процедуры модуля. Эти процедуры описываются здесь же в описании модуля.

Стартовая процедура вызывается в момент включения модуля и может использоваться для выполнения предварительных действий (по усмотрению автора модуля): присваивания начальных значений, открытия рабочих файлов, выдачи титульной и справочной информации и т.д.

Финишная процедура вызывается при выключении модуля и служит для выполнения аналогичных завершающих действий: закрытия файлов, очистки экрана, выдачи результатов работы модуля и т.д.

Стартовая и финишная процедуры (не функции!) не должны иметь параметров, т.к. непонятно, какие параметры подавать им на вход.

Конструкции СТАРТ и ФИНИШ являются необязательными, т.е. разрешается не предусматривать стартовую и финишную процедуры (например, если начальные или конечные действия не нужны).

Наконец, можно вообще не описывать в модуле никаких процедур. Но такие модули интереса не представляют, и потому в дальнейшем не рассматриваются.

В описываемых версиях ввод из ДЗУ внутри модуля запрещен. Поэтому подзагрузка невозможна, и все процедуры модуля следует помещать в одном файле.

II.4. Включение и выключение модуля

Включение и выключение модуля осуществляется по предписаниям

ВКЛ МОДУЛЬ имя;

ВЫКЛ МОДУЛЬ имя;

Включение модуля заключается в следующем:

I. Проверка, не включен ли уже модуль с указанным в предписании ВКЛ именем. Если включен, выдается сообщение об ошибке.

2. Поиск в библиотеке на текущем активном ДЗУ текстового файла с указанным именем.

3. Анализ содержимого этого файла как описания модуля с диагностикой замеченных синтаксических ошибок (см. Приложение 4) и формирование соответствующих объектов в программной среде.

4. Вызов стартовой процедуры, если она предусмотрена.

5. Возврат управления к пользователю.

Включенный модуль называется активным.

Выключение модуля заключается в следующем:

I. Вызов финишной процедуры, если она предусмотрена.

2. Уничтожение модуля: т.е. всех его объектов и доступа к его именам.

Внимание! Имя, указываемое в предписаниях ВКЛ и ВЫКЛ, не имеет ничего общего с именами в языке. Оно обозначает только имя файла с описанием того или иного модуля и используется в системе, чтобы различать уже включенные модули. Следующий пример иллюстрирует это:

```
# I2 -> ЛЮБОЕ_ИМЯ;
# ВКЛ МОДУЛЬ ЛЮБОЕ_ИМЯ;
# ? ЛЮБОЕ_ИМЯ;
   I2
# ЕСЛИ ЛЮБОЕ_ИМЯ ВИДА 0 ТО I23 -> ЛЮБОЕ_ИМЯ ВСЕ;
# ВЫКЛ МОДУЛЬ ЛЮБОЕ_ИМЯ;
# ? ЛЮБОЕ_ИМЯ;
   I23
```

Чтобы не путаться в терминах, будем в дальнейшем говорить о названии модуля, которое является именем файла на диске, содержащем описание модуля. Синтаксически название модуля записывается так же, как и имя в языке.

В описываемых реализациях допустимо до I6 одновременно активных модулей.

II.5. Доступ к именам модуля

Открытые имена (т.е. описанные в модуле со словом "ДОСТУПНО") могут быть использованы в любой программе, в любой процедуре независимо от места ее вызова. Обращение к открытому имени имеет вид:

название модуля, открытое имя

например:

ШАГА'КООРД_Х

ДЕЖ/РИК'ОКНО

Открытые имена могут быть использованы в предписаниях языка в полях "выражение" и " уточненное имя", например, в операциях, предписаниях вывода и присваивания (в обеих частях!). Обратите внимание, что открытые имена модуля нельзя использовать в предписаниях "ОТКРЫТЬ" и "ВВОД ИЗ ДЗУ", нельзя изменять их защиту и признак прокрутки.

Невозможность изменить защиту открытого имени модуля вне модуля позволяет сделать часть из них защищенными от повторного присваивания, т.е. разрешить только пользоваться в программе значением имени, не разрешая изменять его. Для этого достаточно поставить на имя защиту внутри модуля (это допустимо). Любое присваивание такому имени будет рассматриваться как ошибка, а снять защиту иначе, как из модуля, нельзя.

Открытые имена удобно использовать для организации взаимодействия модуля с программой-пользователем, например, в качестве носителей состояния описываемого с помощью модуля процесса.

Примечание. Вызывать процедуры модуля, открытые для внешнего доступа, через штрих неудобно. Поэтому рекомендуется в стартовой процедуре предусмотреть присваивание этих процедур некоторым глобальным именам, которые затем и использовать в программе. Тогда в финишной процедуре желательно присвоить им пустые значения.

II.6. Неделимость модуля

РАПИРА, как язык программирования, обеспечивает стандартные средства обработки данных, представленных в виде объектов языка. Действия над объектами обозначаются операциями или предписаниями, они элементарны и представляют собой лишь "кирпичики", из которых можно строить более сложную обработку данных.

Описывать с помощью языковых средств дополнительные возможности и использовать их в своих программах программисту позволяет аппарат процедур.

Основное назначение аппарата модулей - максимально приблизить использование этих дополнительных возможностей к уровню языка. В II.8 будет показано, как с помощью модулей в языке РОБИК создаются новые предписания. В РАПИРЕ, имеющей развитую структуру выражений и лаконичный синтаксис, для использования искусственных возможностей достаточно вызова процедур и функций.

Полному сходству исполнения языковых операций и предписаний с исполнением процедур и функций пользователя мешают, как ни странно, отладочные возможности языка, а именно - возможность приостанова.

Мы можем остановить процедуру и получить доступ к ее локальным именам, т.е. раскрыть внутреннюю структуру работающей программы. Например, было бы нежелательно оказаться в режиме приостанова функции синус при отладке использующей ее программы: ведь сама-то она отлажена.

Поэтому модуль считается целостным и неделимым, он становится полным подобием "черного ящика", внутренняя структура которого неизвестна. Это отражается в следующем.

I. Имена, объявленные в модуле, - как локальные имена всего модуля, так и локальные имена описанных в нем процедур - пользователю неизвестны: они не выдаются в каталоге имен.

2. Это же правило действует и в режиме останова (останова по ошибке) любой из процедур модуля.

3. Точное место в модуле, где произошел останов, в диагностическом сообщении не указывается, например:

ОСТАНОВ В "МОДУЛЬ"

ОСТАНОВ В "МОДУЛЬ" В СТРОКЕ I5 "РАКЕТА"

(здесь "РАКЕТА" - это процедура, которая обратилась к модулю).

4. Прокрутка имен, объявленных в модуле, и след его процедур не ведутся, если они были включены вне модуля (т.е. если того не захотел автор модуля).

Единственным признаком, который не может контролироваться системой и по которому можно судить о работе модуля, остается состояние открытых имен модуля и внешних устройств. Например, модуль "ЧЕРТЕЖНИК" к моменту останова мог успеть нарисовать линию, но не успеть исправить координаты последней отрисованной точки.

Таким образом, хотя модуль и может быть остановлен, программная среда, доступная пользователю в этот момент не засоряется ненужной информацией, которая обратилась к модулю.

По директиве ПУСК будет продолжено выполнение остановленной процедуры модуля и всех вызвавших ее процедур.

II.7. Исполнитель в РОБИКЕ

Описанный в предыдущих параграфах аппарат модулей разрабатывался в основном для реализации исполнителей в другом языке системы - РОБИКЕ.

В силу специфики учебного языка основными "действующими лицами" в РОБИКЕ являются исполнители. Исполнитель - это гипотетический робот, имеющий собственное множество предписаний (МПИ) и действующий в определенной операционной обстановке (модели реального мира). Пользователь выступает в роли оператора, подающего роботу предписания из его МПИ, которые тот исполняет.

Исполнитель и его операционная обстановка описываются программой (пакетом процедур) на РАПИРЕ, оформленной в виде модуля. В этой программе можно использовать все средства, предоставляемые РАПИРОЙ (кроме ввода из ДЗУ). Здесь же определяется МПИ, указывается, какая процедура модуля интерпретирует то или иное предписание.

При реализации исполнителя особенно важны три свойства модуля.

I. Целостность. Пользователь может с полной уверенностью считать, что он имеет дело с роботом: структура моделирующей программы надежно скрыта.

2. Постоянность существования. Исполнитель включен и всегда готов к работе, даже если с ним в данный момент не работают: робот ждет команду. Так можно работать одновременно с несколькими исполнителями.

3. Глобальность. Предписание исполнителю можно подавать с любого уровня вложенности процедур. Это позволяет создавать и хранить программы, написанные для конкретных роботов на их языке.

II.8. Описание синтаксиса исполнителя

РОБИК представляет собой язык с динамическим синтаксисом: в нем разрешено использовать предписания всех активных исполнителей. Эти предписания автор исполнителя может конструировать по следующим правилам.

I. В предписании допустимы

ключевые слова, записываемые по правилам записи имен,

имена,

выражения,

двоеточия и запятые, разделяющие имена, выражения и ключевые слова.

2. Ключевое слово должно быть не длиннее 20 символов.

3. Предписание должно начинаться с ключевого слова.

4. При включении исполнителя проверяется, нет ли в его МПИ предписания, которое начинается с того же ключевого слова, что и некоторое предписание языка или другого активного исполнителя.

В остальном правила записи лексем и ограничения на их длину приняты те же, что и в РОБИКЕ.

При описании исполнителя в заголовке пакета-модуля появляется еще один раздел: описание синтаксиса. Его вид представлен диаграммой 42.

В квадратных скобках через точку с запятой перечисляются описания всех предписаний, образующих МПИ данного исполнителя. Описание отдельного предписания начинается с имени процедуры пакета, которая его исполняет. Через запятую в апострофах перечисляются возможные варианты записи этого предписания. Например:

```
[ СУММА: "СЛОЖИТЬ  <ВЫР>  С  <ВЫР>  И ПРИСВОИТЬ
         <ИМЯ> ",
       "СУММА:  <ВЫР>  ,  <ВЫР>  ,  <ИМЯ>  ";
  ЧАСТНОЕ: "РАЗДЕЛИТЬ  <ВЫР>  НА <ВЫР>  И
         ПРИСВОИТЬ <ИМЯ>  ";
       "ЧАСТНОЕ: <ВЫР>  ,  <ВЫР>  ,  <ИМЯ>" ]
```


Переменные поля " <ВЫР> " и " <ИМЯ> " в тексте предписания означают, что в этой позиции должны стоять выражение или имя, соответственно. Пробелы имеют смысл только там, где возможно принятие двух лексем за одну. Точки с запятыми в конце предписания указывать не надо.

На процедуру, реализующую то или иное предписание накладываются следующие ограничения.

I. Это должна быть именно процедура, а не фукнция, т.к. предписание не выдает результата. Если результат необходим, его следует поместить в какое-нибудь имя, как в примере выше.

2. Она должна иметь столько параметров, сколько переменных полей указано в тексте предписания; поле с номером N соответствует параметру с номером .

Внимание! Если указывается несколько допустимых вариантов предписания, то они должны содержать одинаковое количество переменных полей, причем соответствующие поля должны иметь один и тот же смысл.

Так, если

```
ВЫЧЕСТЬ <ВЫР> ИЗ <ВЫР> И ПРИСВОИТЬ <ИМЯ>
```

И

```
РАЗНОСТЬ: <ВЫР> , <ВЫР> , <ИМЯ>
```

это две формы одного предписания, то

```
ВЫЧЕСТЬ А ИЗ В И ПРИСВОИТЬ Е;
РАЗНОСТЬ: А,В,Е;
```

вычисляет А-В.

3. Если требуется передать выработанное модулем значение, то в соответствующем поле предписания надо требовать имя и использовать выходные и возвратные параметры. Если достаточно иметь значе-

ные имени или выражения, следует использовать входные параметры.

И в заключение отметим одно важное различие между использованием модулей в РАПИРЕ и исполнителей в РОБИКЕ. МИИ становится известно системе только после включения исполнителя. Это значит, что ни одна процедура, содержащая предписания исполнителя не будет понята системой, пока он не включен. Такая ситуация может возникнуть, например, при выходе из Редактора по директиве "КОНЕЦ ОПИСАНИЯ".

II.9. Методические рекомендации по написанию исполнителей

При разработке исполнителей для РОБИКА полезно знать следующие положения.

1. При описании модуля можно пользоваться всеми возможностями РАПИРЫ, даже если использоваться он будет в конфигурации системы с одним из более узких концентров.

2. В описываемых версиях реакция на ошибки - системная. Продолжить исполнение программы после останова по ошибке невозможно. Поэтому следует обратить особое внимание на разбор возможных ошибочных ситуаций и их "ручную" обработку. Достаточно легко, например, организуется проверка видов значений передаваемых параметров, контроль выхода за область допустимых значений при работе с экраном и при выполнении арифметических операций над текстами.

3. Следите за тем, чтобы модуль (исполнитель) при выключении закрывал все использованные файлы, уничтожая ненужные значения глобальных переменных, отключая потоки ввода и вывода, настроенные на внешние устройства, переходил в стандартный режим диалога, т.е. оставлял порядок в программной среде.

ПРИЛОЖЕНИЕ I. ЛЕКСИКА ЯЗЫКА И СООБЩЕНИЯ СИСТЕМЫ

I. Ключевые слова языка РАПИРА

БИС	БУМАГУ	В	ВИДА	ВВОД
ВКЛ	ВСЕ	ВСЕХ	ВЫБОР	ВЫВОД
ВЫКЛ	ВЫХОД	ДАННЫХ	ДЗУ	ДЛЯ
ДО	ЗАКРЫТЬ	ЗАПЕРЕТЬ	ЗАЩИТУ	И
ИЗ	ИЛИ	ИМЕН	ИМЕНА	ИНАЧЕ
КАК	КАТАЛОГ	КНЦ	КОНТРОЛЬ	МЕНЮ
НА	НЕ	ОТ	ОТКРЫТЬ	ОТПЕРЕТЬ
ОТЛАДКУ	ПАКЕТ	ПОВТОР	ПОЗИЦИЯ	ПОКА
ПРОКРУТКУ	ПРОТОКОЛ	ПРОЦ	ПУСК	РАЗ
РАЗА	РАПИРА	РЕЗ	РОБИК	СЛЕД
СТЕРЕТЬ	СТОП	СТРОК	ТЕКСТОВ	ТО
ФАЙЛ	ФАЙЛА	ФАЙЛОВ	ФУНК	ЧАСТ
ШАГ	ЭКРАН			

2. Названия режимов (информационная строка экрана)

Режим	Надпись в информ.строке
Основной	## РАСПИРА-АГАТ I.I ##
	## РОБИК-АГАТ I.I ##
Приостанова	>> ПАУЗА <<
Ост.по ошибке	>> О Ш И Б К А <<

4. Меню редактора

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x                                    x
x  ТЕКСТОВЫЙ РЕДАКТОР СИСТЕМЫ        x
x     "Ш К О Л Ь Н И Ц А"            x
x        НОВОСИБИРСК  I985           x
x                                    x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

КОНЕЦ ОПИСАНИЯ	КАТАЛОГ
ПЕРЕЙТИ К СТРОКЕ	СЧИТАТЬ ФАЙЛ
ЗАПОМНИТЬ ТЕКСТ	ЗАПИСАТЬ ФАЙЛ
ВЫДАТЬ НА БУМАГУ	СТЕРЕТЬ ФАЙЛ
ОЧИСТИТЬ БУФЕР	ДОПИСАТЬ ФАЙЛ
ВЫЙТИ	РАБОТА С ДЗУ 2

ИМЯ ФАЙЛА:ИМЯ-ТЕКУЩЕГО-ФАЙЛА

Прочие сообщения:

НОМЕР СТРОКИ(I) -

С КАКОЙ СТРОКИ (I) -

ДО КАКОЙ СТРОКИ (75) -

НАЖМИТЕ ЛЮБУЮ КЛАВИШУ

ПАМЯТЬ ПЕРЕПОЛНЕНА

ВЫ НАЖАЛИ "СБРОС"

ОШИБКА ОБМЕНА С ДЗУ

ИМЯ ФАЙЛА:

ТЕКСТ:

ПРИЛОЖЕНИЕ 2. СИМВОЛЫ.

В этом приложении описывается символьный набор системы "ШКОЛЬНИЦА", ввод и отображение символов, их свойства в различных режимах работы системы.

I. Понятие о символах и их обработке

Весь диалог пользователя с ЭВМ ведется с помощью символов. Символ - это минимальная единица текстовой информации в системе (в языке это литера). В системе "ШКОЛЬНИЦА" предусмотрена обработка 256 различных символов. Сопоставление им букв, цифр, знаков и даже некоторых действий позволяет организовать хранение и обработку текстовой информации в ЭВМ.

Символы выполняют три основные функции:

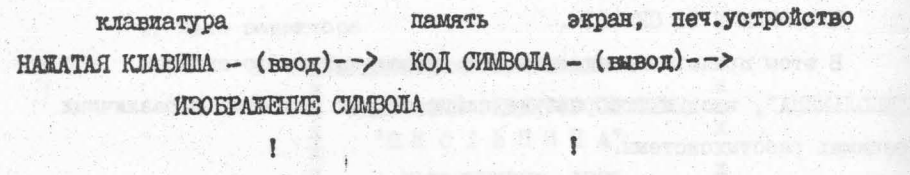
вводятся с клавиатуры ЭВМ (каждая клавиша, нажатая на том или ином регистре, посылает в машину код, рассматриваемый системой как код некоторого символа);

из них состоят хранящиеся в памяти текстовые данные;

выводятся на экран, печатающее устройство и т.п. (некоторым символам соответствует определенное изображение, возможно, разное на разных устройствах).

Кроме того, поступление с клавиатуры или на вывод некоторых символов рассматривается системой как сигнал пользователя или программы о необходимости выполнить дополнительные действия, предусмотренные в данном режиме работы. Такие символы часто называют управляющими.

Схематически процесс ввода, хранения и отображения символов, можно изобразить так:



Поскольку многие вводимые символы сразу же поступают на вывод, то есть символы, которые несут дополнительную нагрузку и на вводе, и на выводе.

Ниже приводится более подробное описание набора символов системы "ШКОЛЬНИЦА" и их функциональной нагрузки.

2. Клавиатура

Клавиатура обеспечивает ввод символов на 3 регистрах: основном (одиночно нажатые клавиши), регистрах "РЕГ" и "УПР". Нажатие любой клавиши, кроме "СБР" и "ПВТ", на одном из этих регистов вызывает ввод в машину некоторого символа.

В Таблице 2.1 показано, какой код выдает та или иная клавиша. Обратите внимание на следующие особенности:

а) существуют клавиши (например, пробел, цифры, стрелки и др.), которые "нечувствительны" к некоторым регистрам и выдают один и тот же символ;

б) клавиатура обеспечивает ввод не всех 256 символов; это может оказаться удобным в тех программах, где потребуются зарезервированные символы, которые нельзя ввести с клавиатуры;

в) некоторые клавиши "АГАТа" на регистре "УПР" вообще не выдают никакого символа;

г) каждая функциональная клавиша выдает тот же код, что и некоторая буквенная клавиша на регистре "УПР", например, клавиша перевода строки и "УПР-М".

Символы (клавиши), которые несут дополнительную функциональную нагрузку на вводе, приведены в таблице 2.3.

3. Программная среда

Алфавитом языка РАПИРА допустим лишь ограниченный набор символов (см.Диаграммы). Но в составе текстов могут встречаться все 256 символов. Задать в программе символ, который не вводится с клавиатуры, можно с помощью стандартной функции *АЛФ*. Она имеет один параметр – десятичный код требуемого символа – и выдает в качестве результата литеру, содержащую этот символ.

Наоборот, определить по символу его код можно с помощью стандартной функции КОД. Ее параметром должна быть литера. Результат – целое число от 0 до 255.

В текстовый файл можно вывести любой символ, кроме символа с кодом 0, – он считается признаком конца файла. При попытке его вывода он заменяется символом с кодом 135.

Редактором программных текстов не вводится и не обрабатывается символ с кодом 127. При загрузке файла с диска этот символ заменяется пробелом.

4. Экран и бумага

Вывод символа на экран и бумагу заключается в формировании изображения, соответствующего этому символу.

При выводе некоторых символов выполняются дополнительные действия (см.Таблицу 2.3). Как можно видеть, все управление цветом выдаваемых символов, очистка экрана и другие полезные возможности возложено на символы. Это позволяет легко управлять изображением на экране, вставляя в выводимые из программы тексты символы переключения.

При вводе на бумагу изображение некоторых символов может не совпадать с их изображением на экране. Это зависит от модели печатающего устройства. Например, D -I00 вместо "," печатает ромбик.

5. Управляющие клавиши основного диалогового режима

Находясь в основном диалоговом режиме, можно достаточно свободно использовать при наборе весь экран.

Клавиша РЕД устанавливает режим свободного перемещения по экрану, это значит, что с помощью клавиш-стрелок можно подвести курсор к любой точке экрана, не вводя при этом в машину ни одного символа. В этом режиме курсор на экране имеет вид мигающего знака " ^ ". Нажатие любой другой клавиши вызывает выход из этого режима.

Клавиши "вверх" и "вниз", нажатые вне этого режима, имеют тот же смысл.

Клавиша "->" кроме перемещения курсора дописывает в конец вводимой строки текущий символ с экрана. Пройдя по некоторому тексту на экране этой клавишей, можно таким образом, ввести его в машину целиком, не набирая заново.

Клавиша <- кроме перемещения курсора стирает последний символ во вводимой строке, оставляя его на экране. Имейте в виду, что стирается не всегда тот символ,на который указывает курсор, Если, например, в ситуации

ПРИМЕР НЕСООТВЕТСТВИЯ

ТЕКСТА ЭКРАНУ -

нажать клавишу "вверх" и пять раз клавишу <-, то из вводимой

строки исчезнут буквы "КРАНУ", а не "ЕСООТ", как это будет изображено на экране.

Клавиша F2 уничтожает на экране текущий символ, сдвигая при этом остаток экранной строчки влево. На освободившееся место вставляется пробел.

Клавиша F3 освобождает на экране одну позицию перед текущим символом. Остаток экранной строчки при этом сдвигается вправо и последний символ в ней пропадает.

Символ "УПР-L " очищает текстовое окно экрана и устанавливает курсор в его левый верхний угол.

Клавиши I, 2, 3, 4, 5, 6, 7, 0, . , = и соответствующие им символы управляют цветом вывода символов и фона на экран (см. Таблицу 2.3).

Клавиша 9 заполняет пробелами остаток экрана, начиная с текущей позиции курсора.

Клавиша 8 переключает экран на показ текущей текстовой и текущей графической страниц в зависимости от того, какая из них показана в данный момент.

Клавиша перевода строки сигнализирует машине о конце ввода пользователем очередной строки.

Клавиша F I является универсальной клавишей останова. Связанные с ней в системе действия уже упоминались в основном тексте.

Нажатие клавиши "УПР-X" означает отказ от ввода строки. При этом на экране появляется символ " \ ", строчка автоматически переводится, и выдается приглашение к новому вводу.

Клавиша "УПР-V " позволяет записать во вводимую строку любой управляющий символ, не выполняя связанных с ним действий. На экране этот символ высвечивается инверсным желтым цветом. Вставка обыч-

ного символа равносильна вводу самого символа. Нельзя вставить только символ перевода строки (код I4I, клавиша "УПР-М").

6. Нестандартное использование клавиатуры

Описанное в предыдущем параграфе использование клавиатуры и управляющих символов не является единственно возможным. Можно организовать диалог программы с пользователем через стандартную функцию КЛАВ, которая запрашивает с клавиатуры символ и возвращает в качестве результата его код. Все дополнительные действия в этом случае организуются программно.

Иное использование клавиатуры принято, например, в Редакторе (см.5.2 и Таблицу 2.4).

Таблица 2.I. Коды клавиш.

Клав.	Одна	РЕГ	УПР	Клав.	Одна	РЕГ	УПР
;+	I87	I7I	I87	Ч^	254	222	-
I!	I77	I6I	I77	Ш[	25I	2I9	-
2"	I78	I62	I78	Щ]	253	22I	-
3№	I79	I63	I79	ъ_	255	233	-
4;	I80	I64	I80	Ы Y	249	2I7	-
5%	I8I	I65	I8I	Ь X	248	2I6	-
6	I82	I66	I82	Э\	252	220	-
7'	I83	I67	I83	Ю@	224	I92	I28
8(	I84	I68	I84	Я Q	24I	209	-
9)	I85	I69	I85	/?	I75	I9I	I75
0	I76	I60	I76	">	I74	I90	I74
-=	I73	I89	I73	,<	I72	I88	I72
AA	I93	I93	I29	:*	I86	I70	I86
БВ	226*	I94*	I30	ПРОБЕЛ	I60	I60	I60
B W	I94	2I5	-	Ф,I	I44	I44	I44
Г G	23I	I99	I35	Ф.2	I45	I45	I45
Д D	228	I96	I32	Ф.3	I46	I46	I46

Продолжение табл.2.I

Клав.	Одна	РЕГ	УПР	Клав.	Одна	РЕГ	УПР
Е Е	I97	I97	I33	Ф.4	I47	I47	I47
К V	246	2I4	-	Ф.5	I48	I48	I48
З Z	250	2I8	-	Ф.6	I56	I56	I56
И I	233	20I	I37	Ф.7	I57	I57	I57
И J	234	202	I38	Ф.8	I58	I58	I58
К К	203	203	I39	Ф.9	I59	I59	I59
Л L	236	204	I40	Ф.0	I29	I29	I29
М М	205	205	I4I	Ф. .	I30	I30	I30
Н N	200	206	I42	Ф. =	I3I	I3I	I3I
О О	207	207	I43	F1	I32	I32	I32
П P	240	208	-	F2	I33	I33	I33
Р R	208	2I0	-	F3	I34	I34	I34
С S	I95	2II	-	↑	I53	I53	I53
Т T	2I2	2I2	-	↓	I54	I54	I54
У U	245	2I3	-	→	I49	I49	I49
Ф F	230	I98	I34	←	I36	I36	I36
Х H	2I6	200	I36	РЕД	I55	I55	I55
Ц C	227	I95	I3I		I4I	I4I	I4I

* Ввод зависит от положения регистра "РУС-LAT" на клавиатуре. При его переключении для буквенных клавиш первую и вторую колонки надо поменять.

Таблица 2.2 Вывод символов на экран

Код	160	176	192	208	224	240
0	!	0	@	P	Ю	П
1	!	1	A	Q	А	Я
2	!	"	2	B	R	Б	Р
3	!	#	3	C	S	Ц	С
4	!	$	4	D	T	Д	Т
5	!	%	5	E	U	Е	У
6	!	&	6	F	V	Ф	Ж
7	!	'	7	G	W	Г	В
8	!	(	8	H	X	Х	Ь
9	!	)	9	I	Y	И	Ы
10	!	*	:	J	Z	И	З
11	!	+	;	K	[	К	Ш
12	!	,	<	L	\	Л	Э
13	!	-	=	M	]	М	Щ
14	!	.	>	N	^	Н	Ч
15	!	/	?	O	-	О	,

Символы с кодами 128–159 (управляющие) на экран не выдаются. Их действия при выводе см. в Таблице 2.3.

Символы с кодами 0–127 выводятся так же, как символы с кодами 128–255.

Таблица 2.3. Управляющие символы (описание см. в 5)

Код	Клавиша	Время	Действия
129	0	вывод	Уст.выдачу нормальных символов
130	.	вывод	Уст.выдачу инверсных символов
131	=	вывод	Уст.выдачу мигающих символов
132	F1	ввод	Универсальная клавиша останова и прерывания
133	F2	ввод	Сдвиг остатка строчки экрана влево на одну позицию

Продолжение табл.2.3

Код	Клавиша	Время	Действия
134	F3	ввод	Сдвиг остатка строчки экрана вправо на одну позицию
135	УПР-G	вывод	Звуковой сигнал
136	<-	вывод*	Сдвиг курсора к предыдущему символу на экране
		ввод	Уничтожение последнего введенного символа
138	УПР-J	вывод	Переход к началу следующей строки
140	УПР-L	вывод	Очистка текстового окна экрана
141		вывод	Переход к началу следующей строки
		ввод	Конец строки ввода
144	1	вывод	Уст.красный цвет выдачи
145	2	вывод	Уст.зеленый цвет выдачи
146	3	вывод	Уст.желтый цвет выдачи
147	4	вывод	Уст.синий цвет выдачи
148	5	вывод	Уст.фиолетовый выдачи
149	->	вывод*	Сдвиг курсора к следующему символу на экране
		ввод	Ввод текущего символа с экрана
150	УПР-	ввод	Вставка упр.символа во вводимую строку
152	УПР-X	ввод	Отказ от вводимой строки
153	"вверх"	вывод*	Сдвиг курсора на строчку вверх
154	"вниз"	вывод*	Сдвиг курсора на строчку вниз
155	РЕД	ввод	Переход в режим свободного перемещения по экрану
156	6	вывод	Уст.голубой цвет выдачи
157	7	вывод	Уст.белый цвет выдачи
		ввод	Переключение текстовой и графической страниц
158	8	вывод	Очистка остатка текущей строки (в пределах окна)
159	9	вывод	Очистка остатка экрана (в пределах окна)

* Те же действия на вводе в режиме РЕД

Таблица 2.4. Управляющие клавиши Редактора

"вверх"	
"вниз"	Перемещение курсора по экрану
→	
←	
"пер.стр."	В режиме редактирования – вставка новой строки в режиме меню – конец ответа на запрос
I	Уничтожить текущий символ
2	Начать вставку символов
3	Перейти к началу следующей строки текста
4	Показать предыдущую страницу текста
5	Показать следующую страницу текста
6	Центрировать текущую строчку
7	Склеить текущую строку текста со следующей
8	Обрезать остаток текущей строки текста
9	Уничтожить текущую строку текста
F 3	Найти очередное вхождение запомненной цепочки символов
F 2	Переключение в уплотненный графический режим и наоборот
F I	Отказ от запроса в меню
РЕД	Переход из режима редактирования в режим меню и наоборот
УПР-V	Вставка в текст управляющего символа
УПР-А	В данной позиции цепочки поиска допустим любой символ

ПРИЛОЖЕНИЕ 3. СИНТАКСИЧЕСКИЕ ДИАГРАММЫ

В этом приложении приведены синтаксические диаграммы входных языков системы РАПИРА и РОБИК с учетом всех ограничений версии AI.I. Синтаксические диаграммы описаний языков соответствуют канонической версии.

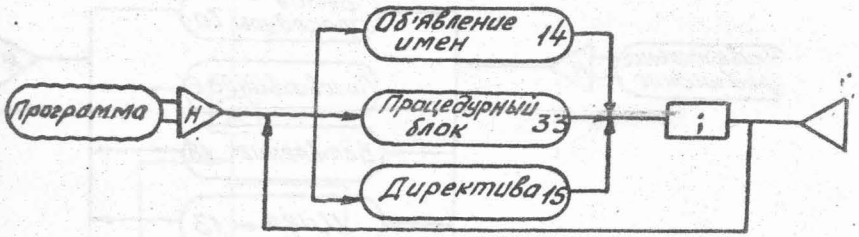


Диаграмма 1

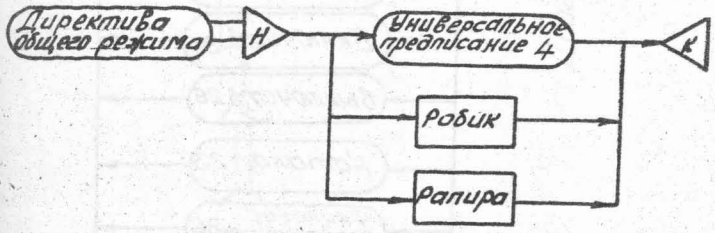


Диаграмма 2

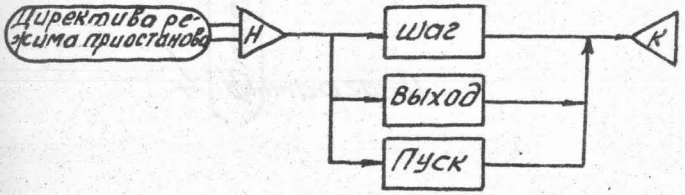


Диаграмма 3.

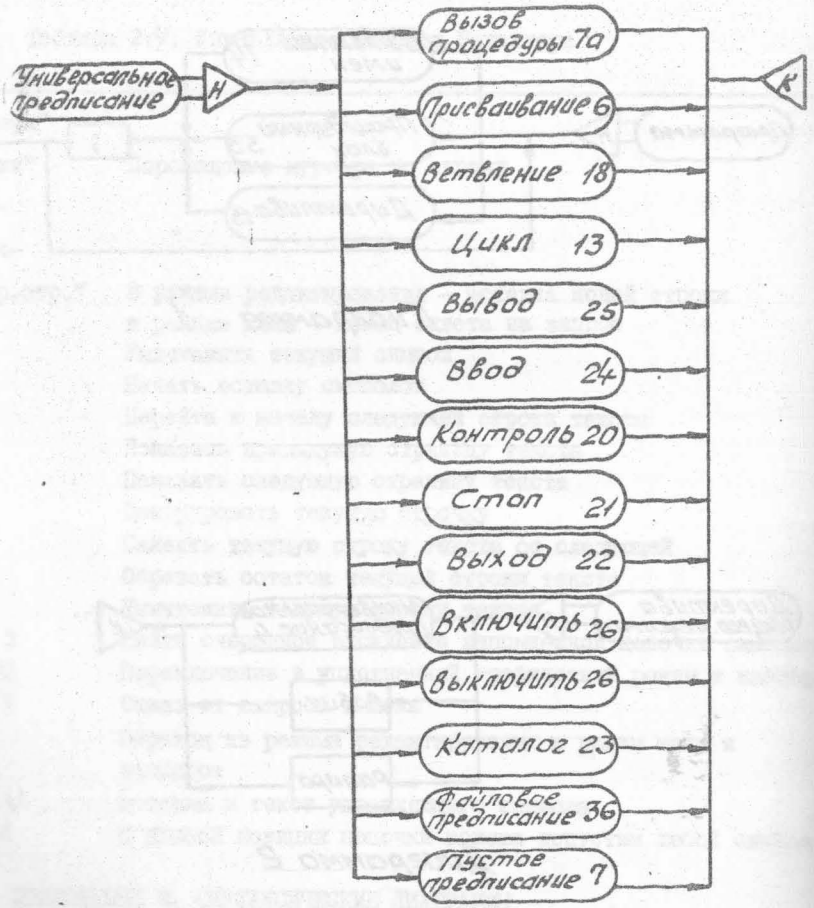


Диаграмма 4.

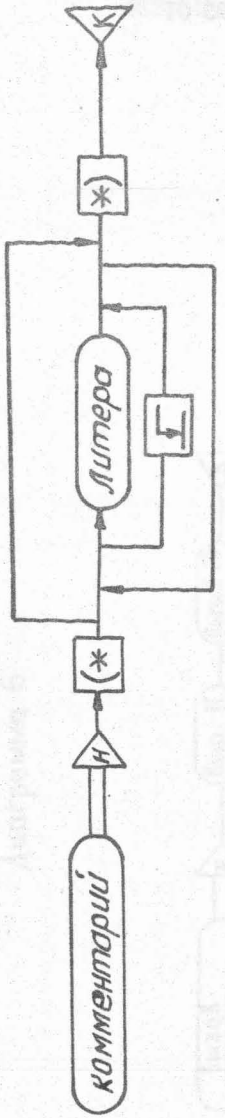


Диаграмма 5.

Диаграмма 7

Диаграмма 6

Диаграмма 7а

Диаграмма 8

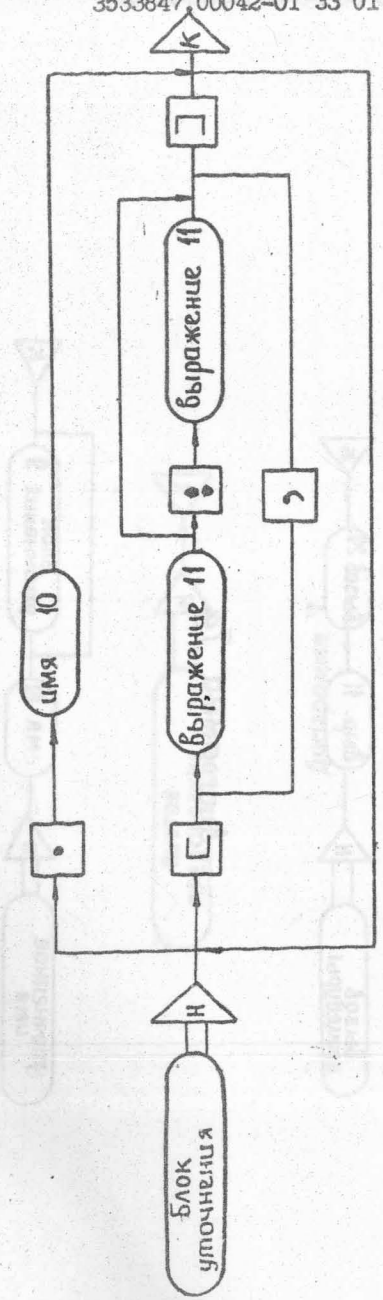


Диаграмма 9.

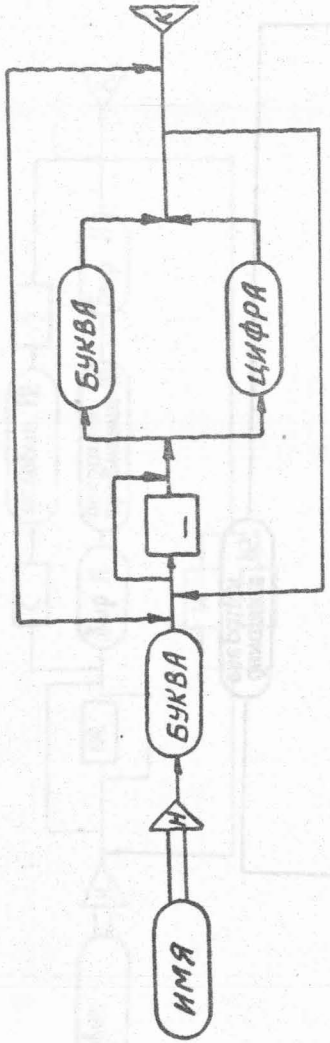


Диаграмма 10

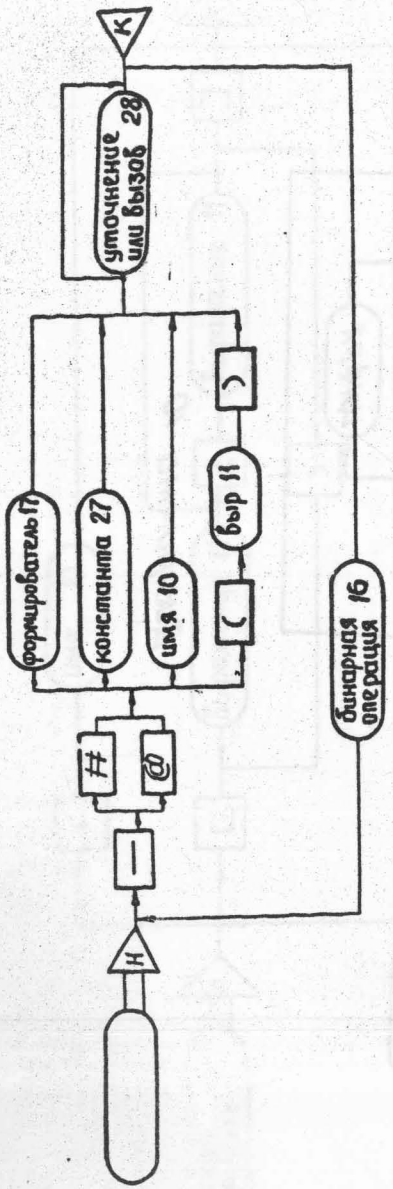


Диаграмма 11

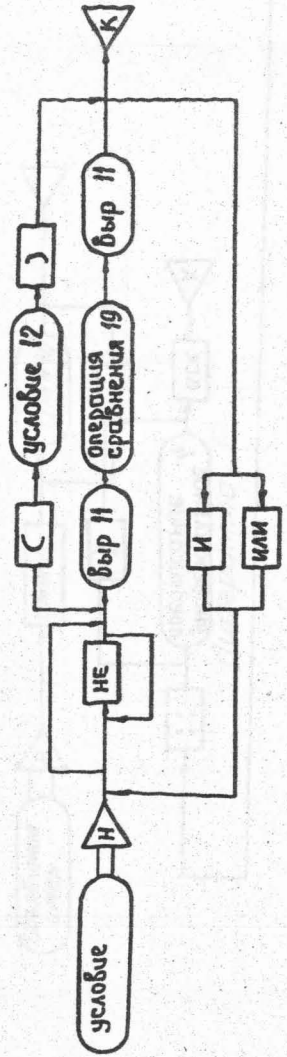


Диаграмма 12

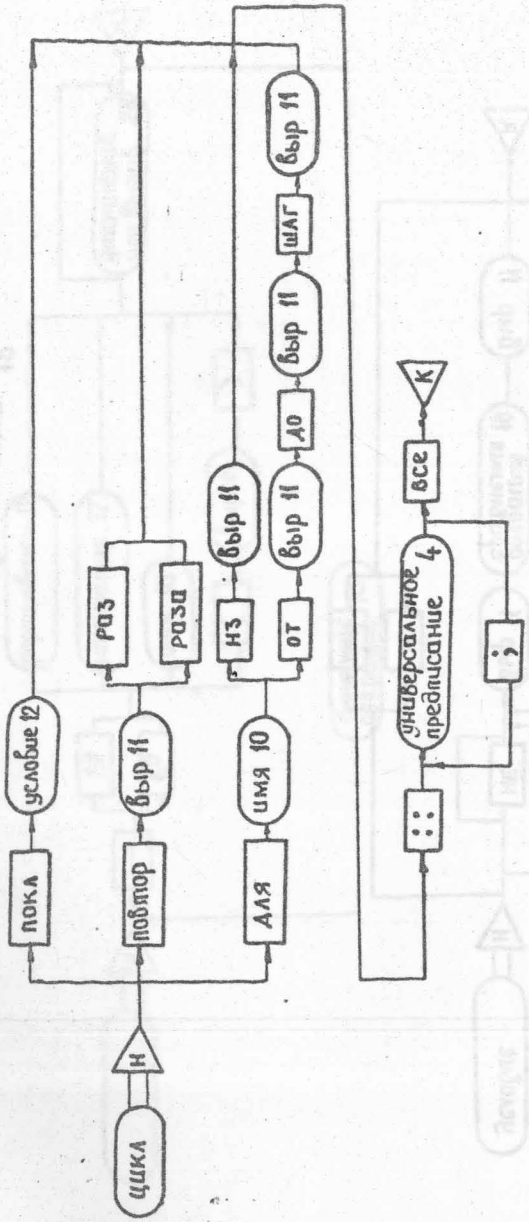


Диаграмма 13

3533847.00042-01 33 01

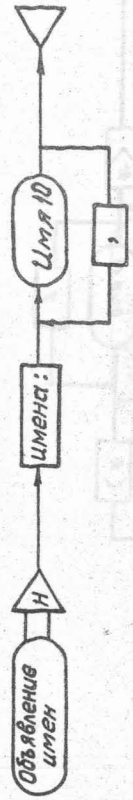


Диаграмма 14

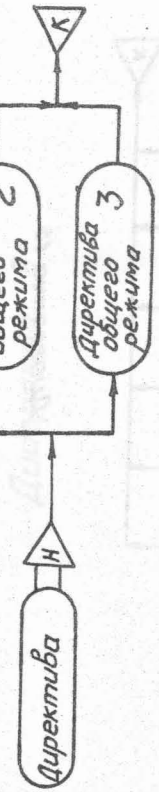


Диаграмма 15

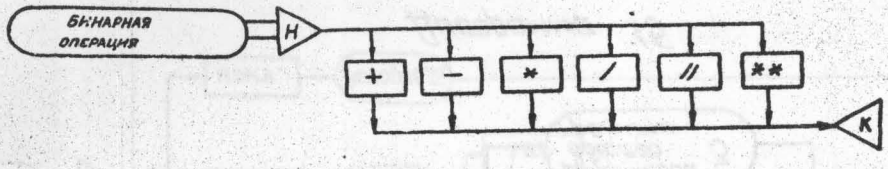


Диаграмма 16

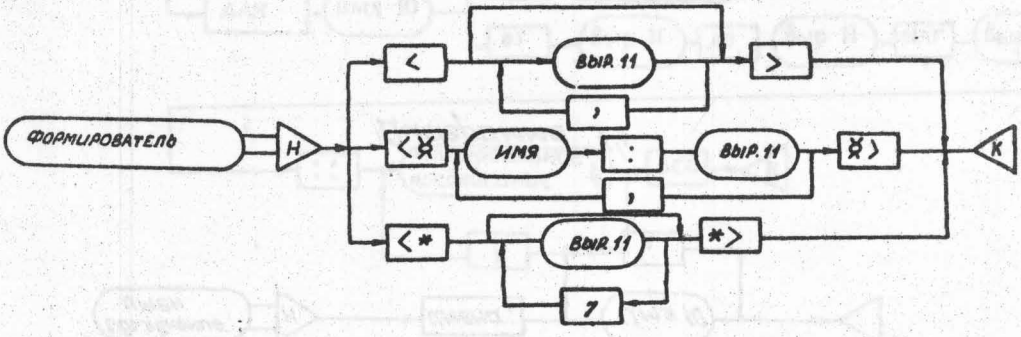


Диаграмма 17

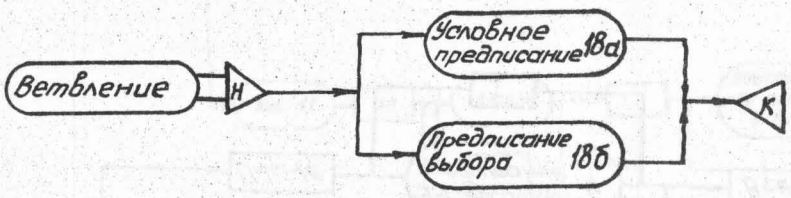


Диаграмма 18

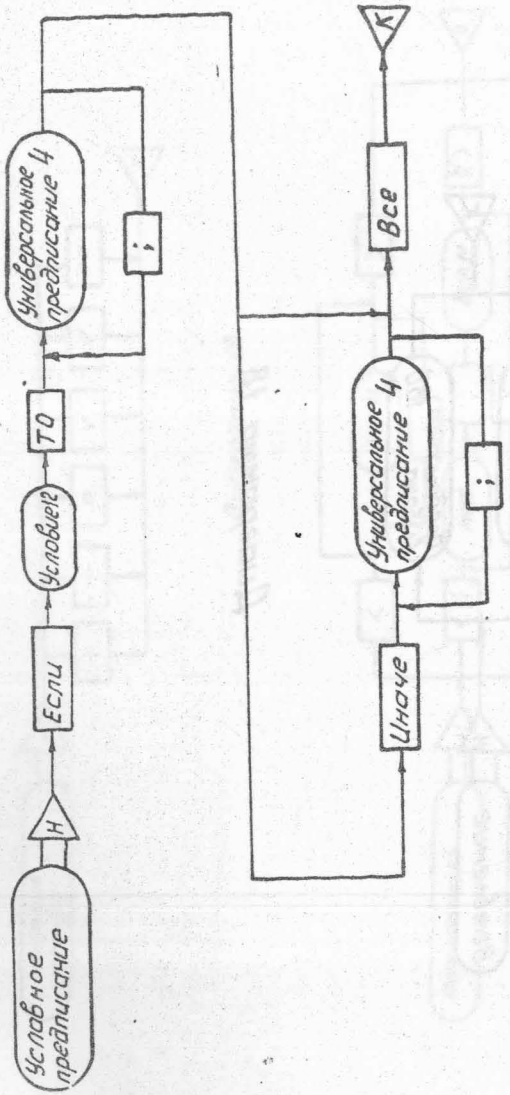


Диаграмма 18а

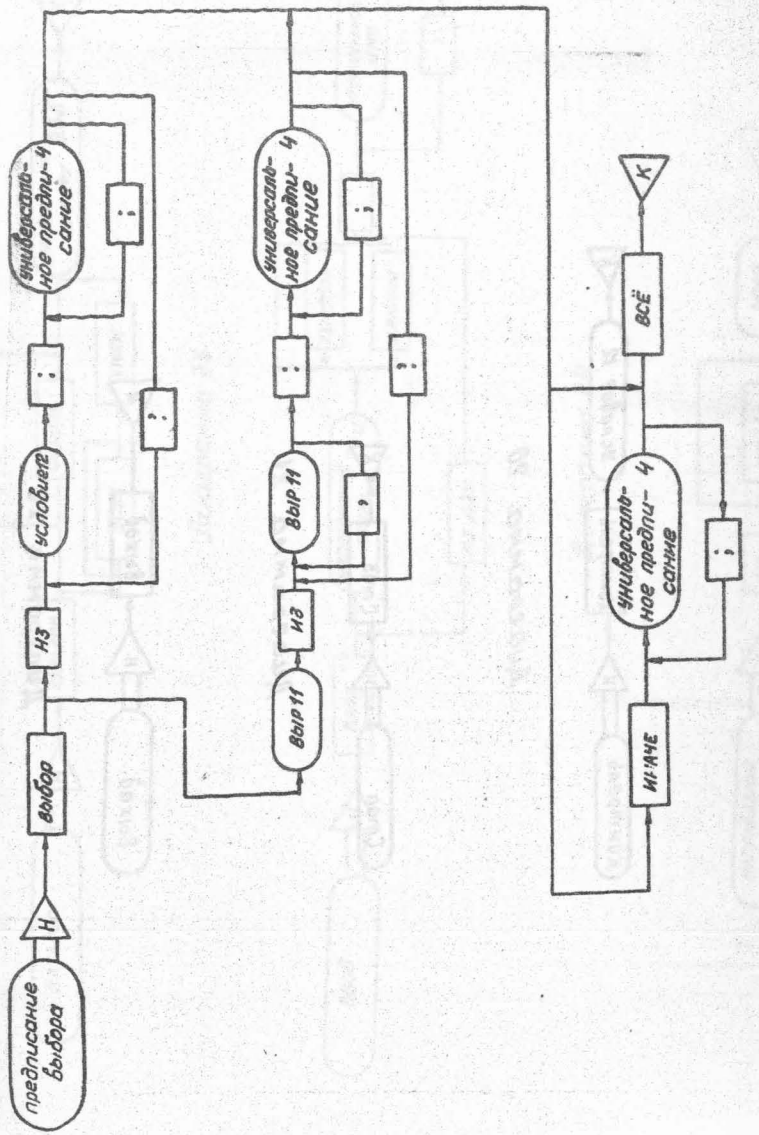


Диаграмма 18б

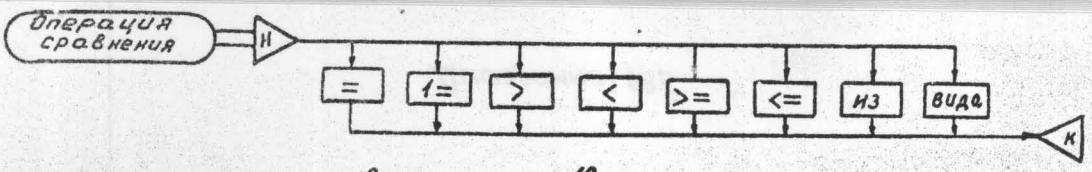


Диаграмма 19

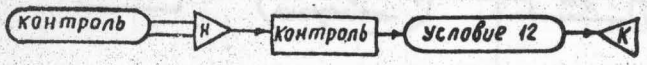


Диаграмма 20

Диаграмма 21

Диаграмма 22

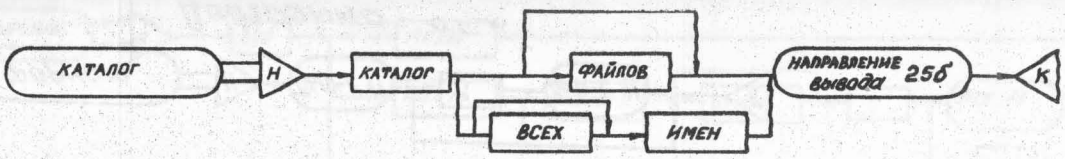


Диаграмма 23

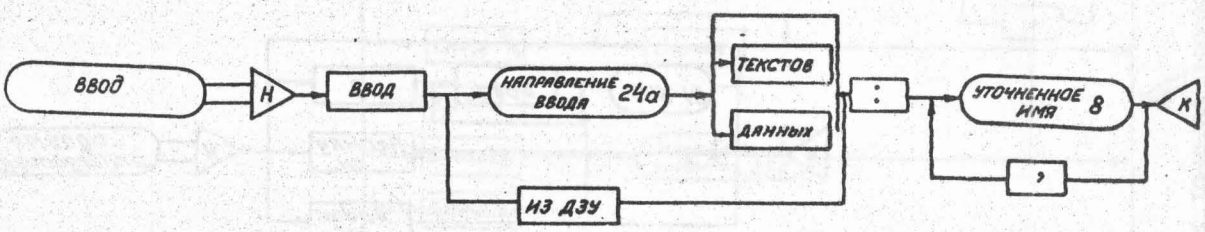


Диаграмма 24

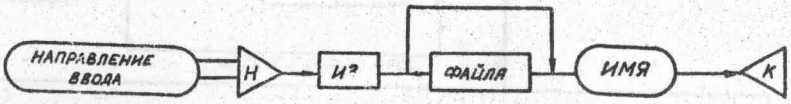


Диаграмма 24а

Диаграмма 25

Диаграмма 25б

Диаграмма 25в

Диаграмма 26

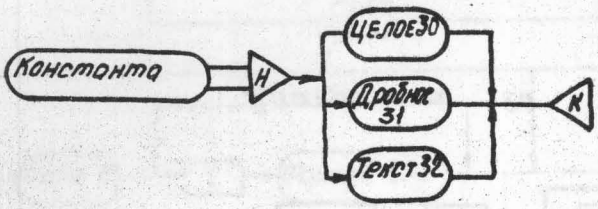


Диаграмма 27

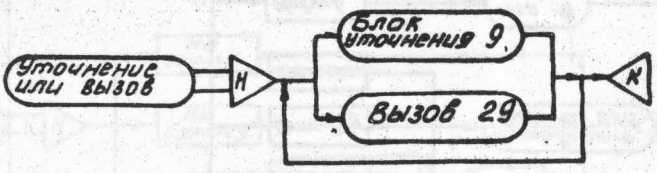


Диаграмма 28

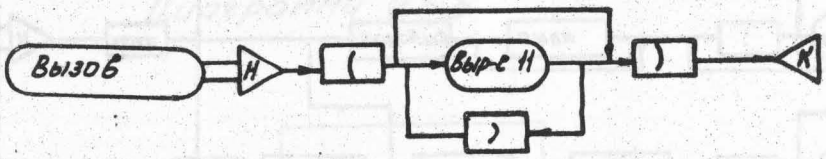


Диаграмма 29

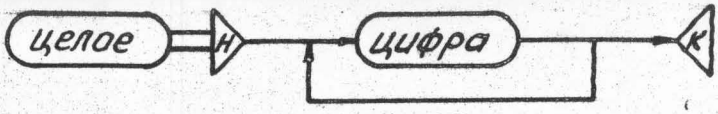


Диаграмма 30

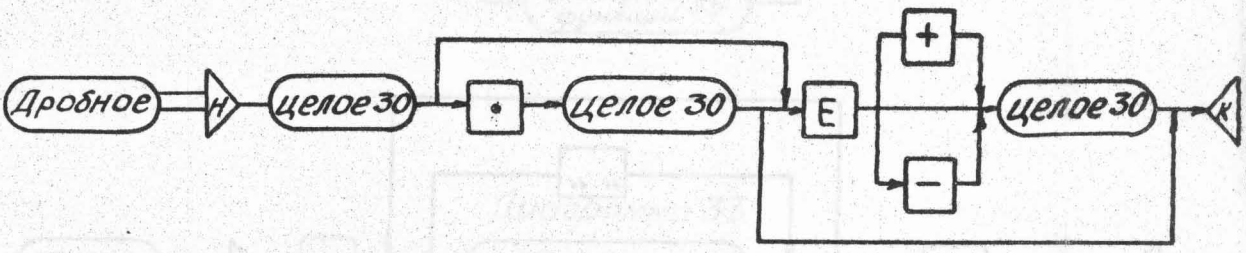


Диаграмма 31

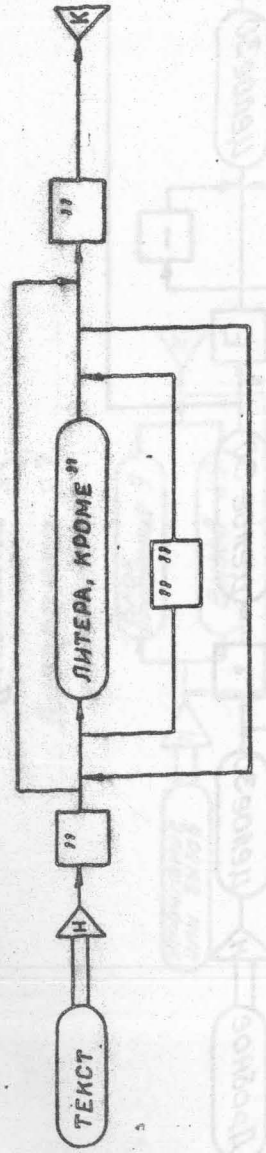


Диаграмма 32

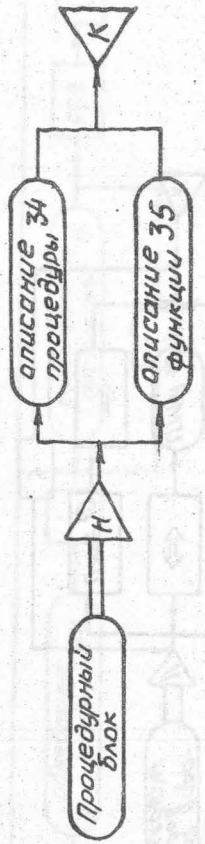


Диаграмма 33

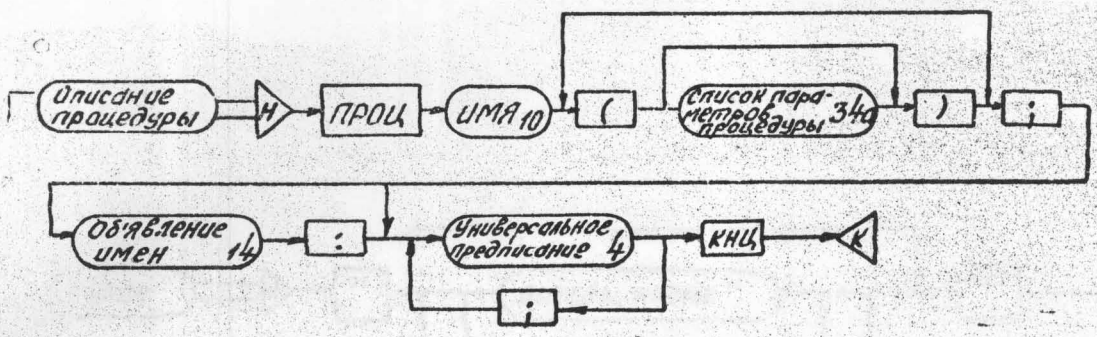


Диаграмма 34.

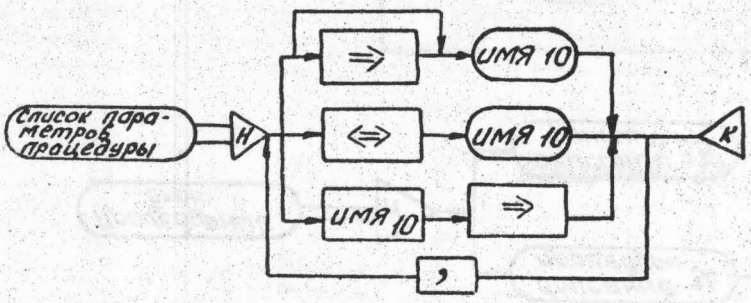


Диаграмма 34а

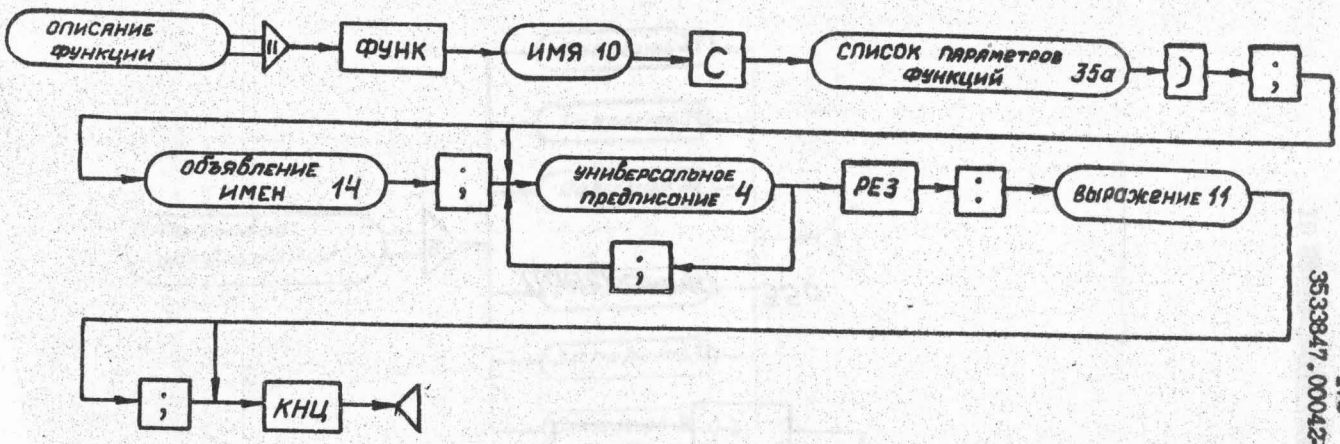


Диаграмма 35

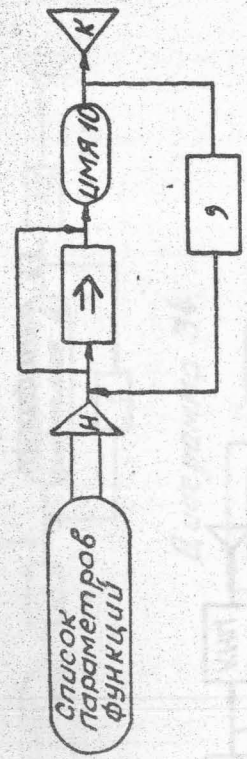


Диаграмма 35а.

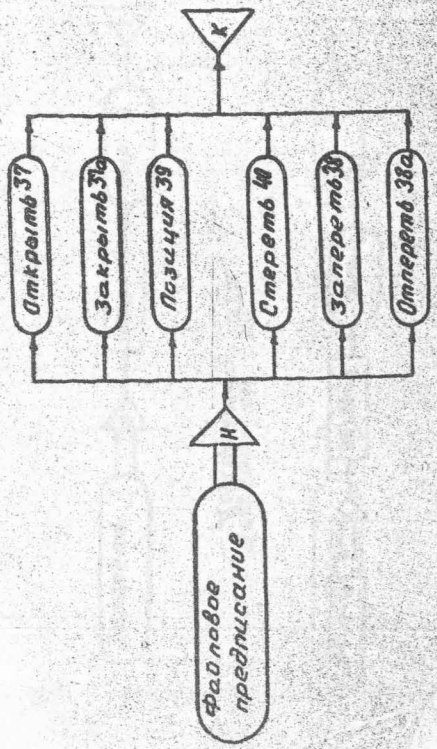


Диаграмма 36

Диаграмма 37

Диаграмма 37а

Диаграмма 38

Диаграмма 38 а

Диаграмма 39

Диаграмма 40

Диаграмма 41

Диаграмма 41а

Диаграмма 42

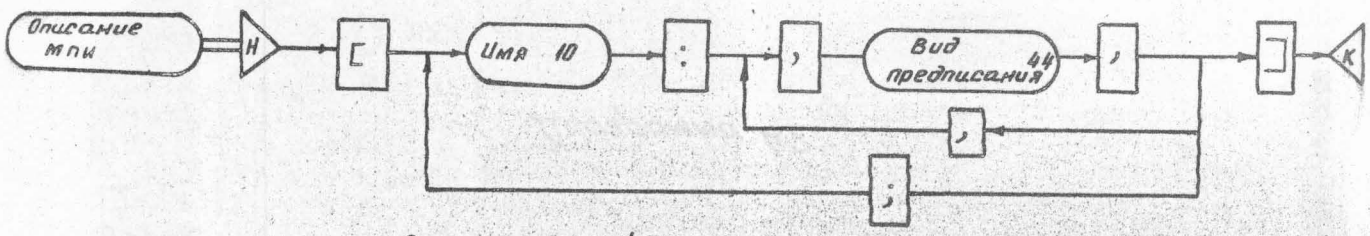


Диаграмма 43

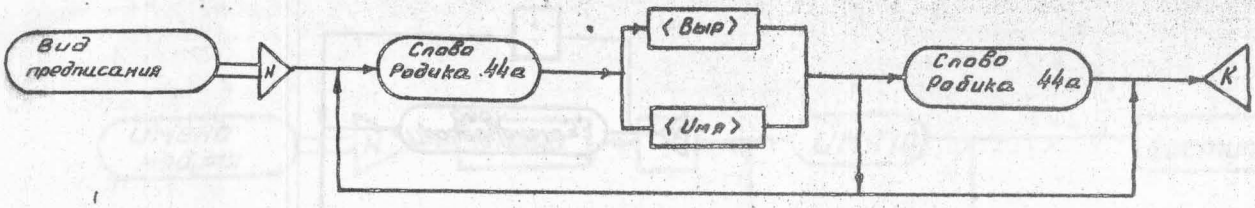


Диаграмма 44

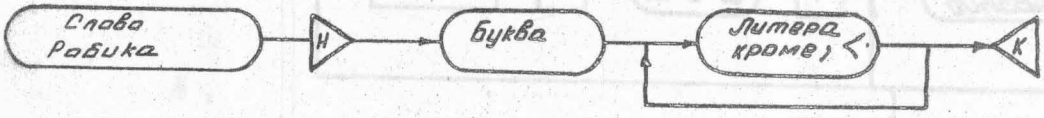


Диаграмма 44а

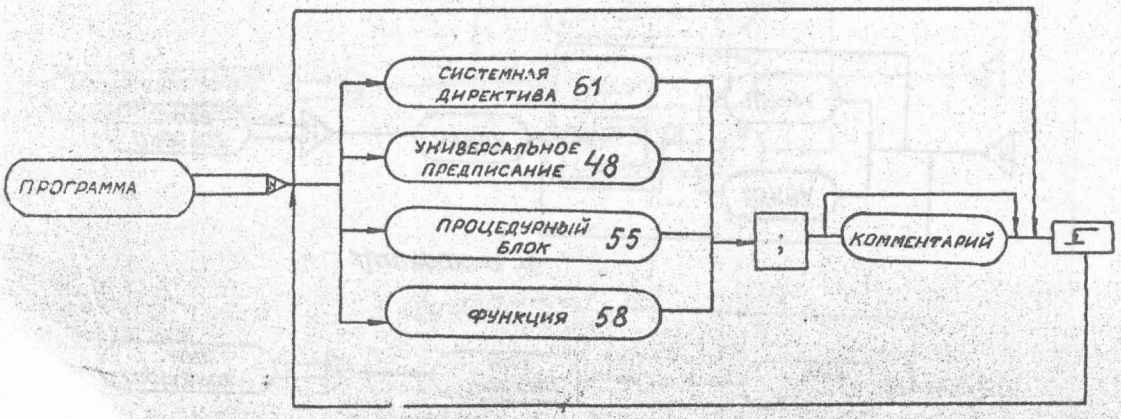


Диаграмма 45

Диаграмма 46

Диаграмма 47

Диаграмма 48

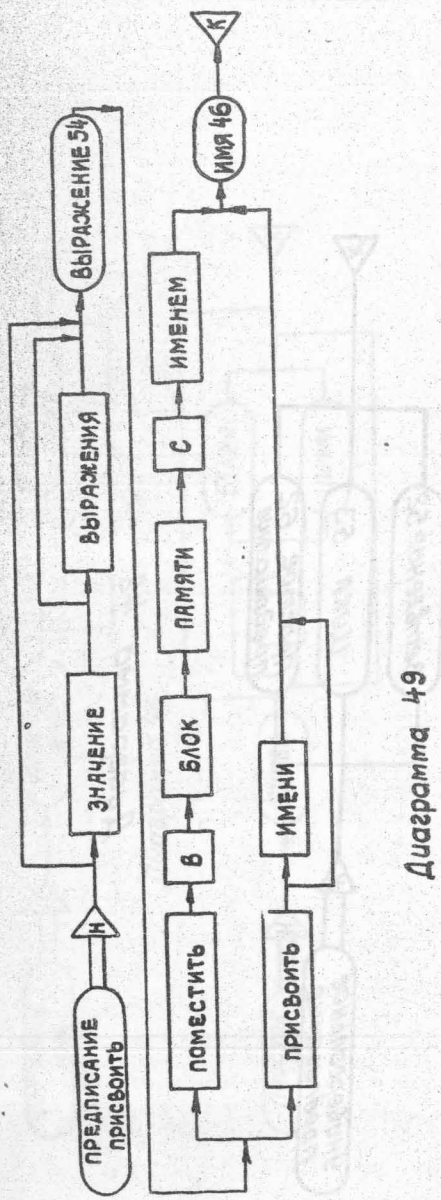


Диаграмма 49

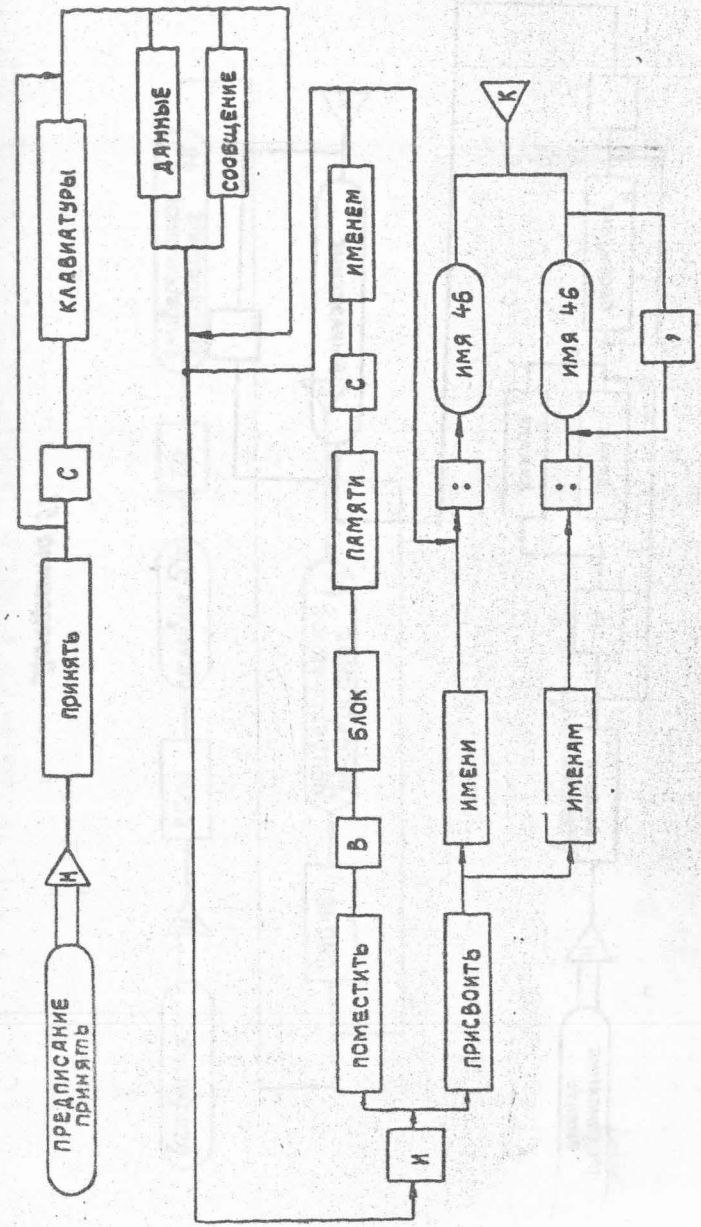


Диаграмма 50

Диаграмма 51

Диаграмма 52

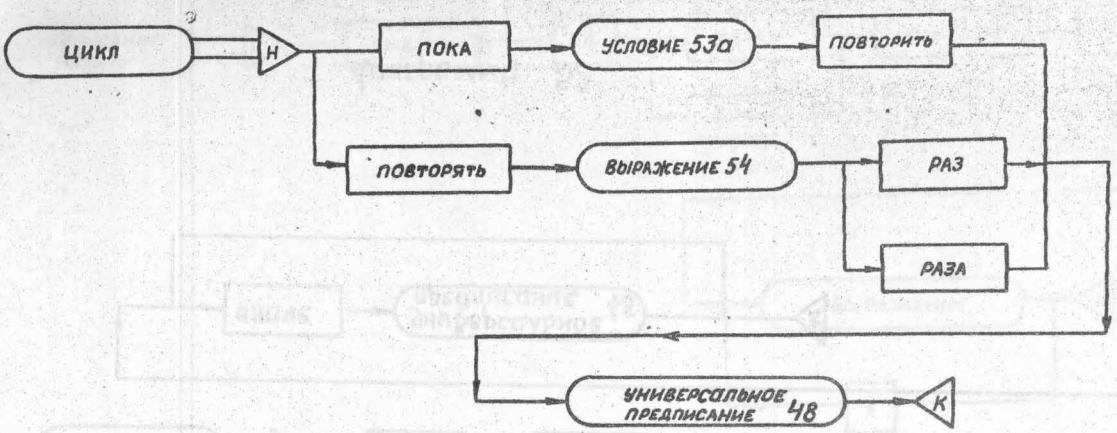


Диаграмма 53

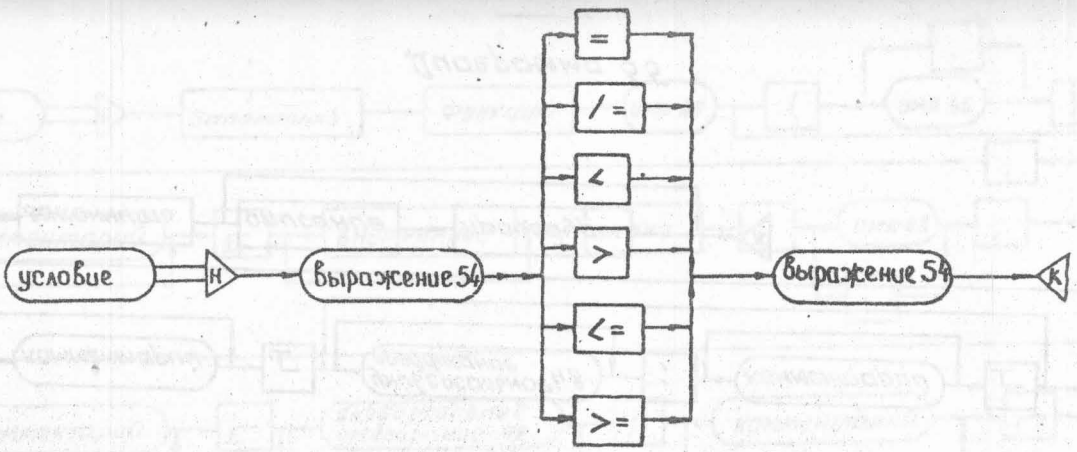


Диаграмма 53а

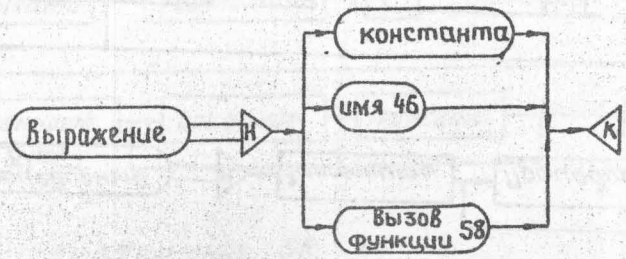


Диаграмма 54

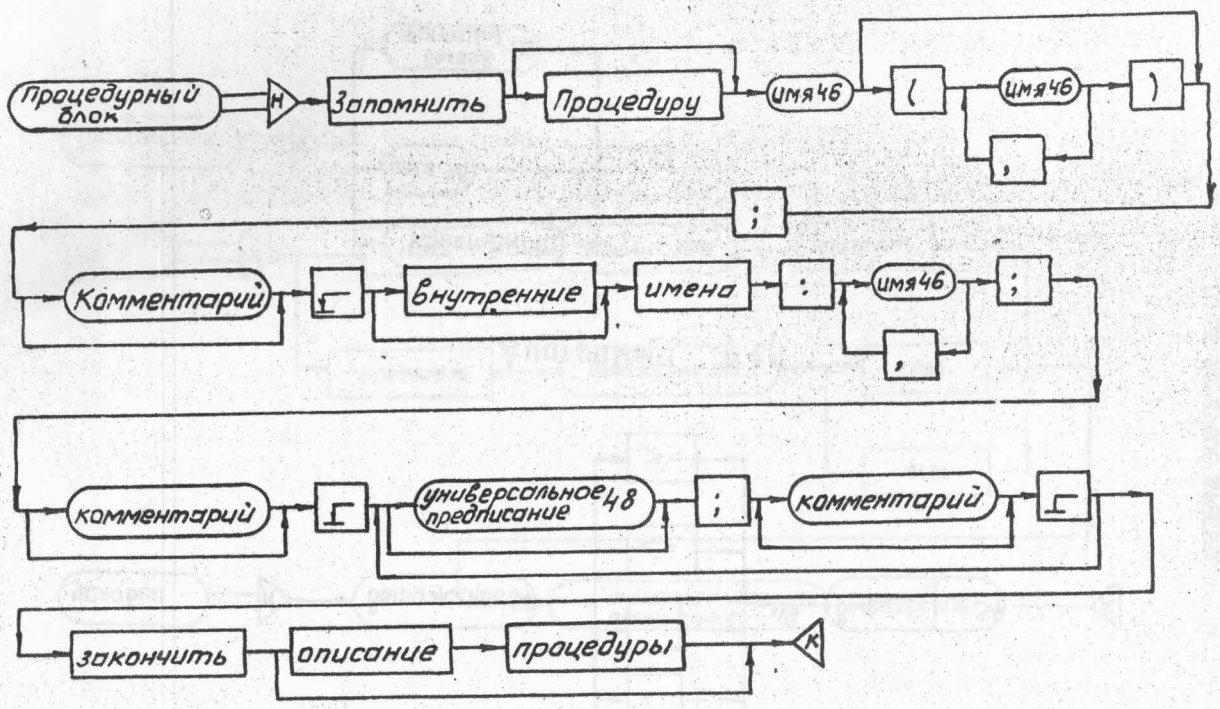


Диаграмма 55

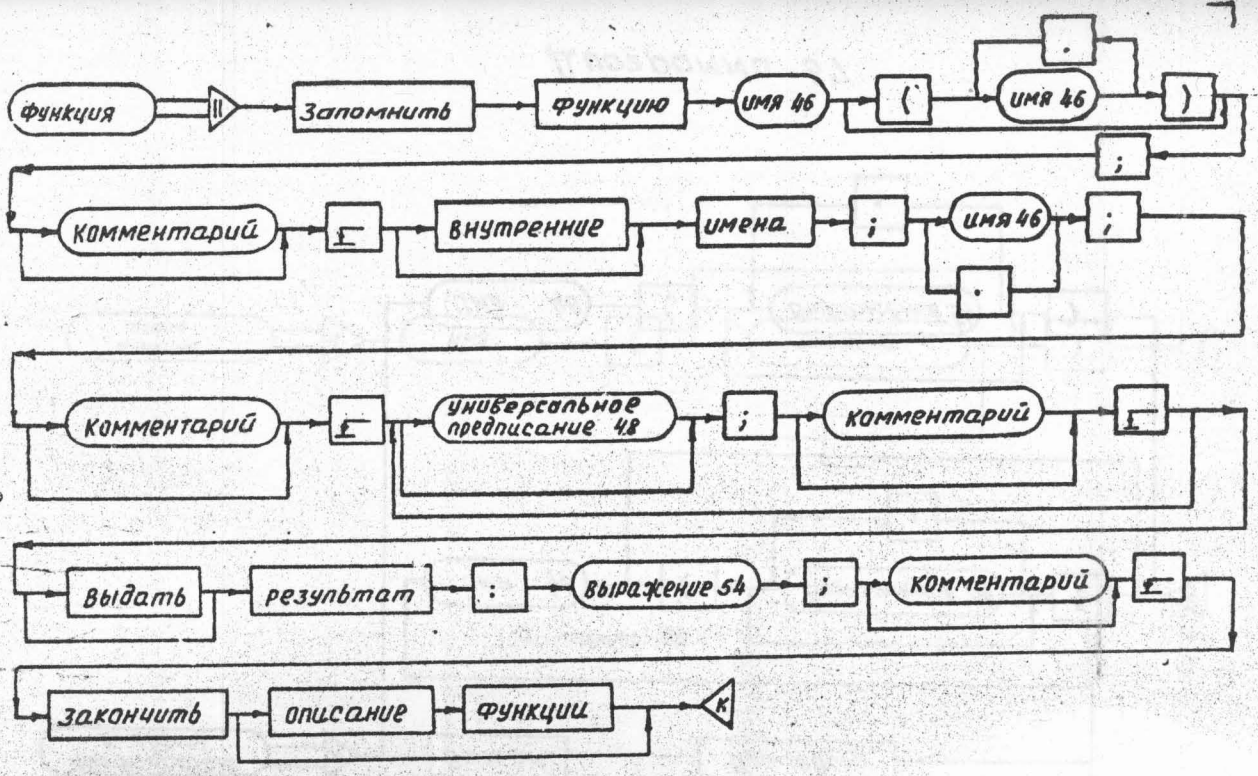


Диаграмма 56

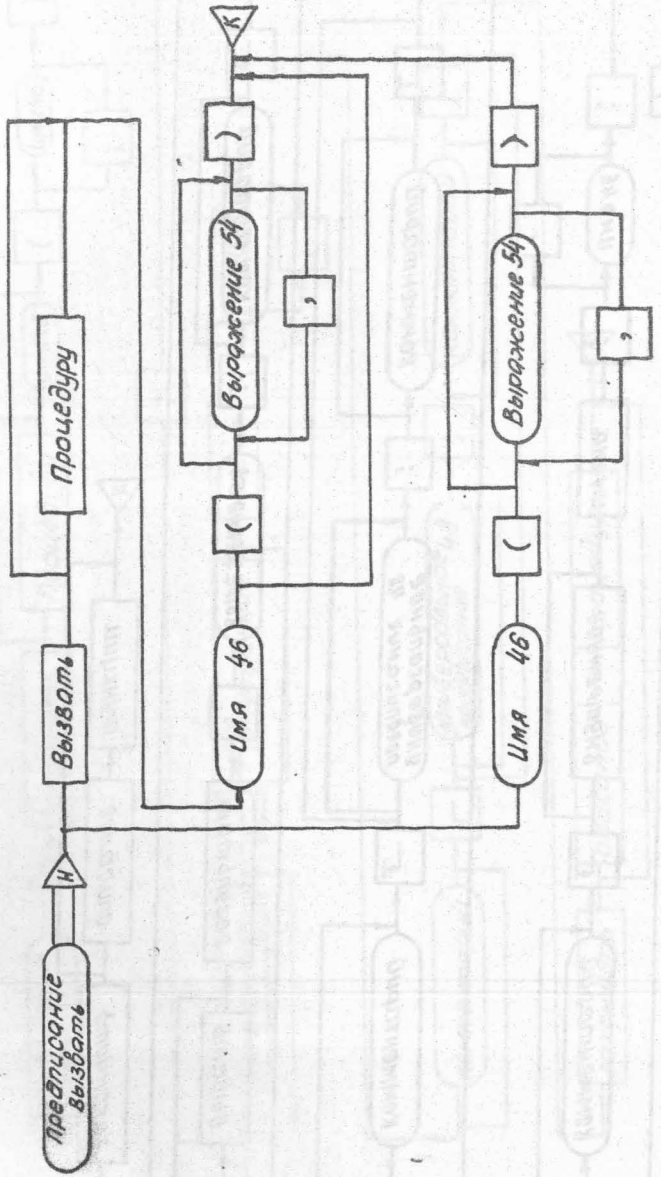


Диаграмма 57

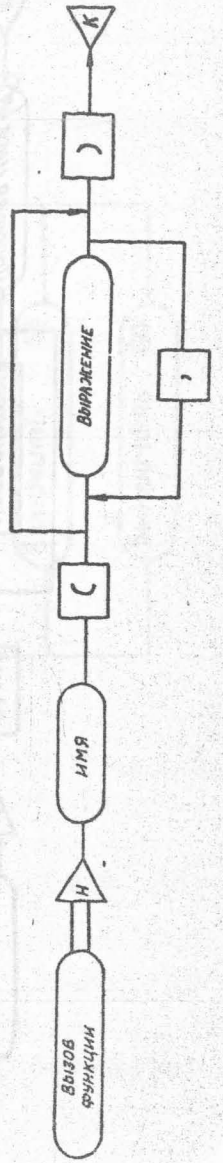


Диаграмма 58

Диаграмма 59

Диаграмма 60

Диаграмма 61

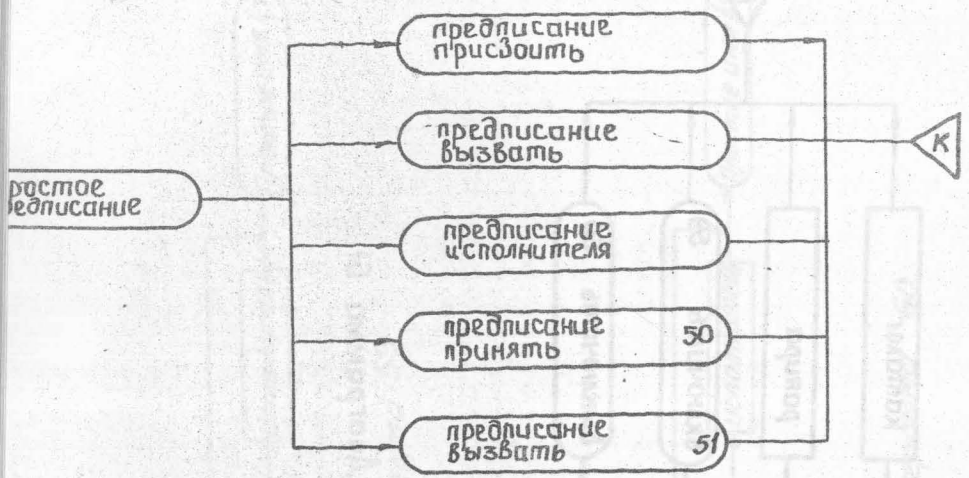


Диаграмма 62.

ПРИЛОЖЕНИЕ 4. ДИАГНОСТИЧЕСКИЕ СООБЩЕНИЯ СИСТЕМЫ "ШКОЛЬНИЦА"

В этом приложении описываются сообщения об ошибках, выдаваемые интерпретатором языков РАПИРА и РОБИК. Все принятые в системе количественные ограничения, нарушение которых приводит к ошибкам, можно найти в Приложении 5.

I. Лексические и синтаксические ошибки

I.I. Причины возникновения

Лексические и синтаксические ошибки – это ошибки в записи программы, т.е. несоблюдение правил построения лексем, выражений и предписаний по синтаксическим диаграммам языка. Ошибки в записи лексем – имен, текстов, чисел, специальных составных символов – называются лексическими. Ошибки при составлении предписаний из отдельных лексем называются синтаксическими.

При попытке ввести такое предписание в машину будет выдано сообщение о синтаксической (лексической) ошибке, и программа не выполнится.

В системе "ШКОЛЬНИЦА" такие ошибки возможны в следующих ситуациях.

I) Неверное предписание в основном диалоговом режиме. Строка с ошибкой дублируется от начала предписания, в котором допущена ошибка, до конца строки; причем неправильная лексема выделяется на экране цветом. Затем выдается сообщение о характере ошибки.

Чтобы все-таки исполнить предписание, в котором допущена ошибка, его необходимо полностью ввести еще раз. Если ошибка допущена в сложном предписании, то требуется повторить ввод всего этого предписания, исправив ошибку. Это можно сделать с помощью описанных в Приложении 2 средств экранного редактирования в основном диалоговом режиме.

2. Неверное предписание в процедурном блоке. При выходе из Редактора и проверке синтаксиса на экран выдается участок редактируемого текста, в котором начало ошибочной лексемы отмечено курсо-

ром. На нижней строке выдается сообщение о характере ошибки. Следует исправить ошибку и повторить выход.

3) Неверная запись данных во время ввода по предписанию ВВОД ДАННЫХ. Если данные вводятся с клавиатуры, то, как и в первом случае, выдается последняя введенная строка с выделенной ошибочной лексемой, а затем – сообщение "ОШИБКА: ПОВТОРИТЕ ВВОД". После этого следует продолжить ввод данных, считая ошибочную лексему переводом строки (см.пример в 4.9 инструкции).

При вводе данных из файла выдается только сообщение "ОШИБКА ВВОДА", и ошибка считается ошибкой исполнения программы (см. 2.2, сообщения П14 и П15).

4) Неверное предписание в процедурном блоке, вводимом из ДЗУ по предписаниям ВВОД ИЗ ДЗУ и ВКЛ МОДУЛЬ. В этой ситуации выдается сообщение о характере ошибки, а сама она рассматривается как ошибка исполнения программы. Исправить ошибку можно средствами Редактора.

При возникновении синтаксической (лексической) ошибки текущий режим работы системы не меняется. Вся диагностическая информация поступает в поток системного вывода.

I.2. Диагностические сообщения

Большинство сообщений имеет вид:

ТРЕБУЕТСЯ список лексем

Такие сообщения выдаются, когда при анализе очередной лексемы оказалось, что по правилам записи предписания она недопустима. Список состоит из тех лексем, которые допустимы в данной позиции. Он, однако, не всегда точно указывает на те изменения, которые необходимо произвести в программе.

Ниже перечислены все возможные диагностики на лексические и синтаксические ошибки и краткие пояснения к ситуациям, в которых они возникают.

Лексические ошибки.

Л1. ТРЕБУЕТСЯ ЦИФРА ПОСЛЕ ТОЧКИ

В записи дробного числа отсутствуют цифры после точки, что недопустимо. Примеры: I0. 25.EI7.

Если необходимо записать число с нулевой дробной частью, следует после точки поставить ноль: I0.0 25.0EI7 или 25EI7

Л2. ТРЕБУЕТСЯ ПОКАЗАТЕЛЬ СТЕПЕНИ

Допущена ошибка в записи порядка дробного числа. Примеры:

IE-A I.IE++7

Л3. НЕ ЗАКРЫТ ТЕКСТ

Отсутствует закрывающая кавычка текста. Если необходим символ перевода строки внутри текста, то его следует вставить программными средствами, например с помощью функции АПФ (CHR).

Л4. "_" В КОНЦЕ ИМЕНИ

Л5. "__" В ИМЕНИ

Такие имена по правилам языка недопустимы.

Л6. НЕПОНЯТНЫЙ СИМВОЛ

Встретился символ, не входящий в алфавит языка (% & ^) управляющие символы и др.) или подчерк стоит не внутри имени, текста или комментария. Непонятными считаются все управляющие символы, вставленные во вводимую строку.

Л7. СЛИШКОМ ДЛИННАЯ ЛЕКСЕМА

Длина лексемы – текста, числа, имени – при вводе с клавиатуры, из файла или при анализе описания процедурного блока больше максимально допустимой.

Л8. НЕДОПУСТИМОЕ ДРОБНОЕ

Вводится слишком большое или слишком маленькое (по модулю) дробное число. Это сообщение будет выдано и в том случае, когда явно задан недопустимый порядок числа: например, хотя число 0.001Е128 возможно, его явно указанный порядок 128 недопустим.

Л9. ТРЕБУЕТСЯ "ж/"

Нет закрывающей скобки комментария в процедурном блоке.

Синтаксические ошибки

С1. ТРЕБУЕТСЯ ";"

Начало предписания может быть воспринято как некоторое другое правильное предписание, а дальнейшая его часть синтаксически неверна. По диаграмме в этой позиции может стоять точка с запятой или некоторая, необязательная часть предписания.

Можно выделить основные типы ошибок, которые сопровождаются этой диагностикой.

а) Нет разделителя между предписаниями, например:

```
A-> B ВЫВОД:B;
```

б) Неверное начало предписания (рассматривается как пустое предписание), например:

```
A-> B; > A+B) ж2-> C;
```

в) Начало выражения, которым оканчивается предписание, представляет собой другое правильное выражение:

```
ВЫБ : (A+B) ! (C+E);
КОНТРОЛЬ A>(K+M) ;
```

Подчеркнута та часть предписания, которая представляет собой синтаксически правильную конструкцию.

г) Пропущена запятая в списке элементов (в предписаниях ВКЛ, ВЫКЛ, ВВОД, ВЫВОД, ИМЕНА), например:


```
ВКЛ ПРОКРУТКУ: A,B C,Д;
ВЫВОД: A+7 Bжж2(7-K);
```

С2. ТРЕБУЕТСЯ "ВСЕ"

Ситуация, описанная в С1, возникла в сложном предписании: т.е. по диаграмме в текущей позиции допустимы слово "ВСЕ" или предписание.

Примеры:

```
ЕСЛИ A> B ТО I-> A ВСЕ;
```

После "I-> A" возможны еще предписания через ";", ключевое слово "ИНАЧЕ" с альтернативной частью условного предписания.

Еще больше возможных продолжений можно найти в следующей ситуации (см.диаграмму I8):

```
ВЫБОР A ИЗ I: I-> B B
```

С3. ТРЕБУЕТСЯ "КНЦ"

С4. ТРЕБУЕТСЯ "РЕЗ"

Ситуация, описанная в С1, возникла в описании процедурного блока.

С5. ТРЕБУЕТСЯ ИМЯ

С6. ТРЕБУЕТСЯ ИМЯ ФАЙЛА

По диаграмме предписания в данной позиции должно стоять имя.

С7. ТРЕБУЕТСЯ ВЫРАЖЕНИЕ ИЛИ"ИЗ"

Ошибка в заголовке предписания ВЫБОР, при которой неизвестно, какая из двух форм этого предписания имелась в виду, например:

```
ВЫБОР:
```

С8. ТРЕБУЕТСЯ ОПЕРАНД

Ошибка в записи операнда одной из операций в выражении.

С9. ТРЕБУЕТСЯ ЗНАК ОПЕРАЦИИ ИЛИ ПРИСВАИВАНИЕ

В начале предписания стоит выражение, после которого нет знака присваивания или скобок вызова. Других предписаний, начинающихся с выражения, в языке нет.

С10. ТРЕБУЕТСЯ ПРОЦ.БЛОК

Нет заголовка описания процедуры или функции "ПРОЦ" или "ФУНК". Эта диагностика возможна в двух случаях:

а) в анализируемом как процедурный блок тексте первая лексема - не "ПРОЦ" и не "ФУНК";

б) после конца описания очередной процедуры или функции есть еще символьная информация, не являющаяся описанием процедурного блока или комментарием.

Таким образом, в транслируемом файле могут располагаться только процедуры и функции, возможно, разделенные комментариями.

С11. ТРЕБУЕТСЯ УСТРОЙСТВО

В предписаниях переключения вывода ВКЛ/ВЫКЛ ОТЛАДКУ/ВЫВОД/ ПРОТОКОЛ не указано направление вывода - "НА БУМАГУ", "НА ЭКРАН" или "В ФАЙЛ имя".

С12. НЕВЕРНЫЙ ЗАГОЛОВОК ЦИКЛА

Нет на месте ключевых слов "ОТ" или "ИЗ" в цикле ДЛЯ, например:

```
ДЛЯ I=5 ДО 7 : :
```

С13. НЕВЕРНЫЙ РЕЖИМ

Указан несуществующий режим в предписаниях ВКЛ/ВЫКЛ, например:

```
ВКЛ СВЕТ;
```


С14. СЛИШКОМ ДЛИННЫЙ СПИСОК

Слишком много элементов кортежа, множества, параметров процедуры на вводе (см.Приложение 5).

С15. ТАКОЕ ИМЯ УЖЕ БЫЛО

Повторение имени в записи, локальных имен и имен параметров процедуры, совпадение имени параметра с локальным именем.

С16. Прочие сообщения:

ТРЕБУЕТСЯ УСЛОВИЕ

ТРЕБУЕТСЯ ВЫРАЖЕНИЕ

ТРЕБУЕТСЯ "] " ИЛИ ","

ТРЕБУЕТСЯ ":" ИЛИ ","

ТРЕБУЕТСЯ "БУМАГУ" ИЛИ "ЭКРАН"

ТРЕБУЕТСЯ " > " ИЛИ ","

ТРЕБУЕТСЯ "ж " ИЛИ ","

ТРЕБУЕТСЯ " Д " ИЛИ ","

ТРЕБУЕТСЯ ")" ИЛИ ","

ТРЕБУЕТСЯ ")"

ТРЕБУЕТСЯ "="

ТРЕБУЕТСЯ ключевое слово

ТРЕБУЕТСЯ ": :"

Смысл этих диагностик легко восстанавливается по контексту и диаграмме предписания.

2. Ошибки исполнения программы

2.1. Причины возникновения

Ошибкой исполнения программы называется ситуация, которая повлекла за собой невозможность правильного исполнения указанных в программе предписаний. Такие ошибки являются, как правило,

следствием незерной обработки данных в программе, что приводит к попыткам выполнения операций над объектами непредусмотренных для данной операции видов, к нарушению области допустимых значений и т.д.

Ряд ошибок связан с исчерпанием ресурсов ЭВМ (оперативной и внешней памяти, программных стеков), сбоями ЭВМ и, особенно, внешних устройств.

Возникновение ошибочной ситуации вызывает прерывание исполнения программы и выдачу диагностического сообщения. Так как ошибка может привести к порче программной среды (засорению памяти, потере отдельных объектов и т.п.), системе приходится выполнять определенные действия по нейтрализации возможных последствий ошибки.

Если ошибка возникла при исполнении предписания, введенного в основном диалоговом режиме, то выдается только сообщение о характере ошибки. При ошибке в процедурном блоке, кроме того, выдается номер строки, в которой произошла ошибка, и устанавливается режим "ОШИБКА". Подробнее об этом режиме см.в 7.2 инструкции.

Ниже приводится список диагностик об ошибках исполнения, контролируемых системой, перечисляются системные действия по нейтрализации их последствий, и даются рекомендации по устранению ошибок.

2.2. Ошибки программиста

П1. 0 ** 0

П2. X / 0

П3. НЕДОПУСТИМОЕ ДРОБНОЕ

П4. СЛИШКОМ БОЛЬШОЕ ЦЕЛОЕ

Ошибки, связанные с нарушением области допустимых значений целых и дробных чисел.

П5. НЕДОПУСТИМОЕ ЗНАЧЕНИЕ i ПАРАМЕТРА имя

Нарушение области допустимых значений при задании параметров стандартных функций (координат точек на экране, кодов символов, длительности и т.п.).

П6. НЕДОПУСТИМЫЕ ОПЕРАНДЫ операция

Сделана попытка выполнить указанную операцию над объектами непредусмотренных для нее видов, например, сложить число с текстом. Здесь операция - это знак операции, имя стандартной процедуры (функции), слова "ФОРМАТНОГО ВЫВОДА", "ВЫРЕЗКИ", "ЦИКЛА"

П7. НЕВЕРНАЯ ДЛИНА ВЫРЕЗКИ

Присваивание вырезке (из текста или кортежа) текста или кортежа другой длины, например:

```
I,2,3 -> A 5:6 ;
```

Слева стоит кортеж из 3 элементов, справа - из двух.

П8. ВЫРЕЗКА НЕВОЗМОЖНА

Операция вырезки (выборки) выполняется не над текстом или кортежем; цикл ДЛЯ-ИЗ использован не для текста, кортежа или множества.

П9. НЕВЕРНЫЕ ИНДЕКСЫ

Неверно заданы индексы в операциях вырезки или выборки: в структуре нет элемента с указанным индексом или начальный индекс больше конечного (в случае вырезки).

П10. имя: ТЕКСТ - НЕ ЛИТЕРА

Параметр указанной стандартной функции, а также первый операнд операции "ИЗ" должны быть односимвольными текстами (литерами).

II1. В ЗАПИСИ НЕТ ПОЛЯ имя

В записи отсутствует поле с указанным именем. Ошибка возникает при попытке доступа к этому имени:

¤A:1,B:2¤ .E

II2. ВЫЗЫВАЕТСЯ НЕ ПРОЦЕДУРА

II3. ВЫЗЫВАЕТСЯ НЕ ФУНКЦИЯ

Вызов процедуры или функции используется над объектом другого вида.

II4. ВЫХОД НЕ ИЗ ПРОЦ/ФУНК

Предписание ВЫХОД (не директива!) выполнено в основном диалоговом режиме.

II5. ВЫ НЕ В РЕЖИМЕ ПРИОСТАНОВА

Исполнение директив ПУСК, ШАГ, ВЫХОД не из режима приостанова, а также исполнение директивы ВЫХОД не из режима останова по ошибке. Текущий режим указывается в информационной строке экрана.

II6. НЕВЕРНОЕ ЧИСЛО ПАРАМЕТРОВ

Число фактических параметров процедуры (функции) не соответствует числу ее формальных параметров.

II7. СРАБОТАЛ КОНТРОЛЬ

Не выполнено условие в предписании КОНТРОЛЬ.

II8. ОШИБКА ВВОДА

Отказ от ввода по предписанию ВВОД: первый символ вводимой строки (или после разделителя при вводе данных) - это символ останова (клавиша "F1"). Это же сообщение выдается при любой ошибке во время ввода из файла.

II9. ОШИБКА: ПОВТОРИТЕ ВВОД

Синтаксическая или лексическая ошибка при вводе с клавиатуры по предписанию ВВОД. Выдается последняя введенная строка, в которой ошибочная лексема выделена, а затем - данное сообщение. Происходит повторный запрос, на который надо ввести невосприятые данные,обращая внимание на соблюдение синтаксических правил их записи; ошибочная лексема при этом считается переводом строки. При аналогичной ошибке во время ввода данных из файла выдается только сообщение "ОШИБКА ВВОДА" (II8) без повторного запроса.

II20. СЛИШКОМ МНОГО КОПИЙ

Слишком много копий процедуры, функции, файла. Копии объектов создаются при выполнении операций присваивания и формирования структур.

II2I. СЛИШКОМ СЛОЖНАЯ СТРУКТУРА

Слишком много элементов или слишком большая вложенность в структуре при вводе данных.

II22. ИМЯ ЗАЩИЩЕНО

II23. ПРИСВАИВАНИЕ имя ЗАПРЕЩЕНО

Присваивание нового значения имени, имеющему полную или абсолютную защиту. Такими именами являются:

имя процедуры (функции) во время ее вызова (абсолютная защита);

имя процедуры (функции) в режиме ее приостанова (абсолютная),

параметры циклов ДЛЯ (абсолютная),

имя, участвующее в операции во время вычисления последующих операндов (абсолютная),

имя, защищенное по предписанию ВКЛ ЗАЩИТУ (полная),

имя, имеющее частичную защиту, если на запрос о разрешении присваивания ему был дан отрицательный ответ.

См.также 3.5 инструкции.

П24. имя СИСТ.ЗАЩИЩЕНО

Попытка изменить защиту или признак контроля абсолютно защищенного имени.

П25. НЕВЕРНЫЙ ТИП ПЕРЕДАЧИ ПАРАМЕТРА

В качестве возвратного или выходного параметра в процедуру передается значение, а не имя (вырезка,выборка,доступ к полю записи) - оно неспособно принять возвращаемое процедурой значение.

П26. ИМЯ ФАЙЛА - НЕ ТЕКСТ

В предписаниях ОТКРЫТЬ-КАК, СТЕРЕТЬ,ОТПЕРЕТЬ,ЗАПЕРЕТЬ в поле имени файла указано не текстовое значение.

П27. ФАЙЛ ОТКРЫТ

Попытка уничтожения, смены защиты, повторного открытия открытого файла. Перед выполнением предписаний ОТКРЫТЬ,СТЕРЕТЬ, ОТПЕРЕТЬ,ЗАПЕРЕТЬ следует закрыть файл, если он открыт.

П28. НЕВЕРНЫЙ ТИП ФАЙЛА

Попытка доступа к нетекстовому файлу (текстовые файлы обозначаются литерой "Т" в каталоге файлов).

П29. СЛИШКОМ МНОГО ОТКРЫТЫХ ФАЙЛОВ

В данный момент открыто максимально допустимое число файлов, сделана попытка открыть еще один. Сообщение выдается при выполнении предписания ОТКРЫТЬ. Файл в этом случае не открывается.

П30. ФАЙЛ НЕ ОТКРЫТ

Попытка обработки закрытого файла или нефайлового объекта операциями и предписаниями файловой системы. К ним относятся:

переключение ввода, вывода, отладки, протокола на файл: ВКЛ/ВЫКЛ,

установка позиции указателя файла: ПОЗИЦИЯ,

функции ЧТФ и КФ,

определение позиции указателя:

закрытие доступа к файлу.

См.также 8.4 инструкции.

Перед выполнением требуемых действий следует открыть файл.

П31. ФАЙЛ имя ЗАКРЫТ НА ЗАПИСЬ

П32. ФАЙЛ ЗАПЕРТ

Запись в защищенный внешний файл или его уничтожение. Первое сообщение выдается, если такой файл открыт и сделана попытка вывода в него. Второе - при записи в него из Редактора или при попытке уничтожения.

П33. ФАЙЛ НЕ НАЙДЕН

Попытка уничтожения, изменения защиты несуществующего внешнего файла. В Редакторе, кроме того, это сообщение выдается при считывании несуществующего файла. Редактируемый текст при этом не портится.

П34. КОНЕЦ ДАННЫХ В имя

В режиме считывания данных из файла или перемещении его указателя считан символ "конец файла". Ошибка возможна при исполнении предписаний ПОЗИЦИЯ, ВВОД ИЗ ФАЙЛА, функции ЧТФ.

П35. НЕВЕРНАЯ ПОЗИЦИЯ УКАЗАТЕЛЯ имя

Обращение к предписанию ПОЗИЦИЯ с отрицательным, нулевым или заведомо превосходящим объем ДЗУ номером позиции указателя файла. Указатель при этом не перемещается.

П36. СЛИШКОМ БОЛЬШОЙ ФАЙЛ

Считываемый файл не помещается в рабочую память Редактора. Считывается столько информации, сколько позволяет память, а оставшаяся ее часть обрезается. Следует не допускать такой ситуации и хранить текст в нескольких файлах допустимого для обработки из Редактора размера.

П37. ПЕРЕСТАВЛЕН ДИСК

Переставлен диск при наличии в системе незакрытых файлов, что недопустимо (см. 8.3 инструкции). Следует установить тот диск, на котором были открыты файлы, чтобы дать возможность системе вести с ними нормальный обмен. Если необходимо все же сменить диск, следует сначала закрыть все открытые файлы и лишь потом вынуть диск из дисковода.

Для нейтрализации возможных последствий ошибки и для сохранения накопленной информации отключаются все потоки ввода и вывода, настроенные на файл, при обращении к которому произошла эта ошибка. Указатель файла не перемещается.

П38. ДИСК ЗАКРЫТ НА ЗАПИСЬ

Попытка записи на диск без прорези или с заклеенной прорезью. Если диск не защищен специально, то, скорее всего, он не проинициализирован, т.е. на нем отсутствует необходимая для хранения данных разметка. В этом случае следует сделать прорезь и разметить диск (см. 5.1 инструкции).

Примечание. Ошибки, описанные в П37 и П38, являются специфичными для работы с дисководами ЕС-5089 в версии А1.1. В версиях системы на других ЭВМ или использующих другие типы ДЗУ возможны другие правила обмена и другие ошибочные ситуации.

П39. СЛИШКОМ МНОГО МОДУЛЕЙ

Предпринята попытка включить 17-й модуль.

Примечание: здесь и ниже вместо модуля может упоминаться исполнитель, если ошибка произошла в Робике.

П40. МОДУЛЬ УЖЕ ВКЛЮЧЕН

Попытка повторного включения работающего модуля.

П41. ТАКОГО МОДУЛЯ НЕТ

Сделана попытка обратиться к несуществующему модулю, т.е. выключить его,
обратиться к имени этого модуля.

П42. МОДУЛЬ В РАБОТЕ

Сделана попытка выключить модуль, одна из процедур которого находится в режиме приостанова.

П43. ТАКОГО ИМЕНИ НЕТ В МОДУЛЕ

Обращение к имени модуля, которое не было описано в нем, как доступное.

П44. ИМЯ НЕ ОПИСАНО

Одна из процедур, находящихся в файле с описанием модуля, не упомянута в списке имен модуля.

П45. ВЫХОД ЗА ГРАНИЦЫ ЭКРАНА

Неверно указаны координаты точки экрана.

П46. ПЕРЕПОЛНЕН ПОТОК ВЫВОДА

Выводимая информация одного из потоков вывода (программного, отладочного или системного) поступает на слишком большое количество устройств вывода; ошибка возникает при исполнении предписания ВКЛ, подключение вывода к указанному устройству в этом случае не производится.

2.3. Ошибки программной среды (системные)

Описываемые ниже ошибки связаны с исчерпанием ресурсов системы, обеспечивающих ее работу и работу пользователя. Набор этих ошибок зависит только от реализации системы на конкретной ЭВМ.

I. ОШИБКА ОБМЕНА С ДЗУ

Является следствием неудачной попытки считывания или записи информации. Наиболее распространены следующие ситуации возникновения:

а) не закрыт дисковод;

б) не размечен диск; следует воспользоваться программой начальной разметки (5.I инструкции);

в) нет диска в дисководе;

г) испорчена часть информации на диске; попробуйте восстановить ее системными средствами или хотя бы защитить испорченную часть диска от повторных обращений. Предвидя такие ошибки, следует всегда иметь копии всех рабочих файлов;

д) неисправен дисковод.

Если ошибка обмена произошла при вводе из файла, выводе в него или смещении указателя, все потоки ввода и вывода, настроенные на этот файл, отключаются.

2. ИСЧЕРПАНА ПАМЯТЬ

Не хватает места в ОЗУ для хранения программы или данных пользователя, а также для работы самой системы. Рекомендации:

перевызвать систему;

уничтожить ненужные в данный момент процедуры и другие значения имен присваиванием их пустых значений;

оптимизировать программу за счет констант, числа предписаний и т.п.;

сделать некоторые блоки программы нерезидентными, т.е. загружать их из ДЗУ по мере надобности и уничтожать после использования.

Если ошибка все равно возникает, значит задача в данной версии системы при выбранном алгоритме и организации данных не может быть решена.

3. ПЕРЕПОЛНЕНИЕ СТЕКА

Слишком большая вложенность структур, длинные списки, большая глубина рекурсии. Следует иметь в виду количественные характеристики системы. В результате возникновения этой ошибки во время исполнения программы возможно частичное засорение памяти.

4. СТЕК ПУСТ

Системная ошибка, возникающая обычно при выходе из режимов приостанова и останова по ошибке, процедур и функций, при обработке структур.

Если ошибка повторяется после перевызова системы при повторном запуске, следует попытаться изменить программу.

5. СЛИШКОМ МНОГО ИМЕН

В данный момент в системе известно максимально допустимое число имен (см. Приложение 5). Следует перевызвать систему, чтобы избавиться от ненужных имен, или использовать уже известные имена.

6. НЕТ МЕСТА В ДЗУ

Следует уничтожить ненужные файлы или вставить другой диск, закрыв предварительно все открытые файлы. При этой ошибке отключается весь ввод и вывод, настроенные на файл, при обмене с которым она возникла.

ПРИЛОЖЕНИЕ 5. ОСОБЕННОСТИ РЕАЛИЗАЦИИ СИСТЕМЫ "ШКОЛЬНИЦА" В ВЕРСИИ А1.1 И ЕЕ ОТЛИЧИЯ ОТ КАНОНИЧЕСКОГО ОПИСАНИЯ.

1. Функциональные отличия

1. Нет типизации имен и операции ТИПА.

2. Нет кадров, рисунков, таблиц и средств их обработки.

3. Обрабатываются только файлы текстового типа.

4. Нет объектного ввода-вывода (кроме ввода процедур и функций).

5. Работа цикла ДЛЯ-ИЗ (FOR-OF) по файлу не реализована.

6. Нет контроля инвариантов.

7. Программируемая обработка ошибок представлена процедурой сбора статистики по допущенным ошибкам СТАТ.

8. Разрешено описание нескольких процедур и функций в одном файле.

2. Синтаксические отличия

Лексика и синтаксис описываемых версий приводятся в Приложении 3. Стандартные процедуры и функции приводятся в Приложении 6.

3. Конфигурация технических средств и параметры системы

Наименование	Ограничение
Базовая ЭВМ	АГАТ
ОЗУ	48+16 К
ПЗУ	32 К (ШПЗУ)
ДЗУ	НГМД ЕС-5088
емкость	140 К

Наименование	Ограничение
телемонитор	цветной
печатающее устройство	-100
Объем доступного ОЗУ	32 К
Программные стеки	1.25К
Экран	
текстовые режимы:	32x32 цвет.сим.
	64x32 нецв.сим.
графические режимы:	ГСР: 128x128 цвет.точ.
	ГВР: 256x256 нецв.точ.
Число различных цветов	8
Рабочая память Редактора	8К
Число различных символов	256
Буфер вводимой строки	256 символов

4. Количественные ограничения и области допустимых значений, принятые в системе

Параметры ограничения	Ограничение
Макс.длина вводимой лексемы	255 сим.
Макс.целое число (модуль)	$2^{15}-1$
Точность представления дроб. чисел	12 знаков
Порядок дробных чисел	$-128 < x < 128$
Макс.порядок дроб.числа на вводе	127
Макс.целая часть дроб.числа на вводе (без незначащих нулей)	127 цифр
Макс.число различных имен (включая стандартные)	256
Макс.размер кортежей на вводе	255 эл-тов
Макс.мощность множества	255 эл-тов
Макс.число параметров проц.блока	127
Макс.число локальных имен проц. блока	255
Различаемая длина имени файла	30 литер

Параметры ограничения	Ограничение
Число одновременно открытых файлов	7
Размер диска в блоках	560
Макс.наполнение потока вывода	2 файла+экран или печ.устр.
	3 файла

ПРИЛОЖЕНИЕ 6. СТАНДАРТНЫЕ ФУНКЦИИ И ПРОЦЕДУРЫ РАПИРЫ

Помимо описанных в инструкции операций и предписаний в языке РАПИРА предусмотрены дополнительные возможности программирования, которые осуществляются с помощью набора стандартных процедур и функций.

В отличие от процедур и функций пользователя они постоянно находятся в памяти, а потому не нуждаются в описании и загрузке.

Стандартные процедуры и функции изначально присвоены некоторым именам, которые имеют абсолютную защиту (чтобы случайно не потерять доступ к системной программе) и не выдаются в каталоге имен (чтобы не загромождать его). В остальном они обладают всеми свойствами процедур и функций языка.

С помощью стандартных процедур и функций более полно реализуются возможности базовых ЭВМ. К ним относятся графика и работа с экраном, звуковые возможности, работа с клавиатурой, файлами, символьным набором в обход стандартных языковых средств, работа с памятью ЭВМ на уровне ячеек и программ в машинных кодах, работа с комплектом потенциометров (если они подключены к машине) и некоторые другие.

Поскольку эти возможности вводятся на различных стадиях знакомства с РАПИРОЙ, стандартные процедуры и функции также рас-

пределены по концентрам языка. В силу большой привязанности к возможностям ЭВМ описываемый комплект стандартных процедур и функций не является каноническим и может иметь некоторые отличия в реализациях языка на других ЭВМ.

Стандартные процедуры и функции Рапиры в версии AI.I

При описании используются следующие обозначения параметров:

X – произвольное число (целое или дробное),
N – целое число от 0 до 255,
A – целое число от 0 до 65535,
H – число I или 2,
T – произвольный текст,
Л – литера (односимвольный текст),
Ф – открытый файл,
П – произвольный объект.

Кх,Ку – координаты: целые числа, задающие координаты реальной точки экрана после выполнения необходимого пересчета относительно начала координат и масштабов (где допускается).

Пустыми скобками обозначается процедура (функция) без параметров.

1. Функция ABS(X)
   Результат – модуль числа X
2. Функция ЦЕЛЧ(X)
   Результат – целая часть числа X (для целого числа – само это число)
3. Функция SQRT(X)
   Результат – квадратный корень неотрицательного X

4. Функция ДСЧ( )

Результат – случайное число из промежутка [0;1[(выдаются только числа, кратные 1/65536).

5. Функция КОД(Л)

Результат – код символа, представленного литерой Л.

6. Функция АЛФ(N)

Результат – символ (литера) с кодом N

7. Функция ФТЕКСТ(А,Л)

Результат – текст длины А, состоящий из литер Л

8. Функция ФКОРТ(А,П)

Результат – кортеж, состоящий из А элементов П.

9. Функция КФ(Ф)

Результат – литера "Д", если в текущей позиции файла Ф находится символ "конец файла" "Н" – в противном случае.

10. Функция ЧТФ(Ф, N)

Очередные символов считываются из файла Ф, оформляются в виде текста и выдаются в качестве результата. Позиция указателя файла Ф сдвигается на N символов к концу файла; при исполнении функции возможна ситуация "встретился конец файла".

11. Функция КЛАВ( )

Ожидает нажатия клавиши на клавиатуре, результат – введенный символ (литера). Во время ожидания ввода в текущей позиции вывода на экране появляется мигающий курсор.

12. Функция НАЖАТО( )

Опрашивает строб клавиатуры, выдает находящийся в нем символ (в виде литеры) или пустой текст, если его нет.

Примечание: строб клавиатуры – это буферная область емкостью в один символ, через которую осуществляется ввод с клавиатуры в ЭВМ. Каждая нажатая клавиша записывает в строб символ, не-

зависимо от того, ожидает машина ввода или нет. После анализа очередного символа системой строб очищается. Данная функция позволяет, например, организовывать обрабатываемые прерывания в программе по нажатию определенной клавиши.

13. Функция РУЧКА(Н)

Выдает состояние потенциометра с номером Н в виде целого числа от 0 до 255.

14. Функция КНОПКА(Н)

Результат – литера "Д", если была нажата кнопка на потенциометре с номером Н, "Н" – в противном случае.

15. Процедура ПАУЗА(N)

Организует задержку исполнения программы на /10 сек.

16. Процедура ЗВОН()

Выдает звуковой сигнал (звонок).

17. Процедура ЗВУК (N1,N2)

Выдает звуковой сигнал, зависящий от частоты N2 и длительности N1.

18. Процедура ПРИГЛ(Л)

Устанавливает новое приглашение ко вводу для предписания ВВОД; будет выдаваться литера, переданная параметром процедуры. Чтобы избавиться от приглашения, следует указать любой управляющий символ, не несущий функциональной нагрузки при вводе, например, УПР-0 (см.Приложение 2).

19. Процедура ДЗУ(Н)

Устанавливает текущим активным ДЗУ дисковод с номером Н. В конфигурации ЭВМ "Агат" с одним дисководом использование этой процедуры не имеет смысла.

20. Процедура ОКНО (КхI,КуI,Кх2,Ку2)

Устанавливает на текущей текстовой странице окно для ведения диалога, одна из диагоналей которого задается координатами (КхI,КуI) и (Кх2.Ку2) (здесь координаты абсолютные и измеряются в символах текущего текстового режима от левого нижнего угла экрана вправо и вверх, начиная с нуля).

21. Функция ОКСИМ (Кх,Ку)

Результат – символ (литера), изображенный на экране в позиции с координатами Кх и Ку. Отсчет координат ведется в символах от левого нижнего угла текстового окна вправо (Кх) и вверх (Ку), начиная с нуля. Если позиция символа с указанными координатами находится вне окна, выдается сообщение об ошибке.

22. Процедура ПОЗ(Кх,Ку)

Перемещает курсор (текущую позицию вывода) к символу окна с указанными координатами. Значения координат не должны превышать соответствующих размеров окна.

23. Процедура РЕЖ(Т)

Управляет организацией графики и диалога в текстовом и графических режимах. Т – текст, представляющий собой список управляющих команд (см.также Раздел XI).

Формат текста: <к1> , <к2> ,..., <кN> , где <кi> – команды

Формат команды: <а> <д1> ... <дN>, где

<а> – адресуемая страница и режим:

Т1,Т2,Т3,Т4,Т5 – 5 цветных символьных страниц,

1, 2, 3, 4, 5 – 5 нецветных символьных страниц,

Н1,Н2,Н3,Н4,Н5 – 5 страниц графики низкого разрешения,

С – страница графики среднего разрешения,

В – страница графики высокого разрешения;

<дi> – действия с указанной страницей:

О – очистка,

П – показ,

Д – переключение диалога (только для символьных режимов) и показ

Г – переключение графику,

К – комбинирование с текущей текстовой страницей и показ.

Например: РЕЖ("Т1Д,СГОП") – установить диалог на первую текстовую страницу в цветном режиме, переключиться на среднюю графику, очистить и показать ее страницу.

24. Процедура ЦВЕТ(N)

Устанавливает текущий цвет рисования в графическом режиме. В несимвольных режимах имеют смысл следующие значения N :

0 – черный и далее через 8

1 – красный –"–

2 – зеленый –"–

3 – желтый –"–

4 – синий –"–

5 – фиолетовый –"–

6 – голубой –"–

7 – белый –"–

128–254 – инверсный к текущему

255 – пустой

в цветном символьном:

0–7 инверсные символы и далее через 16

8–15 мигающие символы и далее через 32

ПРИЛОЖЕНИЕ 7. КОМПЛЕКТУЮЩИЕ ПРОГРАММЫ

Настоящее приложение содержит краткие руководства по пакетам программ, поставляемым в пакете "Школьницы" на диске III. Эти программы имеют следующие назначения:

ИНСТР выводит на экран или бумагу настоящую инструкцию или указанные ее элементы;

ФУНКЦИИ г едоставляет для пользователя на языке РАПИРА набор тригонометрических и показательных функций;

СТАТ выводит на экран статистику по типам ошибок, допущенных в течение сеанса (после последнего запуска интерпретатора);

МУРАШ реализует исполнителя "муравей" интерпретатора РОБИК;

ЗАДМУР и РЕШМУР служат примерами начальной обстановки и программы для исполнителя "муравей";

МАШИНИСТ, ПР, НАЗАД реализуют исполнителя "машинист" интерпретатора РОБИК;

ЗАДМАШ и РЕШМАШ служат примерами начальной обстановки и программы для исполнителя "машинист";

АРИФ реализует исполнителя "МАЛЫШ", предоставляющего для языка РОБИК арифметические операции;

01 Модуль ИНСТР

Включение модуля ИНСТР не дополняет операционную обстановку новыми именами, все функции выполняются при работе стартовой процедуры, поэтому при аварийном выходе из модуля (например, "файл не найден" при ошибочной установке диска) следует выйти из режима ОШИБКА, выключить модуль и включить его снова.

Модуль ведет диалог в уплотненном (32x64) текстовом режиме, в инверсном (черном по белому) изображении. Первый вопрос диало-

га – о режиме вывода. Не задавайте вывода на бумагу, если к машине не подключен принтер: программа "повиснет" и систему придется перезагружать.

После установки режима на экране предлагается список имен файлов, образующих настоящую инструкцию. Содержание каждого файла соответствует его названию, точные сведения о размещении материала по файлам можно получить из "Содержания" инструкции (файл ИГ. ПОСТАВКА). Набором номера файла выбирается отдельный файл, 0 обозначает вывод всей инструкции, слишком большой номер – конец работы, отрицательный номер – вывод окончания инструкции, начиная с указанного файла.

При выводе отдельного файла и окончания инструкции запрашивается номер страницы и строки – это позволяет продолжить печать с прежним разбиением на страницы в случае нехватки или обрыва бумаги, случайного сбоя печати и т.п. Нумерация страниц начнется с указанного номера, причем на этой странице начало требуемого файла будет выдаваться после пропуска заданного числа строк.

После печати каждого файла выдача приостанавливается и на экран выдается обозначение диска, на котором располагается следующий файл.

В процессе печати "пробел" приостанавливает выдачу до нажатия любой другой клавиши, "РЕД" прекращает вывод текущего файла (для режимов выдачи нескольких файлов происходит переход к следующему).

Размеры страницы заданы в строках I45 (ширина строки – 70 символов) и I27 (длина страницы – 60 строк) текста модуля.

02 Модуль ФУНКЦИИ

Включение модуля ФУНКЦИИ вводит в операционную систему имена:

SIN
COS
TG
CTG
ARCSIN
ARCTG
EXP
LN
LG
СТЕП

Все функции имеют традиционный смысл. У функции СТЕП (x,y) первый аргумент – основание, второй – показатель степени. Обратные показательные функции (LN – натуральный и LG – десятичный логарифмы) в данной версии ограничены по диапазону аргумента от 1E-3 до 9999.9... То же ограничение относится к основанию степени, если показатель – дробное число.

@3 Процедура СТАТ

Выполнив запуск процедуры СТАТ (ВВОД ИЗ ДЗУ:СТАТ; СТАТ( )), можно получить на экране распределение допущенных в сеансе работы ошибок по типам диагностических сообщений. Затем предлагается очистить счетчики ошибок системы для того, чтобы обозначить начало следующего сеанса. При перезагрузке счетчики ошибок устанавливаются в 0.

@4 Исполнитель МУРАШ

Для запуска исполнителя необходимо вызвать интерпритатор РОБИК (директива РОБИК;) и включить его директивой

ВКЛ МУРАШ;

Полное множество предписаний исполнителя (МШИ) включает

ГЕН; – генерация задания;
ЗП "ИМЯ ФАЙЛА"; – запись обстановки на диск;
ЧТ "ИМЯ ФАЙЛА"; – чтение обстановки с диска;
ВВЕРХ;
ВНИЗ;
ВПРАВО;
ВЛЕВО; – движение "муравья";
ЭКРАН; – восстановление экрана.

При исполнении ГЕН на поле появляется курсор в виде мигающего квадрата синего цвета. Перемещение курсора выполняется стрелками, расстановка букв – соответствующими клавишами, стирание – клавишей "пробел".

Муравей устанавливается нажатием клавиши ⌧ , выход из режима генерации – перевод строки.

Принцип составления и выполнения заданий становится ясен на примере задания ЗАДМУР и программы РЕШМУР.

Чтобы познакомиться с ними, включите исполнителя и выполните следующие директивы:

ЧТ"ЗАДМУР";	(на экране начальная обстановка)
ЗАПОМНИТЬ ПРОЦЕДУРУ РЕШМУР;	(на экране текст программы)
<РЕД>, <перевод строки>	(выход через конец описания)
ЭКРАН;	(вновь начальная обстановка)
ВЫЗВАТЬ РЕШМУР;	(выполнение программы).

05 Исполнитель МАШИНИСТ

Запуск исполнителя производится директивой

ВКЛ МАШИНИСТ;

в интерпретаторе РОБИК.

МПИ исполнителя включает следующие директивы:

ГЕН; – генерация задания;

ЗП "имя файла" – запись обстановки на диск;

ЧТ"имя файла" – чтение задания;

ВПЕРЕД; –

НАЗАД; – движение состава;

ПРИЦЕПИТЬ ВАГОН;

ОТЦЕПИТЬ ВАГОН;– относятся к крайнему левому вагону;

ПЕРЕВЕСТИ СТРЕЛКУ; – относится к стрелке слева перед составом;

ЭКРАН; – отображение текущего состояния на экране.

После прохождения состава стрелки устанавливаются в исходное положение.

Файлы ЗАДМАШ и РЕШМАШ содержат условие и пример решения задачи для исполнителя МАШИНИСТ, их просмотр выполняется аналогично примеру для исполнителя МУРАШ.

Генерация задания происходит в два этапа: сначала формируется схема путей и стрелок, затем задается положение вагонов и исходное состояние стрелок. На первом этапе курсором служит попеременный вывод " > " и "=", нажатие " > " формирует стрелку, "=" – путь налево от курсора. На втором этапе курсор – мигающая цифра (номер стрелки) или буква (обозначение пути). Положение стрелки задается

нажатием "/" (стрелка вниз) или "\" (стрелка вверх). Размещение вагонов на каждом пути происходит слева направо, нажатие любой клавиши формирует вагон с изображением соответствующего символа. Заполнение пути заканчивается либо нажатием перевода строки, либо в случае полной занятости пути. Генерация заканчивается после заполнения всех "левых" путей и установки всех стрелок размещением локомотива на "правом" пути.

06. Исполнитель АРИФ

Включение исполнителя предоставляет в язык РОБИК четыре арифметических действия в виде следующего МПИ:

СЛОЖИТЬ выр С выр И ПРИСВОИТЬ имя;

ВЫЧЕСТЬ выр ИЗ выр И ПРИСВОИТЬ имя;

УМНОЖИТЬ выр НА выр И ПРИСВОИТЬ имя;

РАЗДЕЛИТЬ выр НА выр И ПРИСВОИТЬ имя;

где выр – выражение (в РОБИКе имя или константа),

имя – имя результата.

В реализации используется целая арифметика РАПИРы, поэтому остаются теми же диапазоны представления операндов и результатов.

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Входящий № сопроводительного докум. и дата	Подп.	Дата
	Измененных	Замененных	Новых	Аннулированных					

Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата

Изм. | Лист | № докум. | Подп. | Дата

Лист

Л-Т-15227 Подписано в печать 13/VIII-85
Формат бумаги 60x84/16. Бумага писчая. Ротапринт.
Объем 14,5 п.л. Тираж 250 экз. Заказ №153 Бесплатно.
Ротапринт НИИШОТСО АПН СССР. 103062, Москва, Лялин п., д. 3а.