ПЕРСОНАЛЬНАЯ ЭВМ
«ЭЛЕКТРОНИКА МС 0513»
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

ЯЗЫК ФОКАЛ
ОПИСАНИЕ ЯЗЫКА
00008-01.35.05

ПЕРСОНАЛЬНАЯ ЭВМ
«Электроника МС 0513»
Программное обеспечение

ЯЗЫК ФОКАЛ
Описание языка

00008-01.35.05

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Одним из алгоритмических языков ПЭВМ «Электроника МС0513» («Электроника БК-0011», далее микро-ЭВМ) является язык Фокал, обладающий полной преемственностью по отношению к стандартной версии языка.

Язык Фокал обладает широкими функциональными возможностями, содержит эффективные средства для отладки программ, лёгок для изучения и использования, содержит широкий набор стандартных встроенных функций.

Операторы языка состоят из коротких английских слов и позволяют описывать вычислительные процессы. Исходная программа и данные вводятся с клавиатуры микро-ЭВМ, результаты вычислений и тексты программ выдаются на экран видеоконтрольного устройства или бытового телевизора. Тексты программ хранятся на магнитной ленте кассетного магнитофона (например, типа "Электроника 302").

Работа с микро-ЭВМ возможна в двух режимах: диалоговом и программном. Во время работы пользователь получает подсказывающие, информационные и диагностические сообщения, что способствует и облегчает установлению "взаимопонимания" человека и машины.

Микро-ЭВМ предназначена для самого широкого круга пользователей: учащихся, студентов, преподавателей, инженеров, научных сотрудников. С её помощью можно проводить работы вычислительного характера, писать обучающие и игровое программы, строить системы автоматизированного сбора, контроля и обработки информации, системы управления научным экспериментом, системы управления бытовыми приборами и прочее.

1.1. Загрузка и запуск

Интерпретатор языка Фокал может быть записан:

1. В ПЗУ;

2. На магнитной ленте.

В первом случае необходимо иметь 2 ПЗУ, хранящих интерпретатор языка Фокал и драйвер-мониторную систему микро-ЭВМ «Электроника БК-0010».

Во втором случае на магнитной ленте необходимо иметь программы интерпретатора языка Фокал и драйвер-мониторной системы микро-ЭВМ «Электроника БК-0010». Для загрузки и запуска языка необходимы следующие команды:

```
@3;1C
@L
  FILE NAME?     FOCAL<BK>
```

После успешного считывания на экране появится сообщение:

```
  FILE:   FOCAL
  100000-140000
@
```

После команды

```
@100000G<ВК>
```

На экране появится приглашение Фокала к диалогу:

```
? 00 AT 0.00
ГОТОВНОСТЬ к РАБОТЕ
*
```

Примечание: поскольку клавиатура микро-ЭВМ может быть плёночной и клавишной, в данном документе использованы условные обозначения. Таблица соответствия условных обозначений обозначениям на клавиатуре микро-ЭВМ приведена в Приложении 2.

2. СТРУКТУРА ЯЗЫКА Фокал

2.1. АЛФАВИТ

Изучению любого алгоритмического языка всегда предшествует знакомство с набором его изобразительных средств.

В качестве алфавита языка Фокал используются буквы, цифры и ограничители.

Буквы. Латинский алфавит:

```
A, B, C, D, E, F, G, H, I, J, K, L, M,
N, O, P, Q, R, S, T, U, V, W, X, V, Z.
```

Фокал позволяет комментарии и текст примечаний записывать русскими заглавными и строчными буквами.

Цифры. Только арабские:

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
```

Цифры служат для образования чисел и номеров строк.

Ограничители. В качестве ограничителей используются:

А) знаки арифметических операций:

^	- возведение в степень,
*	- умножение,
/	- деление,
+	- сложение,
-	- вычитание;

Б) знаки, используемые как разделители:

=	- знак присваивания,
␣	- пробел,
ВВОД	- ввод информации в микро-ЭВМ,
.	- точка,
,	- запятая,
;	- точка с запятой;
:	- двоеточие.

[ ]
< > - скобки для записи выражений;
( )

В) знаки, используемые как управляющие:

"	- кавычки для печати текстовой информации,

% - процент для задания формата вывода чисел,
! - переход в начало новой строки,
$ - вывод на печать таблицы переменных;
? - знак трассировки,

Г) знаки, используемые при редактировании:

СДВП - сдвиг курсора на одну позицию вправо,
СДВЛ - сдвиг курсора на одну позицию влево,
РЗМ - размыкание,
СМК - смыкание.
УПС - удаление последнего введённого символа,
УСТР - удаление строки с позиции курсора,
СБР - сброс (очистка) экрана,
ВС - возврат в начало строки,
ГТ - горизонтальная табуляция,
ПОДЧ - сброс выполнения строки.

Д) знаки, управляющие действиями Фокала:

ШАГ - вызывает приостанов выполнения программы;
НР/СБР - вызывает коррекцию экрана и предоставляемой пользователю оперативной памяти;
СТОП - прерывание любых действий Фокала, вызывает сообщение об ошибке – 25 "останов по клавише "СТОП".

Язык позволяет дополнительно использовать специальные символы, имеющиеся на клавиатуре микро-ЭВМ.

2.2. РЕЖИМЫ РАБОТЫ И СТРУКТУРА ПРОГРАММЫ

Язык Фокал позволяет работать в двух режимах:

- диалоговом,
- программном.

В диалоговом режиме операторы вводятся без номера строки непосредственно после звёздочки (символ *), выполняются немедленно посте нажатия клавиши <ВК>. Строка без номера может содержать несколько операторов, разделённых ограничителем ";", и называется прямой командой. Выполнившись один раз, прямая команда не может выполняться вторично, пока не будет набрана снова.

Если данная строка оператора должна быть сохранена в оперативной памяти для дальнейшего использования, то перед ней необходимо поместить номер строки. Строки операторов, помеченные номерами, называются косвенными строками операторов или косвенными командами.

Формат косвенной команды:

* ТТ.ПП <ОПЕРАТОРЫ>,

где ТТ.ПП номер строки,

ТТ – номер группы строк,
ПП – номер шага.

Косвенная команда запоминается в памяти после нажатия клавиши <ВК> и не выполняется до тех пор, пока ей не будет передано управление.

Последовательность косвенных команд образует программу на языке Фокал, которая выполняется в порядке увеличения номеров строк, если нет дополнительных указаний, организуемых операторами DO, GOTO, IF, RETURN. Операторы, составляющие строку, реализуются в порядке их следования, начиная с первого, часть операторов строки может остаться нереализованной вследствие передачи управления другой строке.

Запуск программы на Фокале выполняется только после введения прямой команды GOTO или DO ALL. Подобный режим работы называется программным, после завершения выполнения программы или прямой команды выдаётся символ "*". Это означает, что система вышла в диалоговый режим и готова выполнять любые команды пользователя.

Для номеров строк программы можно использовать любые номера с 1.01 до 99.99 с шагом 0.01 за исключением тех, которые оканчиваются на .00. После точки необязательно указывать два знака, то есть 2.1 эквивалентно 2.10.

Разрешается также использовать номера от 100.1 до 127.9 с шагом 0.1 за исключением тех, которые оканчиваются на .0.

2.3. ОПЕРАТОРЫ

Язык Фокал включает 17 операторов. Операторы языка состоят из английских слов, использующих латинские заглавные буквы. Названия операторов несут смысловую нагрузку, например:

GOTO – перейти,

MODIFY – изменить и т.д.

Пользователю предоставляется возможность использовать сокращённую мнемонику в названиях операторов при программировании; все операторы можно обозначать одной или двумя начальными буквами названия оператора. Благодаря этому увеличивается скорость написания программы, уменьшается объём занимаемой программой оперативной памяти и, до некоторой степени, увеличивается скорость вычисления.

Язык Фокал включает следующий набор операторов:

ASK - для вывода данных с клавиатуры и вывода текстов примечаний на экран телевизора;
COMMENT - для комментариев, невыполняемых программных шагов;
DO - для выполнения подпрограммы (отдельной строки или группы строк);
ERASE - для уничтожения строки, группы строк, всей программы или переменных;
FOR - для организации циклов;

GOTO - для передачи управления строке с наименьшим номером (при отсутствии числовой метки) или строке с определённым номером (по метке);
HELP - для распечатки справочной информации;
IF - для передачи управления по условию;
KILL - для выработки импульса сброса внешних устройств;
LIBRARY - для выполнения операций с магнитофоном:
L GET – считывание файла программы в память,
L SAVE – запись файла программы на ленту,
L FGET – фиктивное чтение файла программы,
L INPUT – считывание файла переменных в память,
L OUTPUT – запись файла переменных на ленту,
L MOTOR – включение двигателя магнитофона,
L RESET – выключение двигателя магнитофона;
MODIFY - для редактирования программной строки;
PASS MONITOR - для передачи управления драйвер-мониторной системе;
QUIT - для останова программы и передачи управления пользователю;
RETURN - для завершения подпрограммы,
SET - для присвоения значения переменной;
TYPE - для вывода текстов примечаний, результатов вычислений, величин переменных на экран терминала;
VACANT - для распечатки количества свободной оперативной памяти в байтах;
WRITE - для вывода части или всего текста программы на экран телевизора;
XECUTE - для вызова и выполнения функций из библиотеки стандартных подпрограмм.

2.4. ЧИСЛА

В Фокале могут быть использованы любые десятичные числа в диапазоне от 10^{-38} до 10^{+38} числа могут быть со знаком "+" или "-", дробные числа – с десятичной точкой или в экспоненциальном формате. Формат – форма представления выводимых чисел. Формат определяет количество позиций в строке, отводимых под число.

Фокал преобразует все числа в экспоненциальный формат с шестью значащими цифрами. Если указано более шести цифр, число будет округлено до шести знаков. Константа в экспоненциальном формате состоит из двух частей: собственно константы и порядка (целой степени десяти). Порядок записывается целым числом, перед которым стоит буква e. Он может иметь знак "+" или "-".

Следующие ниже представления чисел идентичны:

```
70
70.00
7E+01
700E-01
70.00003
```

Запись вида N.E+n является неверной. Запись вида NE+n является верной (n – любое допустимое в Фокале число).

Заметим, что цифра "0" перед латинской заглавной буквой или набором букв информирует Фокал о том, что следующие после нуля буквы, за исключением буквы "E", следует рассматривать как цифры, равные порядковому номеру буквы в алфавите, а не как имя переменной, например:

```
* TYPE 0NO
155.0000
```

где $0NO = 10^1*N + 10^0*O = 10*14 + 15 = 155$

Стандартный формат, используемый Фокалом для ввода и вывода чисел, включает 8 десятичных знаков: 4 до десятичной точки и 4 после. Первые нули не печатаются, последние печатаются. Существенны, однако, только первые 6 знаков, причём шестой может быть неточным вследствие ошибок округления.

Пример:

```
* TYPE 55.55555
       55.5556*
```

Форму вывода чисел можно изменять с помощью указателя формата:

```
%W.0D,
```

где % – символ формата;

W – общее число знаков;

D – число знаков после точки.

Числа W и D должны быть всегда положительными целыми и не превышать 8.

```
* TYPE %4.02, 27.35
27.35 *
```

Общее число знаков здесь равно 4, а число знаков после запятой – 2.

После указателя формата ставится запятая, которая отделяет его от данных. Все последующие числа будут выводиться в указанном формате до тех пор, пока пользователь не изменит или не перезапустит систему.

В форматном символе вместо числа может использоваться название переменной за исключением букв латинского алфавита A и F.

Пример:

```
* SET A = 7; SET B = 3.02
* TYPE %8, A
7.00*
```

Если масштаб выводимого числа позволяет ему разместиться в отведённом количестве позиций, или формат представлен знаком "%" (без указателя количества позиций и положения десятичной точки), число выводится в виде мантиссы и порядка. Такая форма представления позволяет выводить числа в диапазоне от 10^{-36} до 10^{+36}.

Пример:

```
* TYPE %, 75.34
  0.753400E+02 *,
```

где E+02 = 10^2.

2.5. ПЕРЕМЕННЫЕ

Язык Фокал позволяет непосредственно обрабатывать только арифметическую информацию, поэтому в нём нет средств для описания типов данных. Для работы с символьной информацией используется встроенная функция FCHR.

Переменные в Фокале описываются их именами и, если переменная имеет индексы, то и значениями индексов.

Пример:

```
X(1), TOP(I), KE(5,6), Y, ZYX.
```

Имена переменных могут состоять из одного или нескольких символов. Первый символ должен быть буквой, но не буквой F, которая используется для имён функций. Остальные символы могут быть буквами или цифрами. Пользователь может написать имя переменной, состоящее более чем из двух символов, но Фокал использует только первые два для идентификации переменной.

Переменные, имена которых не начинаются с букв A и F, могут быть использованы в качестве номеров строк в операторах. Причина ограничения состоит в том, что буква A используется в операторах DO ALL, WRITE ALL, ERASE ALL, а буква F – в названиях функций.

Пример:

```
* 1.10 SET X = 3
* 1.20 DO X
```

Эти же операторы можно записать и в таком виде:

```
* 2.20 DO 3
```

В Фокале используются простые переменные и переменные с индексами. Переменные с индексами могут иметь не более двух индексов. Индексы заключаются в круглые скобки и разделяются запятой (в двухиндексных переменных).

Наряду с константами в качестве индексов могут использоваться арифметические выражения, но в качестве значения индекса будет принята целая часть значения выражения. Индексы могут принимать значения в диапазонах от -128 до +127 для двухиндексной переменной и от -32768, до +32767 для одноиндексной переменной. При превышении этих диапазонов значения берутся по модулю соответствующей границы.

Переменные в Фокале не требуют объявления. Пользователь должен знать, что переменные, появляющиеся в тексте программ или в прямых командах, интерпретируются Фокалом следующим образом. Если переменная уже была введена ранее, то интерпретатор обращается к таблице переменных и отыскивает её там, устанавливая указатель на ячейку, содержащую значение переменной. Если же переменная в таблице не обнаружена, то она заносится в память в том смысле, что отводится место для размещения, её имени, и очищаются ячейки для размещения её значения.

Если переменная "X" ранее не была введена, то команда:

```
* TYPE X
```

приведёт к распечатке:

```
0.0000 *
```

Переменная занимает в памяти 4 слова: одно слово для имени переменной, одно слово для индексов (для простых переменных это слово равно 0), два слова для значения. Простые переменные отыскиваются интерпретатором быстрее в том случае, когда их имена начинаются с разных символов, по сравнению с индексными переменными, имеющими одно имя, но разные индексы.

2.6. МАССИВЫ

Фокал позволяет использовать одно и двумерные массивы (или иначе, одно- и двухиндексные переменные).

Индекс может быть числом, названием переменной или арифметическим выражением.

Пределы изменения индекса:

От -128 до +127 для двумерного массива;

От -32768 до +32767 для одномерного массива.

Индексы заключаются в круглые скобки, а в случае двумерного массива разделяются запятой.

Например:

```
X(137), PO(I,J).
```

Массивы в "Фокале " не требуют объявления.

2.7. АРИФМЕТИЧЕСКИЕ ВЫРАЖЕНИЯ

Арифметические выражения в Фокале аналогичны математическим формулам. Арифметическое выражение может состоять из следующих элементов: чисел, переменных, функций, знаков арифметических операций и скобок. Следует иметь в виду, что отдельное число, отдельная переменная, отдельная функция также считается арифметическим выражением и является его частным случаем.

Арифметические выражения вычисляются в соответствии с приоритетом операций и наличием скобок, т.е. с учётом следующих обстоятельств:

1 Выражения, заключённые в скобки, имеют наивысший приоритет; это означает, что они вычисляются прежде выражений, стоящих вне скобок;

2 При отсутствии скобок порядок выполнения операций следующий:

- возведение в степень (^);
- умножение (*);
- деление (/);
- сложение (+);
- вычитание (-).

3 В конфликтных ситуациях, когда п. 1 и п. 2 недостаточны для определения порядка вычислений, следует помнить, что интерпретатор вычисляет выражение слева направо.

Если выражение содержит скобки внутри скобок, то сначала вычисляются внутренние скобки, а затем внешние.

В Фокале можно использовать 3 вида скобок: круглые ( ), квадратные [ ] и угловые < >. Интерпретатором они воспринимаются одинаково, но следует помнить, что каждой открывающейся скобке должна соответствовать закрывающаяся скобка того же типа.

Например:

```
* TYPE 2+2*2
  6.0000 *
* TYPE [2 + (2 + 2) * 2] / 5
  2.0000 *
```

3. ОПЕРАТОР ПРИСВАИВАНИЯ SET

Оператор присваивания set служит для присваивания переменной или индексной переменной, стоящей слева от знака " = ", вычисленного значения арифметического выражения, стоящего справа от знака " = ".

Синтаксис:

```
<ИМЯ ПЕРЕМЕННОЙ> = <АРИФМЕТИЧЕСКОЕ ВЫРАЖЕНИЕ>
```

Пример:

```
* SET X = 3.141593/180
```

В каждом операторе присваивания может быть инициализирована только одна переменная, т.е. конструкции следующего вида запрещены:

```
* SET X = Y = 5
* SET X = 5, Y = 5,
```

где X и Y – переменные.

При выполнении оператора присваивания SET вначале вычисляется значение выражения слева направо, а затем это значение присваивается переменной.

Если при выполнении оператора SET A=C, "C" есть переменная, ранее не использованная в программе, то её значение считается равным 0. Поэтому переменная "A" получает значение, равное 0.

Пример:

```
* SET D = 2
* SET B = 2; SET A = B * FSQT (D)
* TYPE A
  2.8284
```

4. ОПЕРАТОРЫ ВВОДА/ВЫВОДА

Язык Фокал обеспечивает работу пользователя со следующими устройствами:

- цветное или чёрно-белое видеоконтрольное устройство или бытовой телевизор (устройство отображения информации);
- микро-ЭВМ (устройство ввода);
- кассетный магнитофон (устройство ввода-вывода).

Видеомонитор и микро-ЭВМ предназначены для ввода-вывода прямых команд и программ пользователя, исходных данных, результатов измерений и вычислений.

Кассетный магнитофон используется для записи на магнитную ленту (МЛ) и считывания с МЛ программ пользователя и переменных.

Операторы ввода/вывода можно разделить на 2 группы:

- операторы, предназначенные для работы с видеомонитором и микро-ЭВМ (TYPE, ASK, WRITE, KILL);

- операторы группы LIBRARY для работы с кассетным магнитофоном.

4.1. ОПЕРАТОР ВЫВОДА ДАННЫХ TYPE

Этот оператор предназначен для вывода текстовой информации, результатов вычислений, таблиц переменных на экран видеомонитора, для управления выводом и форматом вывода. Оператор TYPE может использоваться в любом месте строки в прямых и косвенных командах.

Синтаксис:

```
TYPE <СПИСОК ЭЛЕМЕНТОВ ВЫВОДА>
```

Элементы вывода разделяются запятыми.

Элементами вывода могут быть:

- арифметическое выражение и, в частном случае, простая переменная, переменная с индексами;
- текст примечания – набор допустимый в языке Фокал символов, заключённых в кавычки,
- указатель формата – символ "%" либо "%W.0D";
- символ, управляющий выводом, "!";
- символ вывода значений всех переменных – $.

В случае, когда элементом вывода является арифметическое выражение, то вычисляется и выводится его значение.

Пример:

```
* SET A = 3; SET B = -7; TYPE "X=", A+B
  X = -4.0000 *
```

При запуске интерпретатора языка Фокал для вывода чисел по умолчанию устанавливается формат печати %8.04, т.е. Оператор TYPE выводит 8 знаков: 4 знака после децимальной точки, 4 знака до децимальной точки, причём, вместо ведущих нулей выводятся пробелы.

Установка другого формата вывода чисел производится следующим образом:

```
TYPE %W.0D
```

где $0 \le W \le 8$, $0 \le D \le 8$.

Установка экспоненциального формата вывода требует использования указателя формата %.

Пример:

```
* TYPE %, 3.951
  .395100E+01 *
```

Заданный в операторе TYPE формат вывода чисел действует до тех пор, пока не будет изменён другим указателем формата.

Управление выводимой информацией осуществляется символом управления выводом "!", а также использованием пробелов в текстах примечаний. Символ "!" переводит курсор в начало следующей строки.

Для вывода на экран значений всех переменных на данный момент используется символ " $ ".

Пример:

```
* SET A = 3; TYPE $
  S   A()= 3.0000 *
```

Элементы вывода, стоящие в операторе TYPE после символа " $ ", а также операторы в строке после TYPE $, игнорируются.

Следует заметить, что информация по оператору TYPE $ выдаётся как запись оператора SET в режиме прямых команд.

В операторе TYPE любые переменные, константы или выражения, исключая символы текста примечаний, должны разделяться или запятой, или точкой с запятой, или символом <ВК>.

4.2. ОПЕРАТОР ВВОДА ДАННЫХ ASK

Оператор ASK служит для ввода с клавиатуры микро-ЭВМ значений переменных, переменных с индексами и элементов массивов по запросу задачи. Для того чтобы программа в нужный момент сделала такой запрос, в неё включают оператор ASK.

Для удобства оператора ввод может сопровождаться печатью на экране видеомонитора текстов примечаний, заключённых в кавычки, с использованием символа "!".

Оператор ASK может использоваться как в прямых, так и в косвенных командах совместно с другими операторами языка Фокал.

Синтаксис:

```
ASK <СПИСОК ЭЛЕМЕНТОВ ВВОДА/ВЫВОДА>
```

Элементы ввода/вывода разделяются запятыми. Элементами в операторе ASK могут быть:

- простые переменные, переменные с индексами, элементы массивов;
- тексты примечаний – набор допустимых в языке Фокал символов, заключённых в кавычки (");
- символ, управляющий выводом, "!"
- указатели форматов;
- символ вывода значений всех переменных "$".

Пример:

```
* ASK X; TYPE !, "X = ", X
```

```
: 10
X = 10.0000 *
```

Ввод данных осуществляется после печати на экране оператором ASK символа ":". Число вводится следующим образом: с клавиатуры микро-ЭВМ последовательно вводятся символы, изображающие число. Ввод числа заканчивается одним из ограничителей:

- запятой;
- точкой с запятой;
- пробелом;
- клавишей <ВК>.

Одним оператором ASK можно ввести значения нескольких переменных.

Пример:

```
ASK X, Y(1), Z(3,2), !, "КОНЕЦ".
: 1
: 2
: 3
КОНЕЦ *
```

Переменной X присваивается значение 1, переменной Y(1) – значение 2, переменной Z(3,2) – 3.

Длина вводимой в ответ на символ ":" последовательности символов не должна превышать 23. Ввод 24-го символа вызывает вывод сообщения об ошибке 16 – "слишком много символов во входных данных" – и останов программы. В ответ на запрос оператора ASK может вводиться последовательность заглавных букв латинского алфавита от "A" до "Z", исключая "E". Это приводит к присвоению переменной значения, вычисляемого следующим образом:

```
* ASK X; TYPE !, X
: NO
155.0000 *
```

В этом случае переменная "X" получит значение:

$$X = 14*10^1 + 15*10^0 = 155,$$

т.к. это число будет представлено в Фокале как двузначное число с весами для единиц и десятков, равными порядковому номеру буквы в алфавите (A=1, B=2, C=3 .... Y=25, Z=26).

Оператором ASK в качестве значения переменной может быть присвоено вычисленное значение арифметического выражения, вводимого с устройства ввода. Если в качестве присваиваемого значения переменной вводится арифметическое выражение, то ему должен предшествовать знак "+" или "-".

Пример:

```
* 1.1 SET A=7; SET B=4
* 1.2 ASK "X=", X; TYPE !, X; QUIT
* GO
  X=: + A - B
  3.0000 *
```

При повторном счёте по программе и, следовательно, при повторном запросе на ввод оператором ASK текущие значения переменных можно сохранить. Для этого достаточно нажать клавишу " @ " (код 100). На экране будет напечатан символ " @ ".

Пример:

```
* 1.10 SET Y=2; TYPE Y, !
* 1.20 ASK X,Y,Z; QUIT
  2.0000
: 1: @: 3
```

Переменная Y сохранит ранее присвоенное ей значение 2.

Символ, управляющий выводом, "!" влияет только на расположение текстов примечаний.

Изменение формата вывода чисел осуществляется в операторе ASK указателем формата аналогично оператору TYPE. Числовые значения переменных, вводимых оператором ASK, не зависят от применённых указателей формата и, кроме того, могут вводиться в произвольной форме.

В операторе ASK может быть использован символ "$" для вывода на экран значений всех переменных аналогично оператору TYPE.

В сочетании с оператором FOR оператор ASK используется для ввода массива чисел, т.е. переменных с индексами.

Пример:

```
* FOR = 1,100; ASK X(I)
```

В примере вводится массив X(I) с клавиатуры микро-ЭВМ.

Использование в операторе ASK текстов примечаний позволяет оператору правильно ориентироваться при вводе чисел, особенно в тех случаях, когда данных много или тогда, когда работа программы связана с ответами на те или иные вопросы (например: игровые или обучающие программы).

Пример:

```
* 10.1 TYPE "ХОТИТЕ ПРОДОЛЖАТЬ?"
* 10.2 ASK !, "НАПЕЧАТАЙТЕ NO ИЛИ YES", ANS
* 10.3 IF (ANS-0NO) 10.5, 10.4, 10.5
* 10.4 QUIT
```

```
* 10.5 TYPE !, "НАЧИНАЕМ СНАЧАЛА"; GO
```

Цифра "0" в "0NO" информирует Фокал о том, что буквы, следующие после нуля, следует рассматривать как соответствующие им цифры, а не как имя переменной (см. раздел 2.4.).

В операторе ASK имеется возможность редактирования вводимых значений переменных.

Для сброса всей введённой строки (до 23-х символов) необходимо нажать клавишу "ПОДЧ".

После этого можно набрать значение переменной заново.

4.3. ОПЕРАТОР ВЫВОДА ТЕКСТА ПРОГРАММЫ WRITE.

Оператор WRITE предназначен для вывода на экран видеомонитора строки текста, группы строк, текста всей программы, находящейся в памяти микро-ЭВМ. Оператор WRITE может использоваться как в прямых, так и в косвенных командах в любом месте строки.

Синтаксис:

```
WRITE N
```

где N – номер строки, номер группы строк, слово ALL, пробел.

N может отсутствовать или быть задано:

- целым числом.
- дробным числом,
- именем переменной.

В случае, когда n является номером группы строк или строки, оператор write выводит эту группу или строку на экран видеомонитора.

Пример:

```
* WRITE 3.1. – печатает строку текста 3.1.
* WRITE 3    – печатает группу строк под номером 3
```

Выполнение оператора WRITE ALL или просто WRITE приводит к печати всего текста программы.

Имя переменной N не может начинаться с латинских букв "A" и "F". Буква "A" будет интерпретироваться как "ALL", "F" – как обращение к функции.

В случае, если N в операторе WRITE указывает на отсутствующую группу или строку, ошибка не фиксируется.

Оператор WRITE форматирует распечатку текста программы по группам.

4.4. ОПЕРАТОР СБРОСА ВНЕШНИХ УСТРОЙСТВ KILL.

Оператор KILL предназначен для установки всех внешних устройств в исходное состояние. При работе оператора KILL выполняется команда ассемблера RESET.

Оператор KILL не требует операндов. При его выполнении прекращается работа всех внешних устройств, работающих в режиме прерывания. Например, после выполнения оператора KILL в программном режиме запрещается прерывание от клавиатуры, и прервать выполнение программы становится невозможным до её окончания.

Оператор KILL может использоваться как в прямых, так и в косвенных командах в любом месте командной строки.

4.5. ОПЕРАТОРЫ ГРУППЫ LIBRARY

Операторы группы LIBRARY предназначены для организации файлов программ и переменных на магнитной ленте и управления двигателем кассетного магнитофона. Операторы группы LIBRARY могут использоваться как в прямых, так и в косвенных командах в любом месте командной строки.

Синтаксис:

```
LIBRARY <ПОДОПЕРАТОР> <ИМЯ ФАЙЛА>
```

Подоператор определяет функцию, выполняемую оператором LIBRARY. Фокал включает в себя семь подоператоров: SAVE, GET, FGET, OUTPUT, INPUT, MOTOR и RESET. Поиск на магнитной ленте осуществляется по имени файла. Имя файла может содержать любые буквы латинского и русского алфавитов (строчные и заглавные), а также цифры. Ограничений на порядок следования символов в имени нет. Длина имени файла – не более 15 символов.

Подоператоры MOTOR и RESET пишутся без имени файла.

4.5.1. ОПЕРАТОР LIBRARY SAVE.

Для записи программы на Фокале на магнитную ленту используется оператор LIBRARY SAVE.

```
LIBRARY SAVE <ИМЯ ФАЙЛА>
```

Пример:

```
L S KUBIK
```

После набора команды необходимо нажать клавиши <ПУСК> и <ЗАПИСЬ> на передней: панели магнитофона и клавишу <ВК> на клавиатуре микро-ЭВМ.

Программа, написанная на Фокале и находящаяся в памяти микро-ЭВМ, запишется на магнитную ленту с именем KUBIK.

Запись программы оператором LIBRARY SAVE на магнитную ленту осуществляется с того места ленты, на которое была установлена магнитная головка. При успешном окончании записи на экран видеомонитора выводится символ готовности "*". В противном случае – сообщение об ошибке. Текст программы в памяти сохраняется.

4.5.2. ОПЕРАТОР LIBRARY GET

Для считывания файла программы, написанной на Фокале, с магнитной ленты в память микро-ЭВМ используется оператор LIBRARY GET.

```
LIBRARY GET <ИМЯ ФАЙЛА>
```

Пример:

```
* L G KUBIK1
```

После набора команды необходимо нажать клавишу <ВК> на клавиатуре микро-ЭВМ и клавишу <ПУСК> на передней панели магнитофона.

Фокал осуществит поиск файла программы по имени KUBLK1 на магнитной ленте (начиная с того места ленты, на котором была установлена головка) и считывание файла KUBIK1 в оперативную память. При этом старый текст программы и переменные в памяти стираются.

При успешном окончании считывания на экран дисплея выводится имя файла и символ готовности "*".

В противном случае – соответствующее сообщение об ошибке.

Для повторного считывания неправильно считанного файла программы необходимо перемотать ленту в начало и повторить считывание.

В процессе поиска файла на ленте на экран видеомонитора распечатываются имена промежуточных файлов.

4.5.3. ОПЕРАТОР LIBRARY FGET

Оператор фиктивного чтения LIBRARY FGET осуществляет поиск файла программы по имени. Но запись его в оперативную память не производится. Таким образом, текст программы и переменные в оперативной памяти сохраняются. После операции фиктивного чтения магнитная головка будет установлена в конце указанного файла программы на ленте.

Порядок выполнения оператора аналогичен оператору LIBRARY GET.

Оператор фиктивного чтения LIBRARY FGET полезен для нахождения конца последнего файла перед записью программы на ленту оператором LIBRARY SAVE, для распечатки и просмотра имён файлов, записанных на ленте.

4.5.4. ОПЕРАТОР LIBRARY OUTPUT

Для записи данных на магнитную ленту используется оператор LIBRARY OUTPUT.

```
LIBRARY OUTPUT <ИМЯ ФАЙЛА>
```

Данные на ленту переписываются со своими именами, т.е. при последующем их считывании все переменные будут иметь те же имена.

Пример:

```
* L O ДАННЫЕ
```

После набора команды необходимо нажать клавиши <ПУСК> и <ЗАПИСЬ> на передней панели магнитофона. А затем клавишу <ВК> на клавиатуре микро-ЭВМ. Переменные, находящиеся в памяти, запишутся на магнитную ленту с именем "ДАННЫЕ".

Запись данных оператором LIBRARY OUTPUT на магнитную ленту осуществляется с того места ленты, на которое была установлена магнитная головка. После нажатия клавиши <ВК> осуществляется сокращение полезной части экрана до 1/4. Далее выполняется запись переменных, которая сопровождается характерным писком. При успешном окончании записи полезная часть экрана полностью восстанавливается (если Фокал не находился в режиме расширенной памяти, см. п. 6.7), и на экран видеомонитора выводится символ готовности. В противном случае выдаётся сообщение об ошибке, текст программы и переменные в памяти сохранятся.

Пользователю следует знать, что переменные на ленту записываются в последовательные файлы блоками по 511 переменных. Каждый файл имеет имя, указанное в операторе L O, но кроме этого файлы переменных помечаются индексами, которые записаны в 16-ой позиции имени.

Пример:

```
* 1.1. FOR 1=1,520; S A(I)=1
* 1.2. L O PRIM
```

В строке 1.1. осуществляется накопление переменных A(I).

В строке 1.2. выполняется запись массива переменных A(I) и переменной I на магнитную ленту. При этом на ленту запишется 2 файла:

PRIM_____0 длиной 511 переменных и

PRIM_____1 длиной 10 переменных.

При считывании в память записанного массива переменных оператором LIBRARY INPUT необходимо указать имя PRIM без индексов.

Индексы в именах файлов данных формируются Фокалом автоматически. Имена файлов программ записываются без индексов.

4.5.5. ОПЕРАТОР LIBRARY INPUT

Для считывания файла данных с магнитной ленты в память микро-ЭВМ используется оператор LIBRARY INPUT.

```
LIBRARY INPUT <ИМЯ ФАЙЛА>
```

Пример:

```
* L I PRIM
```

Считать с ленты в память переменные (записанные в п. 4.5.4).

После набора команды необходимо нажать клавишу <ВК> на клавиатуре микро-ЭВМ и клавишу <ПУСК> на передней панели магнитофона.

Фокал осуществит поиск файла переменных по имени PRIM_____0 на магнитной ленте (начиная с того места ленты, на которое была установлена магнитная головка) и считывание файла PRIM в оперативную память.

При этом текст программы и старые переменные в памяти сохраняются. Поэтому, если старые переменные не нужны, то их надо убрать оператором E перед выполнением оператора L I.

Перед началом поиска файла переменных Фокал сократит полезную часть экрана до 1/4, а после окончания считывания полезная часть экрана восстановится (если не был установлен режим расширенной памяти, см. п. 6.7).

В процессе поиска на ленте файла PRIM на экран терминала распечатываются имена всех просматриваемых файлов, а при нахождении и считывании файла PRIM – имена PRIM_____0 и PRIM_____1.

При успешном окончании считывания на экран выводится символ готовности "*". В противном случае – соответствующее сообщение об ошибке.

Для повторного считывания неправильно считанного файла переменных необходимо перемотать ленту в начало и повторить считывание.

4.5.6. ОПЕРАТОРЫ LIBRARY MOTOR И LIBRARY RESET

Операторы LIBRARY MOTOR и LIBRARY RESET предназначены соответственно для включения и выключения двигателя магнитофона.

Примеры:

```
1) * LIBRARY MOTOR
```

включить двигатель

```
2) * LIBRARY RESET
```

выключить двигатель

В операторах LIBRARY SAVE, LIBRARY GET, LIBRARY FGET, LIBRARY OUTPUT и LIBRARY INPUT включение и выключение двигателя магнитофона осуществляется автоматически.

5. ОПЕРАТОРЫ УПРАВЛЕНИЯ ПРОГРАММОЙ

Операторы Фокала в программном режиме выполняются друг за другом в порядке возрастания номеров строк, если среди них не встречаются операторы, управляющие ходом программы. Фокал имеет широкие возможности управления ходом программы, реализуемые следующими операторами:

- оператор безусловного перехода GOTO;
- рекурсивный оператор DO;
- оператор выхода из программы RETURN;
- оператор условного перехода IF;
- оператор цикла FOR;
- оператор выполнения XECUTE;
- оператор останова программы QUIT.

Управление ходом программы заключается в передаче управления строке с указанным номером.

5.1. ОПЕРАТОР БЕЗУСЛОВНОГО ПЕРЕХОДА GOTO

В Фокале в качестве оператора безусловного перехода используется оператор GOTO.

Действие оператора GOTO сводится к тому, что следом за ним выполняется строка с номером, указанным в операторе GOTO.

Оператор GOTO может использоваться как в прямых, так и в косвенных командах в любом месте командной строки.

Синтаксис:

```
GOTO N
```

где N – номер строки, которой требуется передать управление, может отсутствовать или быть задан:

- дробным числом;
- именем переменной.

Если N в операторе GOTO отсутствует, то управление передаётся строке Фокала с наименьшим номером. Это свойство оператора GOTO можно использовать для запуска программы, если начало программы совпадает со строкой с наименьшим номером.

Если в качестве N использовано имя переменной, то её значение будет интерпретироваться как номер строки, которой требуется передать управление.

Ссылка оператором GOTO на несуществующую строку вызывает сообщение об ошибке 5 – "несуществующий номер строки".

Операторы Фокала, находящиеся после оператора GOTO в командной строке, не исполняются и рассматриваются как комментарии.

Примеры:

```
1.   * GOTO
```

Управление передаётся строке с наименьшим номером.

```
2.   * GOTO 3.15; TYPE "ПЕРЕХОД"
```

Управление передаётся строке 3.15. Оператор TYPE не выполняется.

```
3.   * 7.1 SET M = 3 * 2.1; GO M
```

При передаче управления на строку 7.1 будет вычислено значение переменной M. Оператор GOTO передаёт управление строке с номером 6.2.

5.2. ОПЕРАТОР БЕЗУСЛОВНОГО ПЕРЕХОДА DO (С ВОЗВРАТОМ)

Оператор DO предназначен для передачи управления на заданную командную строку или группу строк с последующим возвратом на оператор, следующий за DO. Оператор DO может быть использован в режиме прямых и косвенных команд в любом месте командной строки.

Синтаксис:

```
DO N
```

Где N – номер командной строки. Либо группы строк, на которые передаётся управление, или ALL.

N может быть задано:

- целым числом;
- дробным числом;
- именем переменной.

Если в качестве N использовано имя переменной "A", то оператор будет интерпретироваться как DO ALL. Оператор DO ALL (в краткой форме D A) или DO вызывает исполнение всей программы, начиная со строки с минимальным номером (аналогично GOTO).

В случае, когда N ссылается на отсутствующий в программе номер строки или группы строк, то работа программы прерывается и выдаётся сообщение об ошибке 6 – "несуществующий номер группы или номер строки в операторе DO".

В случае, когда N – дробное число, т.е. указывает на номер строки, то по оператору DO управление передаётся этой строке, а после её исполнения – оператору, следующему за оператором DO.

Пример:

```
* 1.1. SET V = 3.4; DO V; SET X = A + B
* 3.4. T V
```

В примере после выполнения DO V выполняется строка 3.4. После этого управление передаётся оператору SET X = A + B.

В случае, когда N – целое число, т.е. указывает на группу строк, то управление передаётся на строку с минимальным в данной группе номером. Строки группы выполняются в порядке возрастания номеров строк, если в них нет операторов перехода, либо до исчерпания номеров строк данной группы в программе, либо до встречи с оператором RETURN. После этого управление передаётся оператору, следующему непосредственно за DO.

Примеры:

```
1.   * 1.1. DO 7
     * 1.2. GO 7.4.
     * 7.1. T 1,!
     * 7.2. T 2,!;RETURN
     * 7.3. TYPE 3,!
     * 7.4. TYPE 4;QUIT
     1.0000
     2.0000
     4.0000*

2.   * 1.1. DO 5
     * 1.2. GO 6.1
     * 5.1. TYPE 1,!
     * 5.2. TYPE 2,!
     * 6.1. TYPE 6.1;QUIT
     1.0000
     2.0000
     6.1000 *
```

Работа оператора DO имеет некоторые особенности. Если в строке или группе строк, на которую передаётся управление оператора DO, находятся операторы перехода GOTO или IF, то они вызывают передачу управления только на одну строку, после выполнения которой (если в этой строке в свою очередь не содержатся операторы перехода) управление перейдёт к оператору, следующему за DO.

Примеры:

```
1.   * 1.1. DO 5.1
```

```
     * 1.2. TYPE 3,!; QUIT
     * 5.1. TYPE 1,!; GO 6.1
     * 6.1. TYPE 2,!
     1.0000
     2.0000
     3.0000 *

2.   * 1.1. DO 5.1
     * 1.2. TYPE 3; QUIT
     * 5.1. TYPE 1,!; GO 6.1
     * 5.2. TYPE 2,!
     * 6.1. TYPE 4,!
     * 6.2. TYPE 5,!
     1.0000
     4.0000
     3.0000 *
```

5.3. ВЛОЖЕННЫЕ ПОДПРОГРАММЫ

Операторы DO могут быть вложены, т.е. во время выполнения оператора DO может встретиться другой оператор DO. Глубина вложенности операторов DO определяется размером свободной памяти в стеке интерпретатора. При большой глубине вложенности операторов DO или большой глубине рекурсии возможно переполнение стековой памяти интерпретатора. Эта ситуация вызовет сообщение об ошибке 9 – "переполнение стека".

Максимальное число вложений – 44

Пример:

```
* 1.1 DO 5
* 1.2 TYPE 4; QUIT
* 5.1 TYPE 1,!; DO 7
* 5.2 TYPE 2,!
* 7.1 TYPE 3,!
1.0000
3.0000
2.0000
4.0000 *
```

5.4. ОПЕРАТОР ВЫХОДА ИЗ ПОДПРОГРАММЫ RETURN

Оператор RETURN используется совместно с оператором DO или функцией FSBR и служит для возврата "досрочно" к оператору, следующему за оператором DO или за оператором, содержащим обращение к функции FSBR.

Оператор RETURN не требует операндов.

Операторы, следующие за оператором RETURN в этой же строке, никогда не выполняются, поэтому, после него есть смысл размещать только комментарии.

Если в группе строк встречается оператор RETURN, то строки группы, следующие за оператором RETURN, не выполняются.

Пример:

```
* 1.1 ASK Y; DO 3
* 1.2 TYPE X; QUIT
* 3.1 IF (Y) 3.2,3.3,3.4
* 3.2 SET X=-1; RETURN
* 3.3 SET X= 0; RETURN
* 3.4 SET X= 1; RETURN
```

Здесь в группе 3 размещена подпрограмма, обращение к которой производится оператором DO 3. Завершение работы подпрограммы, а, следовательно, и выход из неё возможны на строках 3.2, 3.3 или 3.4. Этим и объясняется наличие в каждой из них оператора RETURN.

Пример:

```
* 1.1 DO 5
* 1.2 TYPE 6;Q
* 5.1 TYPE 1,!
* 5.2 TYPE 2,!; GO 5,7
* 5.3 TYPE 3,!
* 5.7 TYPE 4,!
* 5.8 TYPE 5,!
1.0000
2.0000
4.0000
5.0000
6.0000 *
```

В этом случае оператор GOTO передаёт управление на строку из группы 5, поэтому выполняются обе строки 5.7 и 5.8 в обход строки 5.3. Для того, чтобы выполнялась только строка 5.7, её надо изменить так:

```
* 5.7 TYPE 4,!; RETURN
```

5.5. ОПЕРАТОР ПЕРЕХОДА ПО УСЛОВИЮ IF.

Нумерация строк определяет естественный порядок выполнения операторов программы. Однако большинство задач редко укладывается в такую линейную схему. Очень часто в задачах встречаются условия, от выполнения которых зависит последующая схема вычислений.

Средством для анализа этих условий и последующего изменения естественного порядка выполнения программы является оператор IF.

Оператор IF позволяет передавать управление на нужные программные строки по результатам сравнения значения некоторого арифметического выражения.

Оператор IF может использоваться как в прямых, так и в косвенных командах в любом месте командной строки.

Синтаксис:

```
IF(A) N1[,N2[,N3]]
```

где N1, N2, N3 – номера командных строк, куда следует передать управление после сравнения;

A – арифметическое выражение.

В таблице 1 даются возможные виды записи условного оператора IF и порядок его выполнения:

Таблица 1.

Вид оператора	Значение A	Куда передаётся управление
IF (A) N1, N2, N3	A<0	командной строке N1
	A=0	командной строке N2
	A>0	командной строке N3
IF (A) N1, N2	A<0	командной строке N1
	A=0	командной строке N2
	A>0	следующему оператору
IF (A) N1	A<0	командной строке N1
	A=0, A>0	следующему оператору

Пример:

```
* 1.1 ASK Y; IF(Y-10)1.2,1.3,1.4
* 1.2 TYPE "Y<10";QUIT
* 1.3 TYPE "Y=10";QUIT
* 1.4 TYPE "Y>10";QUIT
```

В качестве номеров строк N1, N2, N3 могут использоваться имена переменных. В случае, когда указанная в операторе IF командная строка отсутствует, то работа программы прекращается, и выдаётся сообщение об ошибке 5 – "несуществующий номер строки".

Если в операторе IF в качестве арифметического выражения используются нецелые числа, то проверка на 0 не всегда осуществима из-за особенностей плавающей арифметики. Чтобы ликвидировать эту проблему, программист должен либо избежать использования нецелой арифметики в

операторе IF, либо проверять, меньше ли нужного отклонения сравниваемая дробная величина, и если меньше, то положить её равной 0.

Пример:

```
*1.1 ASK X
*1.2 IF(FABS(FSQT(1-(FCOS(X))-FSIN(X))-1E-6)1.3;
     TYPE "ОШИБКА";QUIT
*1.3 TYPE "ВЕРНО";QUIT
```

5.6. ОПЕРАТОР ЦИКЛА FOR

Оператор FOR используется для записи алгоритмов, в которых параметр меняется с некоторым шагом, т.е. для организации циклов и итераций. Циклы – это участки программы, которые в процессе её выполнения повторяются указанное число раз.

Синтаксис:

```
FOR A=B[[C,],D];[ОПЕРАТОРЫ],
```

где A - параметр цикла;
B и D - соответственно начальное и конечное значения параметра цикла;
C - шаг изменения параметра цикла.

Операторы, расположенные после оператора FOR в той же командной строке, образуют область действия цикла FOR. В область действия цикла включаются и те строки, на которые передаётся управление операторами DO, GOTO и IF, стоящими в той же командной строке после оператора FOR.

Параметром цикла может быть любая переменная. В качестве B, C и D могут использоваться арифметические выражения. Если величина "C" в записи оператора отсутствует, то по умолчанию цикл выполняется с шагом 1.

Оператор цикла работает следующим образом:

1. Параметру цикла присваивается значение "B" (SET A=B);
2. Выполняются операторы из области действия цикла;
3. К параметру цикла прибавляется величина шага цикла (SET A=A+C);
4. Если A<D или A=D, то выполняется переход на операторы из области действия цикла, если A>D, управление передаётся командной строке, следующей за строкой, содержащей оператор цикла.

Таким образом, операторы из области действия цикла всегда выполняются хотя бы один раз при A=B.

Когда в записи оператора цикла отсутствуют "C" и "D", т.е. для конструкции вида "FOR A=B; операторы", оператор FOR аналогичен оператору SET.

Как следует из п. 4, параметр цикла A при своём изменении возрастает, т.е. должно выполняться условие D>B и C>0. При D<B при любом значении "C" операторы из области действия цикла выполняются один раз.

Пример:

```
* FOR X=-10,5,0; TYPE X
-10.0000    -5.0000    0.0000 *
```

При программировании циклов следует иметь в виду:

1. При выходе из цикла при его исчерпании (A>D) параметр цикла сохраняет своё значение на момент выхода из цикла;
2. Выход из цикла может быть осуществлён только по исчерпании параметра цикла;
3. Параметр цикла может применяться операторами из области действия цикла. После этого цикл выполняется при этом изменённом значении, увеличенном на шаг цикла, если A<D. Таким образом, операторами из области действия цикла можно управлять числом повторений цикла;
4. Применение шага цикла "C" или конечного значения параметра цикла "D" операторами из области действия цикла не изменяет числа повторений цикла;
5. Циклы могут быть вложенными, например:

```
* FOR I=1,2;FOR J=3,4;TYRE I,!,J,!
1.0000
3.0000
1.0000
4.0000
2.0000
3.0000
2.0000
4.0000
*
```

В примере цикл по "J" является внутренним для цикла по "I" и при каждом фиксированном "I", "J" пробегает все свои значения, с которыми выполняются операторы области действия цикла.

Пример:

```
* 1.1 FOR I=1,2,TYPE "I=",I,!;DO 3;GO 5.1;
* 1.2 TYPE "END",!,TYPE "I=",I;QUIT
* 3.1 TYPE 3.1,!
* 3.2 TYPE 3.2,!;RETURN
* 3.3 TYPE 3.3,!
* 5.1 TYPE 5.1,!
* 5.2 TYPE 5.2,!
```

```
I=1.0000
  3.1000
  3.2000
  5.1000
I=2.0000
  3.1000
  3.2000
  5.1000
END
I=3.0009
*
```

5.7. ОПЕРАТОР ВЫПОЛНЕНИЯ ФУНКЦИИ XECUTE

Оператор XECUTE производит выполнение функции из библиотеки стандартных подпрограмм Фокала или арифметического выражения.

При этом вывод на экран видеомонитора вычисленного значения не производится. Оператор XECUTE может использоваться как в прямых, так и в косвенных командах в любом месте строки.

Синтаксис:

```
XECUTE <АРИФМЕТИЧЕСКОЕ ВЫРАЖЕНИЕ>
```

Обычно оператор XECUTE используется для выполнения функций.

Примеры:

```
1. * XECUTE FCHR(65,66)
     AB *
2. * XECUTE FCHR(FCHR(-1))
     Эхо-печать вводимых символов.
3. * XECUTE FSBR(10,5)
4. * XECUTE FX(1,10000)
5. * XECUTE FK(8,12)
```

Список встроенных функций библиотеки стандартных подпрограмм Фокала приведён в гл. 8.

5.8. ОПЕРАТОР ОСТАНОВА ВЫПОЛНЕНИЯ ПРОГРАММЫ QUIT

Оператор QUIT предназначен для останова выполнения программы и перевода Фокала в режим прямых команд. При встрече с ним работа программы приостанавливается с сохранением всех значений переменных, и на экране видеомонитора печатается символ "*".

Работа программы может быть продолжена с необходимой строки с помощью оператора GOTO N.

Оператор QUIT может использоваться в любом месте командной строки.

Оператор QUIT не имеет операндов.

6. ОПЕРАТОРЫ ДЛЯ ОТЛАДКИ ПРОГРАММ

Отладка программ включает в себя следующие моменты:

- редактирование исходного текста, внесение в него изменений и дополнений;
- трассировку – управление распечаткой текста программы во время её выполнения;
- автоматическую выдачу кода и текстового сообщения об ошибке.

В Фокале имеется возможность управления программой с помощью управляющих клавиш.

Кроме того, в Фокале существуют сервисные операторы для определения памяти, свободной для текста программы, и распечатки справочной информации.

6.1. ОПЕРАТОР УДАЛЕНИЯ ПРОГРАММНЫХ СТРОК И ПЕРЕМЕННЫХ ERASE.

Оператор ERASE предназначен для уничтожения в оперативной памяти либо только всех переменных, либо только текста программы (всего или заданных строк), либо и всего текста и всех переменных.

Синтаксис:

```
ERASE N
```

где N может быть:

- номером строки;
- номером группы строк;
- символ "T" или "TEXT";
- символ "A" или "ALL";
- пусто или пробел.

В качестве N может использоваться имя переменной, не начинающееся с символов "A" и "F".

Ссылка оператором ERASE на строки с номерами 1 – 127 не вызывает ошибку. Ссылка на строки, не входящие в этот интервал, вызывает ошибку с кодами 1, 4 или 5.

Примеры:

* ERASE 7.1 - уничтожает строку 7.1 и все переменные;
* ERASE 7 - уничтожает группу строк 7 и все переменные;
* ERASE TEXT - уничтожает только весь текст программы (оставляет переменные в памяти);
* ERASE ALL - уничтожает весь текст программы и все переменные;
* ERASE - уничтожает все переменные (оставляет текст программы в памяти).

Оператор ERASE без N для уничтожения всех переменных может использоваться как в прямых, так и в косвенных командах в любом месте строки. С помощью такого оператора ERASE можно динамически управлять памятью, отводимой под переменные, в процессе выполнения программы.

Оператор ERASE N для уничтожения текстов и переменных также может использоваться как в прямых, так и в косвенных командах, но следует иметь в виду, что после его выполнения на экран видеомонитора выводится символ "*".

6.2. ОПЕРАТОР РЕДАКТИРОВАНИЯ ПРОГРАММНЫХ СТРОК MODIFY

Оператор MODIFY используется для экранного редактирования программной строки, находящейся в оперативной памяти, без перепечатывания всей строки заново. Оператор MODIFY обычно используется в режиме прямых команд. Операторы, следующие за оператором MODIFY, не выполняются.

Синтаксис:

```
MODIFY N
```

где N – номер строки, которую требуется скорректировать.

После того, как пользователь набрал MODIFY и нажал клавишу <ВК>, Фокал печатает на экране видеомонитора содержимое нужной строки. Далее пользователь может корректировать строку, используя редактирующие клавиши клавиатуры микро-ЭВМ:

- СДВП – сдвиг курсора вправо на одну позицию;
- СДВЛ – сдвиг курсора влево на одну позицию;
- РЗК – размыкание строки вправо с позиции, где расположен курсор;
- СМК – смыкание строки справа от позиции, где расположен курсор;
- УПС – удаление последнего введённого символа;
- УСТР – стирание строки с позиции курсора;
- СБР – сброс экрана (выполняется очистка экрана и сбрасывается выполнение набранной строки);

- ВС – возврат курсора в начало строки;
- ГТ – горизонтальная табуляция (сдвиг курсора на 8 позиций вправо);
- ПОДЧ – выполняется сброс набранной строки.

Процесс редактирования строки с использованием оператора MODIFY заканчивается нажатием клавиши <ВК>, после появления на экране видеомонитора символа "*" система готова для выполнения любых действий пользователя.

Если в процессе редактирования произошло переполнение строки, то Фокал автоматически заканчивает выполнение оператора MODIFY.

Оператор MODIFY не может использоваться для изменения номера строки.

Строку в программе можно изменить и без оператора MODIFY. Чтобы заменить номер строки надо набрать номер строки и новые операторы.

Редактирование при вводе новой строки или прямой команды также осуществляется редактирующими клавишами. Если в процессе ввода и редактирования новой строки или прямой команды произошло переполнение строки, то Фокал" игнорирует введённую строку и выдаёт сообщение об ошибке – "переполнение входного буфера".

Ссылка оператором MODIFY на несуществующую строку вызывает сообщение об ошибке 5 – "несуществующий номер строки".

Пользователю следует знать, что редактирующие клавиши управляют положением курсора в строке, но сами коды редактирующих клавиш в памяти не записываются. Коды клавиш СТРЛ, СТРП, СТРВ, СТРН, СПВ, СПН, СЛН, СЛВ, СУГ, вообще игнорируются, т.е. эти клавиши не управляют положением курсора, и их коды не записываются в память.

В Фокале имеется возможность записи кодов вышеуказанных клавиш в память. Для этого необходимо нажать комбинацию клавиш "НР" и УПС. Эта процедура вызовет индикацию в информационной строке вверху экрана слова "РЕД", что означает переход в режим "РЕД". Для перевода Фокала в обычный режим необходимо нажать НР / УПС, при этом слово "РЕД" в информационной строке исчезнет.

В режиме "РЕД" коды редактирующих клавиш и клавиш СТРЛ, СТРП, СТРВ, СТРН, СПВ, СПН, СЛН, СЛВ, СУГ записываются в память микро-ЭВМ и изменяю, положение курсора на экране, но их изображение на экране не высвечивается. Это позволяет рисовать на плоскости экрана различные фигуры, что может быть полезно при составлении текстов примечаний в операторах TYPE и ASK.

Пользователю следует иметь в виду, что в режиме "РЕД" никакая редакция введённой информации невозможна. Для коррекции введённой информации необходимо выйти из режима "РЕД", скорректировать строку и,

при необходимости, вернуться в режим "РЕД". Максимальное количество символов в строке равно 80.

Пример:

На экране дисплея необходимо нарисовать следующую картину

```
   AAA
BBB   CCC
```

Для этого необходимо использовать оператор TYPE и вышеприведённую картину оформить в виде примечания:

```
* 1.1. TYPE " AAA
           BBB   CCC"
```

При наборе этой строки после первых кавычек необходимо перейти в режим "РЕД" и для получения нужной картины набрать следующую последовательность символов:

```
СТРП, СТРП, СТРП, СТРП, СТРП, А, А, А, СТРН,
СТРЛ, СТРЛ, СТРЛ, СТРЛ, СТРЛ, СТРЛ, В, В, В,
СТРП, СТРП, СТРП, С, С, С.
```

После этого надо выйти из режима "РЕД" нажатием НР/ УПС, закрыть текст картинки кавычками и нажать клавишу <ВК>. При выполнении данной строки на экране дисплея появляется вышеприведённая картинка.

При запуске системы автоматически устанавливается обычный режим редактирования.

6.3. ОПЕРАТОР COMMENT

Оператор COMMENT служит для введения в программу комментариев для удобства чтения программы. Всё, что следует после этого оператора до конца командной строки, не выполняется, но оператором WRITE может быть выведено на экран видеомонитора. При вводе программ в оперативную память комментарии размещаются в ней и, следовательно, сокращают объём памяти, предназначенной для размещения переменных.

Синтаксис:

```
COMMENT < ТЕКСТ КОММЕНТАРИЯ >
```

При трассировке программы на экран видеомонитора выдаётся сам оператор COMMENT и весь текст комментария до первого ограничителя. Если при трассировке необходима печать всего комментария, можно использовать символ ":" вместо пробелов.

6.4. ОПЕРАТОР РАСПЕЧАТКИ СПРАВОЧНОЙ ИНФОРМАЦИИ HELP

Оператор HELP распечатывает на экране терминала справочную информацию о языке Фокал. Оператор HELP используется в прямых командах.

Оператор HELP не требует операндов.

6.5. ОПЕРАТОР VACANT

Оператор VACANT предназначен для определения оперативной памяти, не занятой текстом программы на Фокале.

Оператор VACANT не требует операндов. Выполнение оператора VACANT приводит к выдаче на экран видеомонитора следующей информации:

```
" СВОБОДНО ------ Б.ОЗУ "
```

где ------ есть восьмеричное число, определяющее количество оставшейся свободной памяти для текста программы, в байтах.

6.6. ИСПОЛЬЗОВАНИЕ ТРАССИРОВКИ

Трассировка используется для управления распечаткой частей текста программы на экране видеомонитора.

Чтобы выполнить трассировку, пользователь вставляет знак трассировки "?" в строку команды в любом месте. Фокал печатает каждый следующий за знаком трассировки символ до тех пор, пока не встретится следующий знак "?", или пока управление не передастся пользователю.

Пример:

```
* 1.1. SET A=1; SET B=2; SET C=3
* 1.2. TYPE ? A + B + C ?; QUIT
* G
  A + B + C     6.0000 *
```

Символ "?" в текстах примечаний в операторах TYPE, ASK не воспринимается Фокалом как знак трассировки.

6.7. ИСПОЛЬЗОВАНИЕ УПРАВЛЯЮЩИХ КЛАВИШ

В Фокале управляющими являются клавиши <ШАГ>, <НР/СБР> и <СТОП>.

Клавиша <ШАГ> используется в диалоговом и программном режимах. Нажатие клавиши <ШАГ> вызывает приостанов всех действий Фокала. Продолжение выполнения осуществляется только нажатием клавиши <ВК>.

Это может быть полезно при просмотре текста программы оператором write, при отладке программы и т.д.

Комбинация клавиш <НР> и <СБР> (<НР/СБР>) используется в диалоговом режиме. Нажатие клавиш <НР/СБР> вызывает сокращение полезной части экрана на 3/4, за счёт чего пользователю предоставляется ещё 4К слов оперативной памяти для текстов программ и переменных. Вторичное нажатие клавиш <НР/СБР> вызывает обратные действия: экран терминала восстанавливается, и память, предоставляемая пользователю, сокращается до исходного значения. Клавиши <НР/СБР> следует использовать в случае появления ошибок 10 или 11, т.е. при нехватке места в памяти для текстов программ или переменных.

При нажатии клавиш <НР/СБР> переменные в памяти стираются, а текст программы сохраняется. Если при попытке перехода из режима расширенной памяти в исходную текст находящейся в памяти программы велик, то никакие действия не производятся, и выдаётся сообщение об ошибке 10 – "переполнение памяти текстом программ".

Клавиша <СТОП> используется в диалоговом и программном режимах. Нажатие клавиши <СТОП> приводит к прерыванию любых выполняемых Фокалом действий и печати сообщения об ошибке 25 – "останов по клавише СТОП".

6.8. ДИАГНОСТИКА ОШИБОК

Фокал может обнаружить многие ошибки пользователя. В случае обнаружения ошибки при выполнении программы или прямой команды пользователя на экран видеомонитора выдаётся сообщение об ошибке.

Сообщение об ошибке включает признак ошибки, номер ошибки, номер строки или группы, в которой обнаружена ошибка, и текстовое сообщение о причине ошибки.

Например:

```
* ? 03 AT 0.00
НЕПАРНЫЕ СКОБКИ
```

Здесь знаки "?" и "AT" это признак ошибки. 03 – код ошибки, 0.00 – номер строки или группы. Полный список сообщений об ошибках приведён в приложении 1.

7. ОПЕРАТОР PASS MONITOR

Оператор PASS MONITOR предназначен для передачи управления системной программе микро-ЭВМ – монитору.

Синтаксис:

```
PASS MONITOR
```

Для возвращения в Фокал из монитора необходимо набрать F и <ВК>.

При возвращении в Фокал текст программы и переменные в памяти стираются.

8. ВСТРОЕННЫЕ ФУНКЦИИ ФОКАЛА

Для вычисления часто встречающихся функций, преобразования кодов, работы с графической информацией и выполнения других операций в языке Фокал существует набор встроенных функций, к которым пользователь может обращаться из языка Фокал. Встроенные функции составляют библиотеку стандартных подпрограмм.

Обращение к встроенным функциям осуществляется по имени, первым символом которого обязательно является буква "F". За именем функции следует список её аргументов, заключённый в скобки. У некоторых функций аргументы отсутствуют. В этих случаях за открывающейся скобкой непосредственно следует закрывающаяся. Функция после выполнения получает значение функции от последнего аргумента или значение последнего аргумента.

Функции могут быть включены в арифметическое выражение. Библиотека встроенных функций языка Фокал включает 21 функцию:

FSIN (X)	- синус (аргумент в радианах);
FCOS (X)	- косинус (аргумент в радианах);
FTAN (X)	- тангенс (аргумент в радианах);
FASIN (X)	- арксинус (| X | < 1);
FACOS (X)	- арккосинус (| X | < 1);
FATAN (X)	- арктангенс;
FLOG (X)	- натуральный логарифм (X>0);
FLOG10 (X)	- десятичный логарифм (X>0);
FEXP (X)	- показательная функция e;
FSGN (X)	- знаковая часть числа;
FITR (X)	- целая часть числа;
FABS (X)	- абсолютная величина числа;
FRAN ()	- генератор случайных чисел;
FSQT (X)	- корень квадратный;
FCHR (X_1,X_2, ...X_n)	- ввод/вывод символьной информации;
FX (КОП, АДРЕС, ДАННЫЕ)	- управление общей шиной;
FSBR (ГРУППА, АРГУМЕНТ)	- функция, программируемая пользователем;
FK (X, Y)	- установка курсора по координатам X и Y;

FT (КОП, X, Y) - формирование точки по координатам X и Y;

FV (КОП, X, Y) - формирование вектора по координатам X и Y

FP (КОП, МАСКА) - работа с портом ввода/вывода.

Другие функции, не вошедшие в библиотеку стандартных подпрограмм, можно применять как функции пользователя FSBR.

8.1. МАТЕМАТИЧЕСКИЕ ФУНКЦИИ

Для вычисления синуса угла предназначена функция FSIN:

Формат:

```
FSIN (X)
```

где X – значение аргумента в радианах, может быть арифметическим выражением.

Пример:

```
* TYPE FSIN (3.14159/4)
  0.7071 *
```

Если угол задан в градусах, то для преобразования аргумента используется множитель (π/180).

Пример:

```
* TYPE FSIN (45 * 3.14159/180)
  0.7071 *
```

Для вычисления косинуса угла предназначена функция FCOS.

Формат:

```
FCOS (X)
```

где X – значение аргумента в радианах, может быть арифметическим выражением.

Пример:

1)

```
* TYPE FCOS (2 * 3.14159)
  1.0000 *
* TYPE FCOS (0.5000)
  0.8776 *
```

2) Косинус угла в градусах:

```
* TYPE FCOS (45 * 3.14159/180)
  0.7071 *
```

Для вычисления тангенса угла предназначена функция FTAN.

Формат:

```
FTAN (X)
```

где X – значение аргумента в радианах, может быть арифметическим выражением.

ПРИМЕРЫ:

1)
```
* T FTAN (0.87)
  1.1853 *
```

2) Тангенс угла в градусах:

```
* TYPE FTAN (45 * 3.14159/180)
  1.0000
```

Для вычисления арксинуса предназначена функция FASIN. Если аргумент по модулю больше 1, выдаётся сообщение об ошибке 20 – "в функциях FASIN и FACOS аргумент по модулю > 1". Функция FASIN получает вычисленное значение угла в радианах.

Формат:

```
FASIN (X)
```

где X – арифметическое выражение.

Пример:

```
* TYPE FASIN (0.5)
  0.5236 *
```

Для вычисления арккосинуса предназначена функция FACOS. Если аргумент по модулю больше 1, выдаётся сообщение об ошибке 20 – "в функциях FASIN и FACOS аргумент по модулю > 1". Функция FACOS получает вычисленное значение угла в радианах.

Формат:

```
FACOS (X)
```

где X – арифметическое выражение

Пример:

```
* TYPE FACOS (0.5)
  1.0472 *
```

Для вычисления арктангенса предназначена функция FATAN. Угол, полученный при вычислении, выражен в радианах.

Формат:

```
FATAN (X)
```

где X – арифметическое выражение.

Пример:

```
* TYPE FATAN (3.91)
  1.3204 *
```

Для вычисления натурального логарифма предназначена функция FLOG. Если аргумент меньше или равен нулю, выдаётся сообщение об ошибке 19 – "логарифм нуля или отрицательного числа".

Формат:

```
FLOG (X)
```

где X – арифметическое выражение.

Пример:

```
* TYPE FLOG (5.17)
  1.6429 *
```

Для вычисления десятичного логарифма предназначена функция FLOG10. Если аргумент меньше или равен нулю, выдаётся сообщение об ошибке 19 – "логарифм нуля или отрицательного числа".

Формат:

```
FLOG10 (X)
```

где X – арифметическое выражение.

Пример:

```
* TYPE FLOG10 (10)
  1.0000 *
```

Для вычисления экспоненты e^x предназначена показательная функция FEXP.

Формат:

```
FEXP (X)
```

где X – арифметическое выражение

Пример:

```
* TYPE FEXP (5)
  148.4130 *
```

Для получения знаковой части числа, являющегося аргументом функции, предназначена функция FSGN. Если аргумент меньше нуля, то функция FSGN получает значение -1, если равен нулю, FSGN получает значение 0, если больше нуля, то функция получает значение +1.

Формат:

```
FSGN (X)
```

где X – арифметическое выражение

Примеры:

1)

```
* TYPE FSGN (6-4)
  1.0000 *
```

2)

```
* TYPE FSGN (0)
  0.0000 *
```

Для получения целой части выражения, являющегося аргументом функции, предназначена функция FITR.

Формат:

```
FITR (X)
```

где X – арифметическое выражение.

Функция принимает значение целой части аргумента со знаком аргумента.

Примеры:

```
* TYPE FITR (55.27)
  55.0000 *
* TYPE FITR (7.86)
  7.0000
* TYPE FITR (-4.1)
  - 4.0000 *
```

Для получения абсолютной величины выражения предназначена функция FABS.

Формат:

```
FABS (X)
```

где X – арифметическое выражение.

Пример:

```
* TYPE FABS (-17)
  17.0000 *
* TYPE FABS (30.05)
  30.05 *
```

Для вычисления квадратного корня из арифметического выражения применяется функция FSQT. Если выражение имеет отрицательное значение,

то выдаётся сообщение об ошибке 17 – "корень квадратный из отрицательного числа".

Формат:

```
FSQT (X)
```

где X – арифметическое выражение.

Пример:

```
* TYPE FSQT (625)
  25.0000 *
```

8.2. ФУНКЦИЯ СЛУЧАЙНЫХ ЧИСЕЛ (FRAN)

Функция FRAN предназначена для генерации псевдослучайных чисел, распределённых равномерно в интервале (-1, 1).

Формат:

```
FPAN ()
```

Пример:

```
* FOR I=1,100; SET R(I)=FRAN() -
```

программа генерирует сто случайных чисел

```
* TYPE FRAN()
 -0.3916 *
* TYPE FPAN()
  0.1659 *
```

При повторном применении FRAN () "перетасовывает" числа и выдаёт, таким образом, непредсказуемую последовательность случайных чисел.

Функция FRAN (1) повторяет последовательность:

```
* XECUTE FRAN(1)
```

FRAN (1) полезно использовать при отладке программ, так как одна и та же последовательность случайных чисел производится каждый раз, когда программа повторяется.

8.3. ФУНКЦИЯ ВВОДА-ВЫВОДА СИМВОЛЬНОЙ ИНФОРМАЦИИ (FCHR)

Функция FCHR предназначена для приёма и/или печати символов, соответствующих кодам КОИ-8.

Формат функции:

```
FCHR (СПИСОК АРГУМЕНТОВ)
```

Основное назначение этой функции состоит в преобразовании символов, вводимых с клавиатуры, из кода КОИ-8 в десятичную форму, или, наоборот, из десятичной формы в коды КОИ-8 с выводом символа на экран.

Для преобразования символов, вводимых с клавиатуры, из кода КОИ-8 в десятичную форму аргумент функции FCHR должен быть отрицательным. Например, FCHR (-1). При выполнении такой функции Фокал будет ожидать ввода символа с клавиатуры микро-ЭВМ. После нажатия нужной клавиши сама функция примет десятичное значение кода КОИ-8 введённого символа.

Пример:

```
TYPE FCHR (-1)
```

После нажатия клавиши <ВК> машина ожидает ввода символа. Нажмём, например, клавишу "А" (не печатается на экране), и на экран выдаётся её десятичное значение:

```
65.0000 *
```

Таким образом, происходит перевод из КОИ-8 в десятичную форму.

Для преобразования из десятичной формы в коды КОИ-8 с выводом символа на экран видеомонитора аргумент функции FCHR должен быть положительным. Целая часть величины аргумента (трактируемой как десятичное число) будет преобразована в соответствующий символ КОИ-8, который будет передан на экран видеомонитора. Значение функции FCHR при этом есть целая часть значения аргумента.

Примеры:

1)
```
* TYPE FCHR (65.8)
  A 65.0000 *
```

2)
```
* XECUTE FCHR (66)
  B *
```

В случае, когда используется несколько аргументов, функция примет десятичное значение последнего аргумента.

Примеры:

3)
```
* TYPE FCHR (65,66)
  AB 66.0000 *
```

4)
```
* TYPE FCHR (65,66, -1)
  AB 67.0000 *
```

В примере 4) печатаются А и В, соответствующие восьмеричным кодам КОИ-8, равным 101 и 102, или десятичным числам 65 и 66, после этого с

клавиатуры считывается следующий символ (пусть "C"), десятичное значение которого (67) и становится значением самой функции.

Функция FCHR может использоваться рекурсивно, например:

```
* SET Z = FCHR(FCHR(-1)).
```

В процессе выполнения этой командной строки будет принят один символ с клавиатуры. FCHR (-1) принимает соответствующее десятичное значение (например, если вводимый символ A, то FCHR (-1) принимает значение 65 в десятичном исчислении). Далее, так как происходит выполнение FCHR (65), то будет напечатан символ A (для данного примера). Переменной Z оператором SET будет присвоено значение 65. Последнее объясняется тем, что оператор SET присваивает переменной значение арифметического выражения, стоящего в правой части.

Пример:

```
* SET X = FCHR (65); TYPE !; X,!
  A
  65.0000 *


SET X = FCHR (193); TYPE !; X, !
  a
  193.000 *
```

Функция FCHR может быть использована в обучающей программе для анализа ответов обучаемого.

Например:

Ответ (A, B или C) на поставленный вопрос может анализироваться так:

```
* 1.01. SET REQ = FCHR (-1)
* 1.03. IF (REQ – 65) 3.10, 4.10, 3.10
* 3.10. T "НЕВЕРНО, ПОПРОБУЙТЕ ЕЩЕ РАЗ", !; G 1.01
* 4.10. T "ВЕРНО",!
```

Ответ "A" является верным.

Для определения десятичных значений символов, расположенных на клавиатуре микро-ЭВМ, можно воспользоваться следующей короткой программой:

```
1.1. TYPE FCHR (-1);G
```

После её запуска на экран видеомонитора будут выводиться десятичные значения символов, вводимых с клавиатуры микро-ЭВМ.

Пользователю следует быть внимательным при использовании функции FCHR с положительным аргументом. Например, прямая команда X FCHR (12) выполнит очистку экрана, т.к. код 12 есть десятичный код клавиши

"СБР". Выполнение строки 1.1 FOR I=1,20; X FCHR (25) вызовет сдвиг курсора на 20 позиций вправо, т.к. код 25 есть десятичный код клавиши СТРП.

Пример:

```
* 1.01 C ЗАПОЛНЕНИЕ ЭКРАНА ШАХМАТНОЙ СЕТКОЙ
* 1.02 C С ПОМОЩЬЮ ФУНКЦИИ FCHR
* 1.1 X FCHR (12)
* 1.2 FOR K=1,3; D 2
* 1.3 QUIT
* 2.1 FOR I=1,40; D 3
* 2.2 FOR I=1,40; D 3.2; D 3.1
* 3.1 FOR I=1,4; X FCHR (127)
* 3.2 FOR I=1,4, X FCHR (32)
```

8.4. ФУНКЦИЯ УПРАВЛЕНИЯ ОБЩЕЙ ШИНОЙ (FX)

Функция FX используется для выполнения дополнительных действий с периферийными устройствами и для обращения к ячейкам памяти.

Формат:

```
FX (КОД ОПЕРАЦИИ, АДРЕС ОБЩЕЙ ШИНЫ, ДАННЫЕ)
```

Первый аргумент "код операции" может принимать значения -1, 0, +1 и определяет тип выполняемой операции:

+1 – чтение слова;

0 – чтение слова логическое с конъюнкцией;

-1 – запись слова по адресу общей шины

Второй аргумент "адрес общей шины" должен быть или восьмеричным числом, или названием переменной и должен быть чётным.

"Адрес общей шины" есть адрес ячейки памяти или регистра периферийного устройства, к которому производится обращение.

Третий аргумент "данные" должен быть десятичным числом в интервале от -32768 до +32767 или арифметическим выражением.

При операции чтения (1) выполняется чтение слова по "адресу общей шины". Третий аргумент не нужен. Функция принимает десятичное значение считанного слова.

При операции запись (-1) выполняется запись слова ("данных") по адресу общей шины. Фокал обладает способностью самозащиты, т.е. запрещается запись в ячейки оперативной памяти, используемые интерпретатором Фокала для своей работы (0 – 2000). При попытке записи в запрещённую область оперативной памяти выдаётся сообщение об ошибке 13 – "запрещённый адрес шины в функции FX". При операции запись функция FX принимает десятичное значение записываемого слова.

При операции чтение слова с конъюнкцией (0) выполняется операция конъюнкции слова памяти и данных. Функция получает десятичное значение результата операции конъюнкции.

Если в функции FX указан несуществующий адрес на общей шине, то выдаётся сообщение об ошибке 21 – "несуществующее устройство".

Примеры:

1)
```
* T FX(1,1000).
```

Данные считываются из ячейки с адресом 1000.

2)
```
* X FX (-1,7550,1).
```

По адресу 7550 записывается 1.

3)
```
* T FX(0, 4320, AH).
```

Информация общей шины выбирается в зависимости от результата операции конъюнкции "И" над содержимым адреса 4320 и значением переменной AH и выводится на экран видеомонитора.

В нижеприведённой таблице содержится справочная информация для работы с функцией FX. В левой графе таблицы записаны десятичные значения чисел для обращения к битам слова. В средней графе – восьмеричные значения чисел. В правой – номера битов слова, соответствующие числам слева.

Десятичное число	Восьмеричное число	Номера битов в слове
- 32768	100000	15
16384	40000	14
8192	20000	13
4096	10000	12
2048	4000	11
1024	2000	10
512	1000	9
256	400	8
128	200	7
64	100	6
32	40	5
16	20	4
8	10	3
4	4	2
2	2	1
1	1	0
-32640	100200	15 и 7
192	300	7 и 6

255	377	7,6,5,4,3,2,1,0
-256	177400	15,14,13,12,11,10,9,8

8.5. ФУНКЦИЯ, ПРОГРАММИРУЕМАЯ ПОЛЬЗОВАТЕЛЕМ (FSBR)

Программируемая пользователем функция FSBR используется для выполнения нумерованной группы строк (или одной строки) как подпрограммы. Подпрограмма представляет собой функцию одной специальной переменной, обозначаемой "&", которая играет роль формального параметра.

Формат обращения к функции FSBR:

```
SET V = FSBR (N, ARG)
```

или

```
XECUTE FSBR (N, ARG),
```

где V - название переменной, которой присваивается вычисленное значение функции;
N - номер группы, реализующей алгоритм программируемой функции (число или переменная);
ARG - аргумент функции FSBR (число, переменная или арифметическое выражение).

Интерпретатор Фокала, встретив в тексте программы обращение к FSBR, вычисляет значение аргумента функции и присваивает полученное значение специальной переменной "&", затем осуществляется передача управления на первую строку программы с номером N. Возврат из программы, реализующей алгоритм программируемой функции, осуществляется по команде RETURN либо по исчерпанию строк, в группе. После завершения выполнения подпрограммы последнее вычисленное значение и становится значением функции.

Пример вычисления экспоненты:

```
* 1.10 A "ВВЕДИТЕ ЗНАЧЕНИЕ АРГУМЕНТА", ARG; C EXP
* 1.20 SET Y = FSBR (7, ARG)
* 1.30 T !, Y; QUIT
* 7.10 IF(&^2-0.01)7.2; SET &=&/2; D7; S &=&^2; RETURN
* 7.20 SET &=1+&+&^2/2+&^3/6+&^4/24+&^5/120+&^6/720
```

В подпрограмме под номером 7 используется специальная переменная "&", которой автоматически присваивается начальное значение, равное ARG. Последнее значение "&" становится окончательным значением функции FSBR.

Функция FSBR может использоваться рекурсивно.

Если строк группы недостаточно для записи подпрограммы, то можно использовать строки других групп, передавая им управление оператором DO. Это объясняется тем, что FSBR, передавая управление группе, работает как оператор DO и, исчерпав данную группу, возвращает управление команде, следующей за вызовом FSBR.

Если алгоритм, реализующий подпрограммную функцию, имеет несколько точек выхода, то каждая строка, содержащая точку выхода, должна оканчиваться командой RETURN.

Пример вычисления факториала:'

```
* 1.10 A N
* 1.20 T !, FSBR (5, N)
* 1.30 QUIT
* 5.10 I (1-&) 5.2; RETURN
* 5.20 SET & = &*FSBR(5,&-1);RETURN
```

8.6. ФУНКЦИЯ УПРАВЛЕНИЯ ПОЛОЖЕНИЕМ КУРСОРА (FK)

Функция FK предназначена для установки курсора по координатам X и Y на плоскости экрана.

Формат функции:

```
FK (X, Y)
```

где X - десятичное значение координаты X, $0 \le X \le 63$
Y - десятичное значение координаты Y, $0 \le Y \le 23$

Если значения X, Y выходят за указанные пределы, то X принимается за значение остатка от деления X на 64, за Y – значение остатка от деления Y на 24.

X и Y могут быть любым арифметическим выражением. Изменение положения курсора на экране функцией FK осуществляется относительно верхнего левого угла экрана.

Функция FK принимает десятичное значение координаты Y.

Пример:

```
* X FK (32, 12)
```

Выполнение данной команды поместит курсор в центр экрана, т.е. сместит курсор на 32 позиции вправо по оси X и на 12 позиций вниз по оси Y относительно верхнего левого угла экрана.

Выполнение команды X FK (-32, -12) также поместит курсор в центр экрана, но он будет смещаться относительно нижнего правого угла на 32 позиции влево по оси X и на 12 позиций вверх по оси Y.

8.7. ФУНКЦИИ ДЛЯ РАБОТЫ С ГРАФИЧЕСКОЙ ИНФОРМАЦИЕЙ

Для формирования точки на экране терминала используется функция FT.

Формат функции:

```
FT (КОП, X, Y)
```

где КОП = 0 – стирание точки;
1 – запись точки;
X и Y - десятичные значения координат, 0 ≤ X≤ 511, 0 ≤ Y≤ 255, по которым формируется точка на экране. X и Y могут быть любым арифметическим выражением.

Функция FT принимает десятичное значение координаты Y.

Координаты X и Y отсчитываются от верхнего левого угла экрана.

Пример:

Построить синусоиду

```
* 1.1 F I=0,360, X FT (1,I,100+50*FSIN(I*3.14159/180))
```

Для формирования вектора на экране видеомонитора используется функция FV.

Формат функции:

```
FV (КОП, X, Y)
```

где КОП = 0 – стирание вектора;
1 – запись вектора;

0 ≤ X ≤ 511

0 ≤ Y ≤ 255

X и Y – десятичные значения координат конца вектора. X и Y могут быть любым арифметическим выражением.

Функция FV принимает десятичное значение координаты Y.

Перед выполнением функции FV должны быть определены координаты начала вектора. Это можно сделать, например, функцией FT.

Примеры:

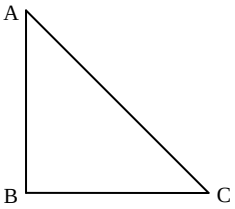
1)
```
* 1.1 X FT (0, 0, 0,); X FV (1, 400, 200)
```

Будет построен вектор с координатами начала вектора (0, 0) и конца вектора (400, 200)

2)
```
* 1.1 X FT (0, 100, 100)
```

```
* 1.2 X FV (1, 100, 200)
* 1.3 X FV (1, 200, 200)
* 1.4 X FV (1, 100, 100)
```

Данная программа построит прямоугольный треугольник ABC



Строка 1.1. устанавливает координаты точки A.

Строка 1.2. строит катет AB, строка 1.3. строит катет BC, а строка 1.4. строит гипотенузу CA.

8.8. ФУНКЦИЯ РАБОТЫ С ПОРТОМ ВВОДА/ВЫВОДА (FP)

Функция FP предназначена для управления портом ввода-вывода. К порту могут быть привязаны устройства пользователя, например, различные бытовые приборы.

Формат функции:

```
FP (КОП, МАСКА)
```

где КОП - код операции, может принимать значение 0,1,2,3
0 – значение регистра ввода по маске;
1 – очистка битов регистра вывода по маске;
2 – установка битов регистра вывода по маске;
3 – чтение регистра вывода по маске.

МАСКА - восьмеричное число в диапазоне от 0 до 177777 или переменная с десятичным значением в диапазоне от -32768 до +32767.

Порт ввода/вывода содержит 2 регистра:

- регистр ввода информации в микро-ЭВМ из порта;
- регистр вывода информации из микро-ЭВМ в порт.

Регистр ввода только читается, регистр вывода читается и записывается.

При операциях чтения с регистра ввода (КОП = 0) и регистра вывода (КОП = 3) выполняется побитная конъюнкция (операция логическое "И") между считываемыми из регистра данными и маской. Если маска равна

177777, то читаются все 16 битов слова. Если маска равна 377, то читается младший байт слова в регистре, если – 177400, то читается старший байт.

Если КОП=1, то выполняется очистка битов регистра вывода по маске. Очистка битов регистра вывода происходит в соответствии с теми битами маски, которые установлены в 1. Для очистки всего слова регистра маска должна быть равна 177777, для очистки младшего байта – 377, для старшего байта – 177400.

Если КОП=2, то выполняется установка битов регистра вывода по маске, установка битов регистра вывода происходит в соответствии с теми битами маски, которые установлены в 1. Для установки всех битов регистра вывода должна быть равна 177777, для установки в 1 битов младшего байта – 377, для старшего байта – 177400.

После выполнения функция FP принимает десятичное значение считанного или записанного слова.

Пример:

```
* 1,1 X FP (2, 177777); T FP (3,10) <ВК>
8.0000 *
```

Первый оператор строки 1.1. Выполнит установку всех единиц в регистре вывода порта. Второй оператор читает бит [3] в регистре вывода порта и т.к. он был установлен в 1, то функция принимает значение 8.

9. ПРИМЕРЫ ПРОСТЫХ ПРОГРАММ НА ФОКАЛЕ

Для уменьшения длины программы и рационального использования памяти, а также для облегчения отладки и наглядности при записи выражений можно воспользоваться следующими рекомендациями:

А) Сокращать команды до первой буквы;
Б) Команды группировать в строки, отделяя команды друг от друга точкой с запятой (длина строки не должна превышать 78 знаков);
В) Оставлять незаполненными некоторые строки, чтобы при отладке программы можно было вставлять, пропущенные операторы;
Г) При записи выражений с большим количеством скобок для наглядности использовать скобки различного типа: (), [ ], < >.

Нижеследующие программы показывают практическое применение Фокала. В случае возникновения ошибок при вводе программ в память с клавиатуры микро-ЭВМ следует применять операторы Фокала, описанные в главе 6.

Для запуска на выполнение программы, находящейся в памяти компьютера, необходимо ввести оператор GOTO. Для этого достаточно нажать клавиши G и <ВК>.

В процессе выполнения программы также могут возникнуть ошибки. Для их устранения следует пользоваться рекомендациями, приведёнными в приложении 1.

9.1. ОПРЕДЕЛЕНИЕ ПАРАМЕТРОВ КРУГА, СФЕРЫ.

В данном примере для любого радиуса r вычисляются следующие параметры:

А) диаметр круга - $2R$;
Б) площадь круга - πR^2;
В) длина окружности - $2\pi R$;
Г) объём шара - $4/3\pi R^3$;
Д) площадь поверхности сферы - $4\pi R^2$.

```
C: M - FOKAL
1.01 S PI=3.141592
1.10 A "РАДИУС", R, " СМ ", !
1.20 T %8.04, !, "ВЫЧИСЛИТЬ ДЛЯ КРУГА:",!
1.25 T "ДИАМЕТР = ", 2*R, "СМ ", !
1.30 T "ПЛОЩАДЬ = ", PI*R^2, " КВ.СМ ", !
1.35 T "ДЛИНА ОКРУЖНОСТИ = ", 2*PI*R, " СМ ", !
1.40 T !, "ДЛЯ СФЕРЫ: ", !
1.45 T "ОБЪЁМ = ", (4/3)*PI*R^3, " КУБ.СМ ", !
1.50 T "ПЛОЩАДЬ ПОВЕРХНОСТИ = ", 4*PI*R^2, "КВ.СМ "
1.60 T !,!,!,!, G 1.1.
```

```
РАДИУС: 1 СМ
    ВЫЧИСЛИТЬ ДЛЯ КРУГА:
    ДИАМЕТР =                 2.0000 СМ
    ПЛОЩАДЬ =                 3.1416 КВ.СМ
    ДЛИНА ОКРУЖНОСТИ =        6.2832 СМ
    ДЛЯ СФЕРЫ:
    ОБЪЁМ =                   4.1888 КУБ.СМ
    ПЛОЩАДЬ ПОВЕРХНОСТИ =    12.5663 КВ.СМ


РАДИУС: 1.414 СМ
    ВЫЧИСЛИТЬ ДЛЯ КРУГА:
    ДИАМЕТР =                 2.8280 СМ
    ПЛОЩАДЬ =                 6.2813 КВ.СМ
    ДЛИНА ОКРУЖНОСТИ =        8.8844 СМ
    ДЛЯ СФЕРЫ:
    ОБЪЁМ =                  11.8423 КУБ.СМ
    ПЛОЩАДЬ ПОВЕРХНОСТИ =    25.1252 КВ.СМ
```

В этой программе команда ASK используется для ввода значений радиуса R с клавиатуры микро-ЭВМ. Команда SET используется для присваивания идентификатору PI значения π; эта команда записывается вне цикла вычислений.

Процесс вычисления можно прекратить в любое время, нажав клавишу "СТОП".

9.2. ОПРЕДЕЛЕНИЕ ДНЯ НЕДЕЛИ

Эта программа (вечный календарь) определяет дни недели. Исходными данными для неё являются: год, месяц, число.

```
C: M - FOKAL
1.01 A !,"ВВЕДИТЕ СЕГОДНЯШНЮЮ ДАТУ(ЧЧ/ММ/ГГГГ)",K,M,C,!;D6.5
1.10 A !,"КАКАЯ ДАТА? (ЧЧ/ММ/ГГГГ)",K,M,C,!
1.12 I (K-31) 1.15, 1.15, 1.13
1.13 I "НЕПРАВИЛЬНЫЙ ВВОД ЧИСЛА"; G 1.10
1.15 I (M-12) 1.20, 1.20, 1.13
1.20 S C=C/100; S D=FITR(.1+100*(C-FITR(C))); S C=FITR(C)
1.30 S M=M-2; I (M) 5.4, 5.4; G 5.5
5.40 S M=M+12; S D=D-1; I (-D) 5.5, 5.5; S D=99, S C=C-1
5.50 S X=FITR(FITR(2.6*M-.2)+K+D+FITR(D/4)+FITR(C/4)-2*C)
5.60 I (X-6) 5.7, 5.7; S X=X-7; G 5.6
5.70 T !, ДЕНЬ НЕДЕЛИ - "; D 6.1
5.80 I (M*1E6+K*1E4+C-Q) 5.9, 5.85, 5.9
5.85 T " - СЕГОДНЯ ! "
5.90 T !, !; G 1.1
6.10 I (X) 6.26, 6.2; I (X-2) 6.21, 6.22, 6.15
6.15 I (X-4) 6.23, 6.24; I (X-6) 6.25, 6.26
6.20 T "ВОСКРЕСЕНЬЕ"
6.21 T "ПОНЕДЕЛЬНИК"
6.22 T "ВТОРНИК"
6.23 T "СРЕДА"
6.24 T "ЧЕТВЕРГ"
6.25 T "ПЯТНИЦА"
6.26 T "СУББОТА"
6.50 D 1.2; D 1.3; S Q=M*1E6+K*1E4+C

     ВВЕДИТЕ СЕГОДНЯШНЮЮ ДАТУ (ЧЧ/ММ/ГГГГ): 16: 6: 1983

     КАКАЯ ДАТА? (ЧЧ/ММ/ГГГГ): 1:1: 1984

     ДЕНЬ НЕДЕЛИ - ВОСКРЕСЕНЬЕ

     КАКАЯ ДАТА? (ЧЧ/ММ/ГГГГ): 16: 6: 1983

     ДЕНЬ НЕДЕЛИ - ЧЕТВЕРГ - СЕГОДНЯ !
```

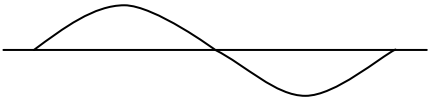
```
КАКАЯ ДАТА? (ЧЧ/ММ/ГГГГ):
```

9.3. ПРИМЕРЫ РАБОТЫ С ГРАФИКОЙ

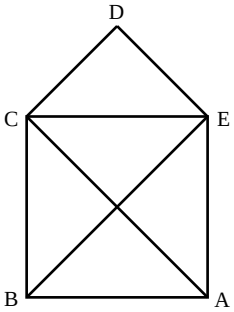
Для построения синусоиды необходимо выполнить следующую строку:

```
* 1.1 F I=0, 360; X FT(1, I, 100-50*FSIN (I*3.14159/180))
```

Координата X в функции FT изменяется в соответствии с параметром цикла I. Координата Y изменяется в соответствии с выражением (100-50*FSIN (I*3.14159/180)). Коэффициент 100 определяет расположение оси X, а 50 – амплитуду синусоиды.



Для построения фигуры ABCDE



необходимо выполнить следующую программу:

```
* 1.1 X FT (0, 300, 200)
* 1.2 X FV (1, 100, 200)
* 1.3 X FV (1, 100, 100)
* 1.4 X FV (1, 200, 200)
* 1.5 X FV (1, 300, 100)
* 1.6 X FV (1, 300, 200)
* 1.7 X FV (1, 100, 100)
* 1.8 X FV (1, 300, 100)
* 1.9 X FV (1, 100, 200)
```

Строка устанавливает точку A – с неё начинается построение фигуры.

Строка 1.2 строит сторону AB, строка 1.3 – BC, 1.4 – CD, 1.5 – DE, 1.6 – EA, 1.7 – AC, 1.8 – CE, 1.9 – EB.

9.4. РЕШЕНИЕ КВАДРАТНЫХ УРАВНЕНИЙ ТИПА $AX^2+BX+C=0$

Программа вычисляет корни квадратного уравнения $AX^2 + BX + C = 0$ по следующей формуле

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Корни уравнения зависят от величины выражения $B^2 – 4AC$:

А) Если $B^2 – 4AC > 0$ корни действительные и неравные;
Б) Если $B^2 – 4AC = 0$, корни действительные и равные;
В) Если $B^2 – 4AC < 0$, корни мнимые и неравные.

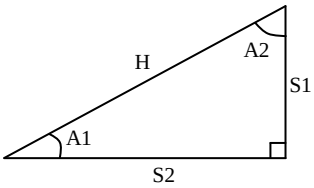
```
C: M - FOKAL
1.10 A !,!, "A", A, "B", B, "C", C; S ROOT=B^2-4*A*C
1.20 I (A) 1.4, 1.3, 1.4
1.30 T !, "ЭТО УРАВНЕНИЕ ПЕРВОЙ СТЕПЕНИ", !; G 1.1
1.40 T %6.03, !, "КОРНИ"; I (ROOT) 1.7, 1.6
1.50 T !,(-B+FSQT(ROOT))/2*A,!,(-B-FSQT(ROOT))/2*A;G1.1
1.60 T !, -B/2*A, !; G 1.1
1.70 T "МНИМЫЕ",!, -B/2*A, "+(", FSQT(-ROOT)/2*A, ")*i"
1.80 T !, -B/2*A, "-(", FSQT(-ROOT)/2*A, ")*i", !;G 1.1

     A: 3 B: 7 C: 2

     КОРНИ
          -0.333
          -2.000

     A: 3 B: 4 C: 5

     КОРНИ МНИМЫЕ
          0.667 +(1.106)*i
          0.667 -(1.106)*i
```

Оператор ASK используется для ввода с клавиатуры значений коэффициентов A, B, C.

Процесс вычисления можно прекратить в любое время, нажав клавишу "СТОП".

9.5. ОПРЕДЕЛЕНИЕ СТОРОН ПРЯМОУГОЛЬНОГО ТРЕУГОЛЬНИКА

Задана одна сторона S1 прямоугольного треугольника и прилежащий угол A2. Требуется вычислить две другие стороны треугольника и неизвестный угол.




```
C: M - FOKAL
1.10 A "СТОРОНА S1 РАВНА", S1
1.20 A "СМЕЖНЫЙ УГОЛ A2 РАВЕН", A2; T "ГРАД.", !!
1.30 S RAT=3.141592/180; S A1=90-A2
1.40 S HYP=S1/FSIN(A1*RAT); S S2=FSQT(HYP^2-S1^2)
1.50 T "СТОРОНА S2 =", S2, !, "ГИПОТЕНУЗА =", HYP,!
1.60 T "УГОЛ A1 =", A1, !

     СТОРОНА S1 РАВНА: 7
     СМЕЖНЫЙ УГОЛ A2 РАВЕН: 40. ГРАД.
     СТОРОНА S2 = 5.874
     ГИПОТЕНУЗА = 9.138
     УГОЛ A1    = 50.000
     *
```

Значения S1 и A2 вводятся с использованием оператора ASK с клавиатуры микро-ЭВМ.

9.6. РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ ОДНОРОДНЫХ УРАВНЕНИЙ

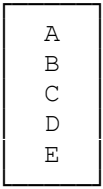
Задана система линейных однородных уравнений.

Требуется найти X1, X2, X3, удовлетворяющие этой системе:

$$1X_1 + 2X_2 + 3X_3 = 4$$
$$4X_1 + 3X_2 + 2X_3 = 1$$
$$1X_1 + 4X_2 + 3X_3 = 2$$

Решение этой системы будет приведено ниже. Так как Фокал оперирует с переменными, зависящими от одного индекса, то для обработки элементов массива требуется введение линейного индекса, т.е. такого индекса, который представляет правильный порядок расположения элементов массива в памяти. Элементы матрицы записываются в память по столбцам. Каждому элементу матрицы присваивается номер, который представляет собой линейный индекс массива.

Рассмотрим одномерный массив элементов.



Любой элемент в этом массиве представлен индексом в диапазоне 0-4 так первый элемент массива имеет нулевой индекс, элемент D – индекс 3

Теперь рассмотрим двумерный массив элементов. Массив состоит из строк и столбцов и может быть представлен в виде A(I, J). В общем виде он представляется в форме A(G), где a есть функция двух переменных I и J, таким образом, A(I, J) = A(G).

Рассмотрим диаграмму

	0	1	2
0	0	5	10
1	1	6	11
2	2	7	12
3	3	8	13
4	4	9	14

Элементы заданной матрицы имеют двойные индексы, внутри диаграммы – линейные индексы. Таким образом, каждая комбинация I и J даёт только единственное значение, т.е. для I=2 и J=1 индексом является 7.

Заметим, что для каждого постоянного I при увеличении J на 1, количество линейных индексов увеличивается на 5. Подобным образом, увеличивая I на 1 при постоянном J происходит увеличение количества линейных индексов на 1.

Для данного массива:

$I_{max} = 5$; $J_{max} = 3$

Общее число элементов массива:

$I_{max} * J_{max} = 15$

Для того, чтобы образовать таблицу, используя соответствующие индексы I и J, можно воспользоваться следующей формулой:

$G = I + I_{max} * J$

$A(G) = A(I + I_{max} * J)$ (1)

В примере решения системы линейных уравнений индексы I, I_{max} и J заменены соответственно на J, L и K, а уравнение (1) – на уравнение (2).

$A(J + L * K)$ (2)

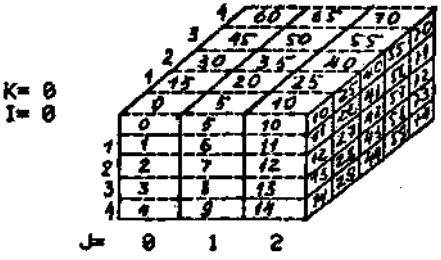
Для двумерного, случая массив представляет плоскость, для трёхмерного случая – объём и может быть представлен как куб.

Для данного куба:

$I_{max} = 5$; $J_{max} = 3$; $K_{max} = 5$

Общее число элементов массива

$I_{max} * J_{max} * K_{max} = 75$



Весь куб разбивается на малые кубы, каждый малый куб имеет свой индекс. Например, верхний левый малый куб, стоящий за первой плоскостью, имеет индекс 15.

$I = 0$; $J = 0$; $K = 1$; $G = I + (I_{max} * J) + (I_{max} * J_{max} * K) = 15$

$A(0, 0, 1) = A(15)$

Для четырёхмерного случая G определяется по формуле:

$G = I + (I_{max} * J) + I_{max} * J_{max} * K) + (I_{max} * J_{max} * K_{max} * L)$

Этот процесс может быть теоретически расширен до N – мерного случая.

Рассмотрим использование индексов на примере решения системы линейных однородных уравнений.

```
1.02 T !,"РЕШИТЬ СИСТЕМУ УРАВНЕНИЙ", !
1.04 A "ЗАДАТЬ РАЗМЕРНОСТЬ N", L, !
1.05 T !,"ЗАДАТЬ КОЭФФ. A(J,K) ... A(J,N) И B(J)", !
1.11 F K=0,N; S R(K)=K+1
1.12 F J=0,N; T !; F K=0,L; A A(J+L*K)
1.14 S M=1E-6
1.16 F J=0,N, F K=0,N; D 4
1.17 S R(P)=0
1.18 F K=0,L; S A(P+L*K)=A(P+L*K)/M
1.20 F J=0,N; D 5
1.22 S I=I+1
1.23 I (I-N) 1.14, 1.26, 1.14
1.26 F J=0,N, F K=0,N; D 7
1.28 F K=0,N; T !,%2, "X(", K+1, ")=", %8.05, X(K)
1.29 T !,!; Q
4.05 I (R(J)) 4.1, 4.3, 4.1
4.10 I (FABS(A(J+L*K))-FABS(M)) 4.3;
```

```
4.20 S M=A(J+L*K)
4.22 S P=J; S Q=K
4.30 R
5.10 I (J-P) 5.2, 5.4, 5.2
5.20 S D=A(J+L*Q)
5.30 F K=0,L; S A(J+L*K)=A(J+L*K)-A(P+L*K)*D
5.40 R
7.10 I (1E-6FABS(A(J+L*K))) 7.2; R
7.20 S X(K)=A(J+L*L)
8.1 T !, "РАЗМЕРНОСТЬ СИСТЕМЫ БОЛЬШЕ ДОПУСТИМОЙ"; R

РЕШИТЬ СИСТЕМУ УРАВНЕНИЙ
ЗАДАТЬ РАЗМЕРНОСТЬ N: 3
ЗАДАТЬ КОЭФФ. A(J,K)...A(J,N) И B(J)
: 3
: 1: 2: 3: 4
: 4: 3: 2: 1
: 1: 4: 3: 2
X (1) =    0.00000
X (2) = -  1.00000
X (3) =    2.00000
*

РЕШИТЬ СИСТЕМУ УРАВНЕНИЙ
ЗАДАТЬ РАЗМЕРНОСТЬ N: 4
ЗАДАТЬ КОЭФФ. A(J,K)...A(J,N) И B(J)
: 4

: 2: 1 : 2 : 3 : 9
: 2: 3 : -4: 5 : 13
: 1: -1: 1 : -1: -9
: 1: 1 : 2 : 1 : 7
X (1) = - 3.57142
X (2) =   4.35714
X (3) =   1.71428
X (4) =   2.78571
```

ПРИЛОЖЕНИЕ 1.

Диагностические сообщения

Код ошибки	Текстовое сообщение о причине ошибки	Рекомендуемые действия
00	ГОТОВНОСТЬ К РАБОТЕ	Система ожидает любых действий пользователя
01	НЕПРАВИЛЬНЫЙ НОМЕР СТРОКИ	Проверьте метку строки
02	НЕПРАВИЛЬНОЕ ИМЯ ФУНКЦИИ ИЛИ ПЕРЕМЕННОЙ	Проверьте имя функции или переменной (должны быть только латинские буквы, а переменная не начинается с буквы F)
03	НЕПАРНЫЕ СКОБКИ	Проверьте соответствие левых и правых скобок (каждой открывающейся скобке должна соответствовать закрывающаяся скобка того же типа)
04	НЕПРАВИЛЬНАЯ КОМАНДА	Проверьте имя и формат команды (должны быть использованы только латинские буквы)
05	НЕСУЩЕСТВУЮЩИЙ НОМЕР СТРОКИ	Проверьте наличие строки с указанным в операторе номером
06	НЕСУЩЕСТВУЮЩИЙ НОМЕР ГРУППЫ ИЛИ НОМЕР СТРОКИ В ОПЕРАТОРЕ DO	Проверьте наличие строки и группы строк, указанных в операторе DO
07	НЕПРАВИЛЬНЫЙ ФОРМАТ SET ИЛИ FOR	Проверьте формат операторов SET или FOR
08	ДВОЙНОЙ ИЛИ ОТСУТСТВУЮЩИЙ ОПЕРАТОР В ВЫРАЖЕНИИ	Проверьте арифметическое выражение, соответствие открывающихся и закрывающихся скобок
09	ПЕРЕПОЛНЕНИЕ СТЕКА	Проверьте логику программы, число рекурсий

Код ошибки	Текстовое сообщение о причине ошибки	Рекомендуемые действия
10	ПЕРЕПОЛНЕНИЕ ПАМЯТИ ТЕКСТОМ ПРОГРАММ	Сократите объём вашей программы
11	НЕТ МЕСТА ДЛЯ ПЕРЕМЕННЫХ	Сократите количество переменных или объём программы
12	ПОРЯДОК БОЛЬШЕ E+38	Проверьте значения переменных и констант вашей программы
13	ЗАПРЕЩЕННЫЙ АДРЕС ШИНЫ В ФУНКЦИИ FX	Проверьте аргумент функции. Замените адрес запрещённой ячейки памяти
14	ПОПЫТКА ДЕЛЕНИЯ НА НУЛЬ	Проверьте значения делителей в выражении
15	ПОПЫТКА ВОЗВЕДЕНИЯ В ОТРИЦАТЕЛЬНУЮ ИЛИ СЛИШКОМ БОЛЬШУЮ СТЕПЕНЬ	Проверьте значение показателя степени в выражении
16	СЛИШКОМ МНОГО СИМВОЛОВ В ВХОДНЫХ ДАННЫХ	Сократите количество символов, вводимых в ответ на запрос оператора ASK, до 23
17	КОРЕНЬ КВАДРАТНЫЙ ИЗ ОТРИЦАТЕЛЬНОГО ЧИСЛА	Проверьте значение аргумента функции FSQT
18	ПЕРЕПОЛНЕНИЕ ВХОДНОГО БУФЕРА	При вводе строки было набрано более 80-ти символов
19	ЛОГАРИФМ НУЛЯ ИЛИ ОТРИЦАТЕЛЬНОГО ЧИСЛА	Проверьте значение аргумента функций FLOG или FLOG10
20	В ФУНКЦИЯХ FASIN ИЛИ FACOS АРГУМЕНТ ПО МОДУЛЮ > 1	Проверьте значение аргумента функций FASIN или FACOS
21	ОШИБКА В ИМЕНИ ФАЙЛА	Проверьте имя файла в операторе LIBRARY
22	ОШИБКА КОНТРОЛЬНОЙ СУММЫ	Повторить чтение или запись файла

Код ошибки	Текстовое сообщение о причине ошибки	Рекомендуемые действия
23	ОШИБКА ПО ДЛИНЕ ФАЙЛА	Повторить чтение или запись файла
24	ОСТАНОВ ОПЕРАЦИЙ МАГНИТОФОНА ПО ПРЕРЫВАНИЮ ОПЕРАТОРА	Вами была нажата клавиша на клавиатуре микро-ЭВМ во время работы оператора группы LIBRARY
25	ОСТАНОВ ПО КЛАВИШЕ "СТОП"	Вами была нажата клавиша "СТОП"
26	НЕСУЩЕСТВУЮЩЕЕ УСТРОЙСТВО	Произошло обращение по несуществующему адресу общей шины. Изменить адрес в программе.
27	НЕПРАВИЛЬНЫЙ КОД ОПЕРАЦИИ В ФУНКЦИИ FP	Проверить и изменить код операции в функции FP

ПРИЛОЖЕНИЕ 2

Таблица соответствия условных обозначений текста обозначениям на клавиатуре микро-ЭВМ.

Условное обозначение	Обозначение на плёночной клавиатуре	Обозначение на кнопочной клавиатуре	Назначение
ВК	ВВОД		Ввод информации в микро-ЭВМ.
СТРЛ	←		Сдвиг курсора на одну позицию влево
СТРП	→		Сдвиг курсора на одну позицию вправо
РЗМ	|→		Размыкание
СМК	|←		Смыкание
УПС	↤-		Удаление последнего введённого символа
УСТР	СБР↦		Удаление строки с позиции курсора
СБР	СБР	СБР	Сброс (очистка) экрана
ВС	ВС	ВС	Возврат курсора в начало строки
НР	НР	АР2	Регистр
СТРВ	↑		Перемещение курсора вверх на соответствующую позицию предыдущей строки
СТРН	↓		Перемещение вниз на следующую строку
ГТ	ГТ	СУ/Т	Горизонтальная табуляция
СПВ	↗	СУ/Щ	Сдвиг курсора по стрелке в режиме редактирования "РЕД"
СПН	↘	СУ/Ч	
СЛН	↙	СУ/Ъ	
СЛВ	↖	СУ/Э	
СУГ	|↖	СУ/R	

Условное обозначение	Обозначение на плёночной клавиатуре	Обозначение на кнопочной клавиатуре	Назначение
ПОДЧ	_	Ъ	Сброс выполняемой строки

ЛИТЕРАТУРА

1. Д. Э. Роберсон. IBM5100: портативная ЭВМ на базе микропроцессора. ТИИЭР. Тематический выпуск: "Техника и применение микропроцессоров", т. 64, 1976, с.197-204.
2. Personal computer PC-8000 series. Nippon electric co., ltd.cat.no.e74-001, 8106-5003-ГМ.
3. ЭВМ "электроника 100/И". Фокал. Математическое обеспечение. И9М2.791.000 01 М018.
4. СМ ЭВМ. СМ3. СМ4. Программное обеспечение. Перфоленточная диалоговая система программирования ДС СМ. Описание языка. 1978 г.
5. Программное обеспечение СМ ЭВМ. Операционная система с разделением функций "РАФОС". Диалоговая система программирования. Описание языка, т. 5, книга 5 часть 3,1980.
6. СМ ЭВМ. СМ3. СМ4. Программное обеспечение. Перфоленточная диалоговая система программирования ДС СМ. Описание программы – интерпретатора. 1978.
7. Таненбаум Э. Многоуровневая организация ЭВМ, М., "Мир", 1979.
8. Пратт Т. Языки программирования: разработка и реализация. М., "Мир", 1979