МИНИСТЕРСТВО ПУТЕЙ СООБЩЕНИЯ СССР
МОСКОВСКИЙ ОРДЕНА ЛЕНИНА
И ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
ИНСТИТУТ ИНЖЕНЕРОВ ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Кафедра математического обеспечения
автоматизированных систем управления

ОСНОВЫ ЯЗЫКА БЭЙСИК ЭВМ ИСКРА-226

Методические указания

по дисциплине

«ПРОГРАММИРОВАНИЕ НА ЭВМ ИСКРА-226»

Часть 1

Москва — 1987

МИНИСТЕРСТВО ПУТЕЙ СООБЩЕНИЯ СССР

Московский ордена Ленина
и ордена Трудового Красного Знамени
институт инженеров железнодорожного транспорта

Кафедра математического обеспечения автоматизированных
систем управления

У т в е р ж д е н о

редакционно-издательским
советом института

ОСНОВЫ ЯЗЫКА БЭЙСИК ЭВМ ИСКРА-226

Методические указания по дисциплине
"Программирование на ЭВМ ИСКРА-226"
для слушателей факультета повышения квалификации
преподавателей и студентов

Ч а с т ь I

Методические указания составил преподаватель МИИТа:
Г.М.Курбатов

Р е ц е н з е н т ы: канд. физ.-мат. наук А.Р.Ротенберг (ВНИИНС), кафедры ПМ и ВТ ВЗИСИ.

В В Е Д Е Н И Е

Алгоритмический язык программирования БЭЙСИК предназначен для работы пользователей в режиме диалога с современными мини- и микро-ЭВМ, в частности с микро-ЭВМ "ИСКРА-226".

Название языка произошло от сокращений английских слов *Beginner's All purpose Symbolic Instzaction Code* (многоцелевой язык символических инструкций для начинающих). Язык отличается легкостью изучения, простотой записи и отладки сложных программ, доступностью устройств ввода-вывода. В настоящее время БЭЙСИК получил широкое распространение как среди начинающих, так и у профессиональных программистов. По своим выразительным возможностям его можно сравнивать с таким языком высокого уровня, как ПЛ-1.

БЭЙСИК можно эффективно использовать:

при решении инженерных, экономических и научно-технических задач;

для проведения оперативно-плановых расчетов с выдачей необходимых печатных документов;

при выполнении расчетов в системах автоматизации научных исследований;

для автоматизации конструкторских расчетов и выполнения графических построений;

для обработки данных списковой структуры и т.п.

Методическое указание состоит из двух частей.

В первой части рассматриваются основные операторы языка БЭЙСИК в той последовательности, в которой они встречаются при написании программ. Программирование на ЭВМ ИСКРА-226 осуществляется в режиме непосредственного исполнения.

Во второй части рассматриваются операторы преобразования информации, матричные и логические операции, функции символьных и логических переменных.

Для пояснения операторов и команд языка в настоящих указаниях использованы элементы формального синтаксиса: в знаки < > заключается текст, указывающий, что должно быть подставлено в данном месте; в знаки [ ] заключается текст, указывающий на необязательные компоненты операторов или команд; в фигурные скобки { } заключается список элементов, один из которых должен присутствовать в конструкции оператора.

I. АЛФАВИТ ЯЗЫКА

Алфавит языка содержит:

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

РУССКИЕ БУКВЫ А, Б, В, Г, Д, Е, Ж, З, И, Й, К, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Э, Ю, Я ;

ДЕСЯТИЧНЫЕ ЦИФРЫ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ;

ШЕСТНАДЦАТИРИЧНЫЕ ЦИФРЫ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F ;

ЧИСЛО π ;

ЗНАКИ ПРЕПИНАНИЯ "." (ТОЧКА), "," (ЗАПЯТАЯ), ";" (ТОЧКА С ЗАПЯТОЙ), ":" (ДВОЕТОЧИЕ), "'" (КАВЫЧКИ), "'" (АПОСТРОФ) ;

ЗНАКИ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ "+" (СЛОЖЕНИЕ), "-" (ВЫЧИТАНИЕ), "*" (УМНОЖЕНИЕ), "/" (ДЕЛЕНИЕ), "^" (ВОЗВЕДЕНИЕ В СТЕПЕНЬ) ;
ЗНАКИ ОПЕРАЦИЙ ОТНОШЕНИЯ ">" (БОЛЬШЕ), "<" (МЕНЬШЕ), "=" (РАВНО), ">=" (БОЛЬШЕ ИЛИ РАВНО), "<=" (МЕНЬШЕ ИЛИ РАВНО), "<>" (НЕ РАВНО) ;

СПЕЦИАЛЬНЫЕ ЗНАКИ "!", "?", "%", "#", "x", "@", "&", "\" ;

КРУГЛЫЕ СКОБКИ "(", ")" ;

КВАДРАТНЫЕ СКОБКИ "[", "]" ;

ПРОБЕЛ ;

ВОЗВРАТ КАРЕТКИ И ПЕРЕВОД СТРОКИ (CR/LF) .

2. ПРОСТЕЙШИЕ КОНСТРУКЦИИ ЯЗЫКА БЭЙСИК

К простейшим конструкциям языка БЭЙСИК относят:

константы;

переменные;

функции;

арифметические выражения.

Константы. В языке БЭЙСИК в качестве данных используются целые, вещественные и символьные константы.

Целая константа представляет собой последовательность цифр без десятичной точки, перед которой может стоять знак плюс или минус. Целая константа представляется в машине точно. Значение целой константы - любое целое число от 0 до 7999.

Примеры записи целых констант:

1, 100, +17, -275, 6375 .

Вещественная константа отличается от целой наличием десятичной точки в записи числа и представляется в машине приближенно. Различают две формы записи вещественной константы: с фиксированной точкой (естественная форма); с плавающей точкой (экспоненциальная форма).

Константа с фиксированной точкой записывается в естественной форме, т.е. целая часть числа отделяется от дробной десятичной точкой. Ноль при записи дробных чисел перед десятичной точкой может отсутствовать. Для записи чисел с фиксированной точкой машина отводит 15 десятичных разрядов: один

разряд используется для записи знака минус (в случае знака плюс ставится пробел) , I4 разрядов - для записи целой и дробной части числа и десятичной точки .

Примеры записи констант с фиксированной точкой:

I. , -I.276 , 0.725 , .I56 , 8546.324 , -I8453.95674

Константа с плавающей точкой представляется следующим образом. В естественной форме записывается мантисса числа, за которой следует символ E (признак порядка десяти) и значение порядка. При нормализованной мантиссе числа (т.е. находящейся в пределах от 0.I до I.0) порядок числа не должен превышать ± 99 .

Примеры записи констант с плавающей точкой:

8.2E8 - (число 820000000 = $8{,}2 \times 10^{8}$) ;

I564.387E-2 - (число I5,64387 = $I564{,}387 \times 10^{-2}$) ;

-.I527E-04 - (число 0,00001527 = $0{,}I527 \times 10^{-4}$) .

С и м в о л ь н а я к о н с т а н т а - это цепочка символов алфавита, заключенных в кавычки или апострофы. Максимальная длина символьной константы - цепочка из 253 символов. Значением символьной константы, например "ABC", является двоичное число 0I000001,0I000010,0I000011 , представляющее коды символов этой цепочки (выделены стрелками) .

Переменные. П е р е м е н н а я - это величина, к которой обращаются используя её имя (идентификатор), и которая в процессе вычислений может принимать различные значения. Различают простые и индексированные переменные.

Именем простой переменной может служить латинская буква или латинская буква, за которой следует цифра.

Примеры имен простых переменных:

X, S9, L1, A .

Именем индексированной переменной может также быть латинская буква или латинская буква с цифрой, за которыми в круглых скобках следует один или два индекса, разделенные запятой. Индекс может быть задан числом, переменной или арифметическим выражением и должен принимать целые положительные значения.

Примеры имен индексированных переменных:

A(I), B2(J), L(K,I), S(X-1,Y+2) .

Переменные, также как и константы, могут быть целыми, вещественными или символьными.

Признаком целой переменной является знак % после ее имени, например: A%, L1%, Z%(I), M%(I,J) .
Переменная целого типа принимает значение целого числа (константы). Отсутствие знака % после имени означает, что данная переменная вещественная. Значением вещественной переменной может быть вещественное число.

Признаком символьной переменной является знак $ после ее имени, например: A$, L$(K), Z1$(I,J) .
Символьная переменная принимает значение символьной константы.

Идентификаторы X, X%, X$, X(I) изображают четыре различные переменные и могут присутствовать в программе одновременно. Язык БЭЙСИК позволяет использовать 286 различных имен, причем в программе можно одновременно использовать до 120 разных имен.

Функции. Язык БЭЙСИК содержит:

элементарные функции;

функцию округления ROUND ;

функцию табулирования TAB .

Элементарная функция задается с помощью указателя функции (идентификатора) и аргумента функции, заключенного в круглые скобки. Идентификаторы функций, как правило, совпадают с названиями самих функций, например:

SIN, COS, TAN, ARCCOS, LOG .

Аргументом элементарной функции может быть цифровая константа, переменная, другая функция или арифметическое выражение, например:

SIN(12), TAN(X), ABS(SIN(X)), LOG(X↑2+5*X) .

Аргументы тригонометрических функций могут восприниматься в градусах или радианах (зависит от заданного оператором SELECT режима работы машины).

Элементарные функции языка БЭЙСИК приведены в табл.2.1.

Функция ROUND предназначена для округления значений с заданной степенью точности.

Форма записи функции:

ROUND(X,N) ,

где X - округляемое значение (число, переменная, арифметическое выражение);

N - выражение, определяющее уровень округления (у нецелого N дробная часть отбрасывается).

Выражение N может принимать положительное, нулевое или отрицательное значение.

ЭЛЕМЕНТАРНЫЕ ФУНКЦИИ ЯЗЫКА БЭЙСИК

ТАБЛИЦА 2.1

СТАНДАРТНАЯ ФУНКЦИЯ	ЗАПИСЬ НА БЭЙСИКЕ	ВЫПОЛНЯЕМОЕ ДЕЙСТВИЕ
$Sin\ x$	SIN(X)	ВЫЧИСЛЕНИЕ СИНУСА X
$Cos\ x$	COS(X)	ВЫЧИСЛЕНИЕ КОСИНУСА X
$tg\ x$	TAN(X)	ВЫЧИСЛЕНИЕ ТАНГЕНСА X
$arcSin\ x$	ARCSIN(X)	ВЫЧИСЛЕНИЕ АРКСИНУСА X
$arcCos\ x$	ARCCOS(X)	ВЫЧИСЛЕНИЕ АРККОСИНУСА X
$arctg\ x$	ARCTAN(X)	ВЫЧИСЛЕНИЕ АРКТАНГЕНСА X
$ln\ x$	LOG(X)	ВЫЧИСЛЕНИЕ НАТУРАЛЬНОГО ЛОГАРИФМА X
e^x	EXP(X)	ВЫЧИСЛЕНИЕ ПОКАЗАТЕЛЬНОЙ ФУНКЦИИ (ЧИСЛА e =2.718... В СТЕПЕНИ X)
$\|x\|$	ABS(X)	ВЫЧИСЛЕНИЕ МОДУЛЯ X
$\sqrt{x}$	SQR(X)	ВЫЧИСЛЕНИЕ КВАДРАТНОГО КОРНЯ ИЗ ЧИСЛА X
	INT(X)	ВЫЧИСЛЕНИЕ БЛИЖАЙШЕГО ЦЕЛОГО ЧИСЛА, НЕ ПРЕВЫШАЮЩЕГО ЗАДАННОГО ЧИСЛА X (НАПРИМЕР, INT(8.56)=8, INT(-8.58)=-9)
ДАТЧИК СЛУЧАЙНЫХ ЧИСЕЛ	RND(X)	ПОЛУЧЕНИЕ СЛУЧАЙНОГО ЧИСЛА МЕЖДУ 0 И 1 ИЗ ПОСЛЕДОВАТЕЛЬНОСТИ РАВНОМЕРНО РАСПРЕДЕЛЕННЫХ СЛУЧАЙНЫХ ЧИСЕЛ
ФУНКЦИЯ ЗНАКА	SGN(X)	ПРИСВОЕНИЕ ЗНАЧЕНИЯ: 1, ЕСЛИ X>0 , 0, ЕСЛИ X=0 , -1, ЕСЛИ X<0 .

В случае $N > 0$ X округляется до N цифр вправо от десятичной точки, например:

а)
```
10 A=ROUND(38.954,2)
20 PRINT A
 38.95
```

б)
```
10 A=ROUND(-38.955,2)
20 PRINT A
-38.96
```

Для $N = 0$ X округляется до ближайшего целого, например:

а)
```
10 A=ROUND(19.4,0)
20 PRINT A
 19
```

б)
```
10 A=ROUND(19.5,0)
20 PRINT A
 20
```

При $N < 0$ X округляется на $|N|$ цифр левее десятичной точки, например:

а)
```
10 A=ROUND(124.275,-1)
20 PRINT A
 120
```

б)
```
10 A=ROUND(124.275,-2)
20 PRINT A
 100
```

Арифметические выражения. Арифметическое выражение записывается в виде строки с использованием числовых констант, переменных, знаков арифметических операций, элементарных функций и круглых скобок.

Примеры записи арифметических выражений:

```
X+Y, X*Y+SIN(X), 1.8*LN(2-EXP(X))
```

В бесскобочном арифметическом выражении соблюдается следующий порядок выполнения операций:

вычисление значения функции;

возведение в степень;

умножение и деление;

сложение и вычитание.

Арифметические операции с одинаковыми приоритетами выполняются последовательно слева-направо. Исключение составляет операция возведения в степень, которая выполняется справа-налево. Например, по записи

L*M/ C*D**E+3*F

будет вычислено выражение

$$\frac{LM}{C^{D^{E}}} + 3F .$$

Порядок выполнения арифметических операций можно изменять с помощью соответствующей расстановки круглых скобок.

По записи

L*M/(C*D**E+3*F)

будет вычислено выражение

$$\frac{LM}{C^{D^{E}} + 3F} .$$

Изменив порядок расстановки скобок

L*M/(C*(D**E+3*F)

можно вычислить другое выражение

$$\frac{LM}{C(D^{E} + 3F)} .$$

При записи арифметического выражения количество открывающихся скобок должно соответствовать количеству закрывающихся скобок, при этом общее число символов не должно превышать 240.

В программе значение арифметического выражения присваивается переменной, стоящей слева от знака равно.

Пример:

```
W=((SIN((X+Y)↑3))↑2)/(1.26*ABS(SIN(X↑2))+EXP((X+Y)↑(1/3))).
```

По этой записи будет вычислено выражение:

$$W = \frac{Sin^2(x+y)^3}{1{,}26 \cdot |Sin\,x^2| + e^{\sqrt[3]{x+y}}} .$$

Сложные арифметические выражения рекомендуется вычислять по частям. Вычисление этого же выражения можно представить в виде:

```
L=(SIN((X+Y)↑3))↑2
L1=EXP((X+Y)↑(1/3))
W=L/(1.26*ABS(SIN(X↑2))+L1)
```

3. СТРУКТУРА ПРОГРАММЫ

Язык БЭЙСИК имеет построчную структуру [1], т.е. основной программной единицей является строка. Строки бывают двух типов: строка непосредственного счета и программная.

Строка непосредственного счета используется при ручном счете. Она состоит из одного или нескольких операторов, разделенных двоеточием,

и не имеет номера. Например:

```
X=2.7:Y=3.8:PRINT (X+Y).
```

Строка поступает в машину при нажатии пользователем на клавишу CR/LF. При этом производится проверка ее синтаксической корректности; при отсутствии ошибок сразу же выполняются действия, заданные строкой.

П р о г р а м м н а я с т р о к а используется при счете по программе и имеет свой номер строки (число от 0 до 9999) . Например:

```
20 X=2.7:Y=3.8:PRINT (X+Y).
```

При нажатии пользователем на клавишу CR/LF программная строка анализируется на синтаксическую корректность; при отсутствии ошибок записывается в память машины.

Программа, записанная на языке БЭЙСИК, предоставляет собой последовательность пронумерованных в порядке возрастания программных строк. Чтобы в дальнейшем иметь возможность вставлять в программу новые строки между уже имеющимися, строки принято нумеровать с шагом 10 (допускается также другой шаг нумерации строк) .

Строки обеих типов состоят из операторов, содержащих имя и параметры.

4. ОПЕРАТОРЫ ЯЗЫКА БЭЙСИК

Оператор определяет объекты, тип и последовательность действий над ними.

4.1 Оператор комментария REM

О п е р а т о р R E M позволяет включать в текст программы различные пояснения, что облегчает чтение программы.

Форма записи оператора следующая:

REM[<цепочка символов>].

Цепочка символов в операторе REM записывается без кавычек. Среди этих символов нельзя использовать двоеточие, т.к. знак ":" служит разделителем операторов при записи их в одной строке.

Оператор REM является неисполняемым оператором и может быть записан в любом месте программы. Программа может содержать несколько операторов REM, число которых неограничено.

Примеры использования оператора REM:

```
10 REM КУРБАТОВ Г.М., КАФЕДРА МО АСУ;
20 REM ПРОГРАММА ВЫЧЕРЧИВАНИЯ ГРАФИКА ФУНКЦИИ SIN(X).
```

4.2. Оператор присваивания LET

О п е р а т о р L E T предназначен для присваивания переменной, стоящей слева от знака равенства, конкретного численного значения, значения другой переменной или арифметического выражения, которые записаны справа от знака равенства.

Форма записи оператора следующая:

LET<цифровая переменная>= {<цифровая переменная> / <числовая константа> / <арифметическое выражение>},

LET<символьная переменная>= {<символьная константа> / <символьная переменная>}

Цифровым переменным можно присваивать цифровые значения, а символьным - символьные.

В операторе LET одно и то же значение может быть присвоено сразу нескольким переменным, стоящим слева от знака равенства. При этом переменные отделяются друг от друга запятыми. Слово LET в операторе может быть опущено.

Примеры записи оператора LET:

```
10 LET A=2.5    ;
20 LET P%=5     ;
30 Y=X↑3+SIN(X)    ;
40 L$="ABCD"  ;
50 M1,M,F=0 ;
60 A=B:L=X↑2:F1=LOG(X)+3:D(J)=A(L,K).
```

4.3. Оператор ввода INPUT

О п е р а т о р INPUT служит для ввода исходных данных с клавиатуры в процессе исполнения программы.

Форма записи оператора следующая:

INPUT [<устройство>,][<символьная константа>,]<список ввода>

Элементами списка являются цифровые и символьные переменные, разделенные запятыми.

Значения переменных вводятся пользователем после

посылки в машину команды исполнения RUN .

При выполнении оператора INPUT на дисплее индицируется знак вопроса. Пользователь при этом с клавиатуры должен задать столько данных , сколько переменных присутствует в операторе INPUT , отделяя их друг от друга запятыми, и нажать на клавишу CR/LF , например:

```
20 INPUT X,L%,A$,M(1),F(1,2)

RUN     CR/LF

 ? 1.2, 15, AKRS, 4.89, -.15        CR/LF
```

При этом переменная X получает значение 1,2; L% – значение 15; A$ – значение "AKRS"; M(1) – значение 4,89; F(1,2) – значение 0,15, после чего машина переходит к исполнению следующих операторов программы. Эти же переменные можно вводить в несколько приемов, например:

```
RUN     CR/LF
? 1.2, 15      CR/LF
? AKRS         CR/LF
? 4.89, -.15    CR/LF
```

Если на знаки вопроса данные не вводить, а нажимать каждый раз клавишу CR/LF , то программа перейдет к исполнению очередного оператора, следующего за оператором INPUT. При этом переменные, данные в которые не введены, сохраняют ранее принятые значения.

Если при вводе появится сообщение об ошибке, то ввод данных можно продолжить, начиная с ошибочного элемента.

В операторе INPUT можно записывать различные сообщения, заключив их в кавычки или апострофы.

Например:

```
INPUT "ВВЕДИТЕ ЗНАЧЕНИЕ XH=",A
INPUT "ВВЕДИТЕ ЗНАЧЕНИЕ XK=",B
INPUT "ВВЕДИТЕ ЗНАЧЕНИЕ ШАГА H=",H
```

При выполнении программы будут выданы сообщения:

```
"ВВЕДИТЕ ЗНАЧЕНИЕ XH= ?
"ВВЕДИТЕ ЗНАЧЕНИЕ XK=?
"ВВЕДИТЕ ЗНАЧЕНИЕ ШАГА H=?
```

Каждый раз после появления очередного сообщения и знака вопроса пользователь должен ввести число и нажать на клавишу CR/LF .

4.4. Оператор посимвольного ввода KEYIN

О п е р а т о р KEYIN служит для посимвольного (побайтного) ввода информации с клавишного устройства.

Форма записи оператора следующая:

KEYIN <символьная переменная>,<номер строки 1>,<номер строки 2>

Если к моменту выполнения оператора KEYIN нажата какая-либо клавиша, то информация в виде кода этой клавиши вводится и помещается в первом байте символьной переменной, после чего управление передается номеру строки 1. Если клавиша не была нажата к моменту выполнения оператора, то выполняется следующий за оператором KEYIN оператор. При нажатии одной из клавиш спецфункций (СФ) в первый байт символьной переменной помещается шестнадцатиричный код данной клавиши и управление передается номеру строки 2.

Пример использования оператора KEYIN :

```
10 DIM A$1
20 KEYIN A$,30,30:GOTO 20
30 PRINT /0C,A$;:GOTO 20

ПРОВЕРКА РАБОТЫ ОПЕРАТОРА  KEYIN .
```

При выполнении строки 20 машина ждет нажатия одной из клавиш. Когда символ введен он печатается по строке 30 и управление снова передается 20 строке. Программа позволяет печатать произвольный текст, набираемый на клавиатуре.

4.6. Оператор печати PRINT

О п е р а т о р PRINT служит для вывода цифровой и символьной информации на указанное устройство.

Форма записи оператора:

PRINT [<устройство> ,] <список вывода>.

Элементами списка могут быть :

цифровые переменные;

символьные переменные;

арифметические выражения;

символьные константы;

функция табулирования .

Элементы списка в операторе PRINT разделяются запятой или точкой с запятой.

Если разделителем списка является запятая, то печать переменных осуществляется в зонном формате, т.е. значение первой переменной печатается, начиная с I позиции, второй переменной - с 17 позиции, третьей переменной - с 33 позиции, четвертой переменной - с 49 позиции и т.д. При

строке в 80 позиций и длине зоны, равной 16 позициям, в одной строке можно отпечатать 5 значений.

Если в операторе PRINT элементы списка разделены точкой с запятой, то печать выполняется в уплотненном формате (компактная печать) . Запятую и точку с запятой можно использовать в операторе PRINT совместно (смешанный формат).

Если очередной элемент PRINT не умещается в одной строке, то он полностью печатается с начала новой строки. Наличие запятой или точки с запятой после последнего элемента списка в операторе PRINT указывает на то, что первый элемент очередного оператора PRINT будет выведен в той же самой строке, где закончил вывод предыдущий оператор PRINT. Если точка с запятой или запятая после последнего элемента списка отсутствует, то элемент следующего оператора PRINT будет выведен с новой строки. Оператор PRINT без элементов списка означает пропуск строки.

В операторе PRINT может быть указано устройство, на которое выводятся данные, например:

```
100 PRINT /0C;A:PRINT /05;B
```

По этой записи значение переменной A будет отпечатано на принтере, а переменной B – индицировано на экране дисплея.

Печать значений цифровых переменных. Числовые константы выводятся в двух формах: естественной и экспоненциальной. Если число находится в пределах от 0,1 до 10^{13}, то оно выводится в форме с фиксированной точкой, в противном случае – в экспоненциальной форме.

Пример печати в зонном формате:

```
10 A=-1.874:B=.0015:D%=-135:F=584.375
20 PRINT A,B,D%,F
-1.874          1.50000000E-03 -135          584.375
```

Пример печати в уплотненном формате:

```
10 A=-1.874:B=.0015:D%=-135:F=584.375
20 PRINT A;B;D%;F
-1.874  1.50000000E-03 -135  584.375
```

Пример печати в смешанном формате:

```
10 A=-1.8744:B=.0015:D%=-135:F=584.375
20 PRINT A;B,D%;F
-1.8744  1.50000000E-03          -135  584.375
```

Печать значений символьных переменных. Символьная константа выводится на печать в виде отдельного символа или цепочки символов, например:

```
10 LX="ALFA BEG14 ZAL1"
20 PRINT LX
ALFA BEG14 ZAL1
```

Если цепочка символов не умещается в одной строке, то она разрывается и ее печать продолжается с первой зоны новой строки.

Печать значений арифметических выражений. По оператору PRINT можно вычислить арифметическое выражение и вывести на печать его значение, например:

```
10 X=1.2
20 PRINT SIN(X)/X
 .776699238306
```

Печать символьных констант. В операторе PRINT можно записывать текстовую информацию, заключенную в кавычки, и выводить ее на печать наравне с числовыми константами, например:

```
10 X=1.2
20 PRINT "ЗНАЧЕНИЕ SIN(X)/X РАВНО";SIN(X)/X
ЗНАЧЕНИЕ SIN(X)/X РАВНО .776699238306 .
```

Функция табулирования TAB. Функция TAB используется для вывода информации на заданную позицию строки (отсчет идет от начала строки).

Форма записи функции:

TAB(<арифметическое выражение>).

Целая часть арифметического выражения определяет номер знакоместа в строке, с которого должна начинаться печать заданной информации. Результат этого выражения не должен превышать максимальной длины строки, равной 255. Если его значение больше заданной длины строки устройства, то информация выводится в начало следующей строки. Если это значение меньше номера знакоместа, на котором уже находится курсор, или если оно отрицательное, то функция TAB не выполняется.

Функция TAB используется в программах для вывода по оператору PPINT различных заголовков, элементов таблиц, точек графиков и т.п.

Примеры использования функции TAB.

```
A. 10 INPUT A,B$
   20 PRINT TAB(10);A+3;TAB(24);B$;TAB(32);"+"
                 8                CD                +
```

При A=5 и B$="CD" программа выведет в 10-й позиции строки число 8, в 24 и 25 позициях символы CD и в 32 пози-

ции - знак + .

В. 10 X=1.2
20 INPUT K
30 PRINT TAB(20+K*SIN(X)); "*" .

*

В результате выполнения программы при заданных X=1,2 и K=10 вычисляется целая часть арифметического выражения, равная 29, и в 29 позиции строки будет отпечатан символ * .

С. 100 PRINT TAB(30); "РЕЗУЛЬТАТЫ РАСЧЕТА"
110 PRINT
120 PRINT TAB(6); "ФУНКЦИЯ"; TAB(40); "АРГУМЕНТ"

РЕЗУЛЬТАТЫ РАСЧЕТА

ФУНКЦИЯ АРГУМЕНТ

В результате выполнения программы печатается: в первой строке, начиная с 30-й позиции "РЕЗУЛЬТАТЫ РАСЧЕТА"; вторая строка пропущена, т.к. в 110 строке программы оператор PRINT не содержит параметров; в третьей строке, начиная с 6 позиции, печатается слово "ФУНКЦИЯ", а с 40 позиции - слово "АРГУМЕНТ".

4.6. Оператор останова STOP

О п е р а т о р STOP служит для прерывания выполнения программы.

Форма записи оператора:

STOP [<символьная константа>] .

При счете по программе на операторе STOP вычисления прекращаются, при этом индицируется слово STOP и значение символьной константы, если она имеется. Для продолжения счета по программе необходимо нажать на клавишу CONTINUE (продолжить). Продолжение программы осуществляется с оператора, следующего за оператором STOP. Текст программы может содержать несколько операторов STOP .

Примеры:

а) `150 STOP ;`

б) `380 STOP "ДЛЯ ПРОДОЛЖЕНИЯ РАБОТЫ НАЖМИТЕ CONTINUE"`

4.7. Оператор конца END

О п е р а т о р END обозначает конец записи текста программы и является последним оператором программы. Его максимально возможный номер – 9999.

Пример записи оператора:

```
9999 END .
```

После выполнения оператора END счет по программе прекращается и на экране высвечивается сообщение:

```
END PROGRAMM
FREE SPACE=XXXXX  ,
```

где: xxxxx – количество байтов свободной памяти.

Оператор END в программе не является обязательным.

4.8. Оператор безусловного перехода GOTO

О п е р а т о р GOTO позволяет передавать управление любому исполняемому оператору программы без предварительной проверки каких либо условий.

Форма записи оператора:

```
GOTO <номер строки>  ,
```

где: номер строки – строка программы, которой передается управление. Строка, к которой осуществляется переход, может находиться до или после оператора передачи управления.

например:

```
10 INPUT A,B
20 C=A+B
30 GOTO 50
40 STOP
50 D=A-B
60 PRINT C,D
70 GOTO 40
```

При выполнении программы после вычисления суммы C=A+B в 30 строке осуществляется передача управления 50 строке. После вычисления разности D=B-A производится печать вычисленных значений C и D и управление передается 40 строке программы, где оператор STOP заканчивает программу.

4.9. Оператор условного перехода IF-THEN

О п е р а т о р IF-THEN предназначен для организации перехода к указанной строке программы при выполнении заданного условия.

Форма записи оператора:

IF<параметр><операция сравнения><параметр>
THEN<номер строки> ,

где: IF - служебное слово "ЕСЛИ" ;

THEN - служебное слово "ТОГДА" ;

<параметр> - константа, простая переменная, индексированная переменная, арифметическое выражение;

<операция сравнения> - =, >, <, >=, <=, <>;

<номер строки> - строка программы, которой передается управление в случае выполнения условия. Если условие не выполняется, управление передается следующему за IF-THEN оператору программы.

Пример:

```
10 INPUT A,B
20 PRINT A;B
30 IF A>BTHEN50
40 PRINT "B=";B:GOTO 60
50 PRINT "A=";A
60 STOP
```

При вводе чисел 1 и 2 (B>A) выбирается и печатается значение B=2, при другой последовательности ввода чисел(2 и 1, A>B) печатается значение A=2.

4.10. Оператор вычисляемого перехода ON

О п е р а т о р **ON** предназначен для организации нескольких ответвлений из одной точки программы.

Этот оператор имеет вид:

ON E GOTO $C_1, C_2, C_3, \ldots, C_m$,

где: *ON* - служебное слово;

E - арифметическое выражение;

$C_1, C_2, C_3, \ldots, C_m$ - список номеров строк программы, на которые возможен переход.

При выполнении оператора **ON** вычисляется целая часть K арифметического выражения E . Если K=I , то управление передается оператору с номером C_1, если K=2 - оператору C_2 и т.д. Если K меньше единицы или больше числа m , то выполняется следующий за оператором **ON** оператор программы.

Пример:

```
10 INPUT X
20 ON X+.5GOTO40,50,60,70
30 Y=SQR(X):GOTO 80
40 Y=X↑2:GOTO 80
50 Y=X-1:GOTO 80
60 Y=X+1:GOTO 80
70 Y=X↑3:GOTO 80
80 PRINT "X=";X,"Y=";Y
```

При выполнении программы управление передается:

```
X= 1          Y= 1            40 строке, K=I ;
X= 2          Y= 1            50 строке, K=2 ;
X= 0          Y= 0            30 строке, K<I ;
X= 3          Y= 4            60 строке, K=3 ;
X= 4          Y= 64           70 строке, K=4 ;
X= 5          Y= 2.236067977  30 строке, K>4 ;
X= 6          Y= 2.449489742783 30 строке, K>4 .
```

4.11. Оператор перехода по ошибке ON ERROR

О п е р а т о р **ON ERROR** служит для обработки ошибки, встречающейся в процессе исполнения программы.

Форма записи оператора:

ON ERROR <символьная переменная 1>,<символьная переменная 2>**GOTO**<номер строки> ,

где: <символьная переменная 1> - переменная, которой присваивается код ошибки;

<символьная переменная 2> - переменная, которой присваивается номер строки, где обнаружена ошибка;

<номер строки> - строка, которой передается управление в случае обнаружения ошибки.

Оператор **ON ERROR** записывается перед строкой программы, где может возникать ошибка (обычно в начале программы). Если ошибка встречается внутри цикла или подпрограммы, то возврат в цикл или подпрограмму невозможен (по операторам **NEXT** или **RETURN** машина выдает сообщение об ошибке). Присутствие в программе оператора **ON ERROR** без параметров отменяет программную обработку ошибки.

Пример:

```
10 ON ERROR A%,B%GOTO60
20 X=0
30 Y=SQR(.5+SIN(X))
40 PRINT X,Y
50 GOTO 70
60 PRINT A%,B%
70 X=X+1.2
80 IF X<=8.5THEN30
 0                .707106781865
 1.2              1.19667835526B
 2.4              1.08418779764
 3.6              .239748945159
03                0030
 6                .469664243690Z
 7.2              1.13739521005Z
 8.4              1.16387224762B3
```

При возникновении ошибки в программе (извлечение корня из отрицательного числа) управление передается 60-й строке, при этом печатается код ошибки (03), номер строки (30), где обнаружена ошибка, и счет по программе продолжается.

4.12. Операторы цикла FOR-NEXT

О п е р а т о р ы FOR-NEXT служат для организации циклов в программе. В циклических программах различают заголовок цикла, тело цикла и конец цикла.

Заголовок цикла имеет вид:

FOR <цифровая переменная> = <выражение 1> TO <выражение 2> [STEP <выражение 3>],

где: FOR - служебное слово "ДЛЯ";

<цифровая переменная> - параметр цикла;

<выражение 1> - задает начальное значение параметра цикла;

<выражение 2> - задает конечное значение параметра цикла;

<выражение 3> - задает шаг изменения параметра цикла;

TO - служебное слово "ДО";

STEP - служебное слово "ШАГ". При отсутствии в заголовке цикла слова STEP шаг принимается равным 1.

Тело цикла - это группа операторов, многократно участвующих в вычислениях в процессе повторения цикла. Тело цикла может содержать группу операторов, образующих свой внутренний цикл.

Конец цикла задается оператором NEXT:

NEXT <цифровая переменная>,

где: <цифровая переменная> - параметр цикла, указанный в заголовке цикла.

При выполнении оператора NEXT текущее значение параметра цикла изменяется на величину шага и проверяется условие выхода из цикла: текущее значение параметра цикла сравнивается с его конечным значением на знак больше, если величи-

на шага больше 0; текущее значение параметра цикла сравнивается с его конечным значением на знак меньше, если величина шага меньше нуля.

Если условие выхода из цикла выполняется, то управление передается следующему за оператором NEXT оператору программы. При невыполнении условия управление передается в начало цикла - следующему за оператором FOR оператору программы.

Пример программы с простым циклом:

```
10 FOR X 1TO6STEP2
20 Y=X↑2+1:PRINT X,Y
30 NEXT X
 1                2
 3                10
 5                26
```

Программа вычисляет значение $Y = X^2 + 1$ при изменении параметра X от 1 до 6 с шагом 2.

Пример программы со сложным вложенным циклом:

```
10 FOR X=1TO2
20 PRINT "X=";X
30 PRINT
40 FOR Y=1TO3
50 Z=X↑2+.5*Y
60 PRINT "Y=";Y,"Z=";Z
70 NEXT Y
80 PRINT
90 NEXT X

X= 1

Y= 1             Z= 1.5
Y= 2             Z= 2
Y= 3             Z= 2.5

X= 2

Y= 1             Z= 4.5
Y= 2             Z= 5
Y= 3             Z= 5.5
```

Программа содержит внешний цикл по X, который включает в себя другой внутренний цикл по Y.

4.13. Оператор объявления размерности DIM

О п е р а т о р DIM резервирует место в памяти машины для массивов, а также задает длины символьных переменных и элементов символьных массивов.

Форма записи оператора:

DIM <список элементов> .

Элементами списка могут быть: имена цифровых массивов, имена символьных массивов, имена символьных переменных, за которыми в круглых скобках указана их максимальная размерность. Максимальная размерность специально не ограничивается. Элементы списка отделяются друг от друга запятыми.

Оператор DIM записывается в программе до первого обращения к переменным или массивам. В одном операторе DIM можно объявлять размерности нескольких массивов и символьных переменных, например:

```
30 DIM L(100),B%(3,5),C$(24)20 .
```

Оператором DIM резервируется 100 ячеек памяти для ввода элементов вектора L , 15 ячеек (3 строки х 5 столбцов) для элементов матрицы B% и 24 ячейки для символьных переменных C$ (длина каждого значения - 20 символов) .

В случае, если оператор DIM в программе отсутствует, то по "умолчанию" для вектора резервируется 100 ячеек памяти, для матрицы - 10 х 10 ячеек памяти, для символьного массива - 100 ячеек памяти с максимальной длиной символьной переменной, равной 16 символам.

4.14. Оператор объявления общей области памяти COM

О п е р а т о р COM служит для объявления переменных и массивов общими для нескольких программ или сегментов программы, резервирования для них места в памяти машины и задания длины символьных переменных и элементов символьных массивов.

Форма записи оператора:

COM <список элементов> .

Переменные, объявленные оператором COM, являются общими для всех сегментов программы. Их значения не стираются при вводе этих сегментов в машину (в отличии от оператора *DIM*) . Стереть их можно только с помощью оператора *CLEAR* .

Длина общих переменных и размерности массивов должны быть одинаковы во всех сегментах программы, объявленных общими.

Оператор COM должен быть записан в программе перед первым оператором *DIM* .

Пример:
```
20 COM L(3,5),S(100),B$(3,7)6
30 DIM D(10,10),M(60),X,Y .
```

Общими в программе объявлены: цифровая матрица *L* размерностью 3x5 элементов, цифровой вектор *S* размерностью 100 элементов и символьная матрица *B$* размерностью 3x7 элементов с длиной каждого элемента, равного 6 символам.

4.15. Оператор блока данных DATA

О п е р а т о р DATA предназначен для хранения числовых или символьных констант, используемых при вычислениях, в тексте программы.

Форма записи оператора:

DATA «список констант».

С помощью оператора DATA организуют блоки данных, используемых в дальнейшем оператором READ для ввода числовых и символьных констант в зарезервированные для них ячейки памяти. При этом числовые константы в операторе DATA должны соответствовать цифровым переменным в операторе READ , а символьные константы - символьным переменным. Если константы не умещаются в одном блоке данных, то для их записи можно использовать несколько блоков данных.

Оператор DATA является неисполняемым оператором и может быть записан в любом месте программы.

Пример:
```
30 DATA 22.7,-6.8,.5,14.01,1E2
40 DATA 3,5,8,17,24,36
50 DATA "AB","CD","LE","GH","RS"
```

4.16. Оператор чтения данных READ

О п е р а т о р READ организует последовательную выборку констант из блоков DATA и присваивает их значения переменным, указанным в операторе READ .

Форма записи оператора:

READ «список вводимых переменных» .

Цифровые переменные в операторе READ должны соответствовать числовым константам в операторе DATA , а

символьные переменные - символьным константам.

Если оператор READ содержит меньше переменных, чем задано констант в блоке DATA , то избыточные значения игнорируются.

Если оператор READ содержит больше переменных, чем имеется констант в блоке DATA , то выдается сообщение об ошибке.

Пример:

```
 10 DIM A(2,2),B(3),C$(4)
20 FOR I=1TO2
30 FOR J=1TO2
40 READ A(I,J):PRINT A(I,J);
50 NEXT J:PRINT :NEXT I
60 FOR I=1TO3
70 READ B(I):PRINT B(I):NEXT I
80 FOR I=1TO4
90 READ C$(I):PRINT C$(I):NEXT I
100 DATA 1,2,-2,3
110 DATA 8,5,4
120 DATA "AL","EF","CD","LN"
```

При выполнении программы оператор READ последовательно выбирает из блоков DATA значения числовых и символьных констант и присваивает их значения элементам матрицы A(2,2) , вектора B(3) и символьному вектору C$(4) ; на печать выводятся:

```
1  2
-2 3      } элементы матрицы A(2,2);

8
5         } элементы вектора B(3);
4

AL
EF
CD        } элементы символьного вектора C$(4)2 .
LN
```

4.17. Оператор восстановления данных RESTORE

О п е р а т о р RESTORE позволяет операторам *READ* считывать данные из блоков *DATA* любое число раз, начиная с указанного данного.

Форма записи оператора:

RESTORE [<арифметическое выражение>] .

Целая часть арифметического выражения указывает на номер данного в блоке DATA, с которого необходимо выполнить повторное считывание. Если в операторе RESTORE параметр <арифметическое выражение> отсутствует, то повторное считывание будет осуществляться с первого элемента.

Пример:

```
10 DIM A(6),B(5),C(2,3)
20 DATA 1,2,3,4,5,6,7,8,9,10
30 FOR I=1TO6
40 READ A(I)
50 PRINT A(I);:NEXT I
55 PRINT
60 RESTORE
70 FOR I=1TO5
80 READ B(I)
90 PRINT B(I);:NEXT I
95 PRINT
100 RESTORE 4
110 FOR K=1TO2
120 FOR J=1TO3
130 READ C(K,J)
140 PRINT C(K,J);
150 NEXT J:PRINT :NEXT K
```

При выполнении программы данные из блока DATA читаются три раза. Считанные значения присваиваются элементам цифровых массивов A(6) , B(5), C(2,3) и выводятся на печать:

```
 1  2  3  4  5  6 - A(6);
 1  2  3  4  5  -   B(5);
 4  5  6 }
 7  8  9 }  - C(2,3).
```

4.18. Оператор объявления функции пользователя DEFFN

О п е р а т о р DEFFN служит для записи многократно встречающихся в программе одних и тех же арифметических выражений в виде оператора-функции и последующих обращений к ней по имени из различных точек программы.

Форма записи оператора:

DEFFN <имя функции>(<формальный аргумент>) =
<арифметическое выражение> ,

где: <имя функции> - цифра или латинская буква, следующая за FN ;

<формальный аргумент> - простая цифровая переменная.

Арифметическое выражение может включать в себя функции, вызывающие другие функции. Обращение к оператору-функции осуществляется с указанием ее имени и следующего за ним в круглых скобках значением фактического аргумента:

FN <имя функции>(<фактический аргумент>) .

где: <фактический аргумент> - числовая константа, цифровая переменная, арифметическое выражение.

При обращении формальный аргумент оператора-функции подменяется фактическим аргументом, происходит вычисление функции с этим аргументом и результат вычисления возвращается в точку программы, откуда было произведено обращение.

Оператор-функцию можно записывать в любом месте программы. Программа может иметь несколько различных операторов-функций.

Пример. Составить программу для вычисления арифметического выражения

$$y = th(x^2 - 1) + \sin x \;;$$
$$F = \sqrt{1{,}05x + |th(2{,}35 + x)|} \;;$$
$$M = th(\cos x + \sin x \cdot th\, x) \;,$$ где

$$th(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}} \;.$$

Вычисление $th(z)$ оформить в виде оператора-функции.

Программа вычислений:

```
10 INPUT X
20 DEFFN T(Z)=(1-EXP(-2*Z))/(1+EXP(-2*Z))
30 Y=FNT(X^2-1)+SIN(X)
40 F=SQR(1.05*X+ABS(FNT(2.35+X)))
50 M=FNT(COS(X)+SIN(X)-FNT(X))
60 PRINT X;Y;F;M

 1.  .8414709848079  1.430923199461  .5512527401882
```

Оператор-функция с именем *FNT* и формальным аргументом z записана в 20 строке программы. Обращение к функции осуществляется по имени *FNT* из 30, 40 и дважды из 50 строк программы. При первом обращении к функции формальный аргумент z подменяется фактическим аргументом $(X^2 - 1)$, при втором и последующих обращениях формальный аргумент z подменяется соответственно фактическими аргументами $(2{,}35 + X)$, X, $(COS(X) + SIN(X) - FNT(X))$. Каждый раз происходит вычисление th с заданным фактическим аргументом и результаты возвращаются в соответствующие формулы.

4.19. Оператор обращения к подпрограмме gosub

О п е р а т о р gosub предназначен для входа в подпрограмму. Подпрограмма - это специальным образом оформленная группа операторов, к которым можно обращаться из разных точек программы.

Форма записи оператора:

gosub <номер строки>

где: <номер строки> - номер строки подпрограммы, которому передается управление.

По оператору gosub можно входить в подпрограмму не обязательно через ее заголовок.

Оператор gosub может находиться внутри одной подпрограммы и вызывать другую подпрограмму. Подпрограмму можно размещать в любом месте исходной программы. Последним оператором подпрограммы должен быть оператор RETURN, который служит для возвращения к основной программе (к оператору, следующему за оператором gosub).

При записи программы с подпрограммами необходимо предусматривать пересылку в них исходных аргументов, а после выхода из подпрограмм сохранять результаты вычислений.

Пример. Составить программу вычисления числа сочетаний из N по M:

$$C_N^M = \frac{N!}{M!(N-M)!} .$$

Вычисление факториала оформить в виде самостоятельной подпрограммы.

Программа вычисления C_N^M :

```
10 INPUT N,M
20 L=N:GOSUB 70:P=A
30 L=M:GOSUB 70:P1=A
40 L=N-M:GOSUB 70:P2=A
50 C=P/(P1*P2):PRINT C
60 STOP
70 REM "ПОДПРОГРАММА GOSUB "
80 A=1:FOR K=2TOL
90 A=A*K:NEXT K
100 RETURN
 10
```

Подпрограмма вычисления факториала записана в 70, 80 и 90 строках программы. Обращение к подпрограмме производится из 20, 30 и 40 строк основной программы по оператору GOSUB . Предварительно переменной L , факториал которой вычисляется в подпрограмме, присваиваются значения N, M, $N-M$. Результаты вычисления факториала запоминаются в ячейках P, P1 и P2. При введенных значениях $N = 5$ и $M = 3$ вычисленное значение C_N^M равно 10.

4.20. Операторы выхода из подпрограммы

RETURN и RETURN CLEAR

О п е р а т о р RETURN используется для указания конца подпрограммы и передачи управления оператору, следующему за оператором GOSUB или GOSUB .

О п е р а т о р RETURN CLEAR служит для стирания в памяти машины адреса возврата к основной программе. После испол-

нения программы реализуется оператор, следующий за оператором RETURN .

4.21. Оператор объявления помеченной подпрограммы DEFFN '

О п е р а т о р DEFFN ' имеет две формы.

Форма I оператора DEFFN ' предназначена для организации помеченных подпрограмм и используется в программах совместно с оператором GOSUB '.

Форма записи оператора:

DEFFN ' <номер подпрограммы>[(<формальные аргументы>)],

где:<номер подпрограммы>– целое число от 0 до 255;

<формальные аргументы> – список переменных.

Форма 2 оператора DEFFN ' служит для ввода часто встречающихся текстов посредством нажатия одного из функциональных ключей.

Форма записи оператора:

DEFFN '<номер специальной функции> <символьная константа> ,

где: <номер специальной функции> – целое число от 0 до 31;

<символьная константа> – цепочка символов алфавита, заключенная в кавычки или апострофы.

Пример:

```
10 DIM A(3,2),B(2,3),C(3,3)
20 DEFFN '3"MAT"
30 MAT INPUT A
40 MAT INPUT B
50 MAT C=A*B
60 MAT PRINT C
```

В 20 строке программы с помощью оператора DEFFN ' записана символьная константа "MAT", которая закреплена за 3 функциональным ключом. При дальнейшем наборе текста программы можно каждый раз не набирать по буквам слово "MAT", а вызывать его ввод нажатием на 3-ю функциональную клавишу. Это ускоряет набор текста программы.

4.22. Оператор обращения к помеченной подпрограмме GOSUB '

О п е р а т о р GOSUB ' предназначен для входа в помеченную подпрограмму, определенную по оператору DEFFN ' .

Форма записи оператора:

GOSUB ' <номер подпрограммы>(<фактические аргументы>) ,

где: <номер подпрограммы> - номер, указанный в операторе DEFFN ' ;

<фактические аргументы> - символьные константы, символьные переменные, арифметические выражения.

Список фактических аргументов в операторе GOSUB ' должен соответствовать по количеству и типу списку формальных аргументов в операторе DEFFN ' .

Вход в помеченную подпрограмму по обращению GOSUB ' возможен только через ее заголовок, содержащий оператор DEFFN '

Рассмотрим программу вычисления числа сочетаний из N по M (см.стр.19) с использованием операторов DEFFN ' и GOSUB ' .

```
10 INPUT N,M
20 GOSUB ' 15(N):P=A
30 GOSUB ' 15(M):P1=A
40 GOSUB ' 15(N-M):P2=A
50 C=P/(P1*P2):PRINT C
60 STOP
70 DEFFN ' 15(L)
80 A=1:FOR K=2TOL
90 A=A*K:NEXT K
100 RETURN
 10
```

Вычисление факториала в программе оформлено в виде самостоятельной помеченной подпрограммы DEFFN '(70-я строка) с номером I5 и формальным аргументом (L) . Обращение к помеченной подпрограмме производится по оператору GOSUB ' из 20, 30 и 40 строк основной программы с указанием номера подпрограммы I5 и фактических аргументов $N, M, N-M$. Результаты вычисления факториала запоминаются в ячейках P , PI и P2 . При заданных значениях N = 5 и M =3 вычисленное значение C_N^M равно I0.

4.23. Оператор форматной печати PRINTUSING

О п е р а т о р PRINTUSING предназначен для вывода цифровой и символьной информации в заданном пользователем формате.

Форма записи оператора:

PRINTUSING <устройство> , <формат> , <список выводимых переменных> .

Формат вывода данных может быть задан одним из

трех способов:

цепочкой символов, заключенных в кавычки;

символьной переменной;

оператором IMAGE .

Объявление формата в виде цепочки символов.

Форма записи оператора:

PRINTUSING <"формат"> ,<список выводимых переменных> .

При выводе цифровой информации используются следующие описания форматов:

для целых чисел, например ### ;

для вещественных чисел с фиксированной точкой, например ##.### ;

для вещественных чисел с плавающей точкой, например #.##^^^^ .

В процессе вывода по оператору PRINTUSING цифровой информации каждое значение выводимого списка печатается в соответствии с заданным форматом. При этом выполняются следующие правила:

если в начале формата не указан знак числа, а число является отрицательным, то при печати индицируется знак минус и число занимает на одну позицию больше, чем указано в формате;

если в формате обозначен знак плюс, то всегда индицируется знак числа, как положительного, так и отрицательного;

если в формате указан знак минус, то знак индицируется только в случае отрицательного числа.

Пример:

```
10 A=25:B=-25
20 PRINTUSING "##",A;B;
30 PRINTUSING "+##",A;B;
40 PRINTUSING "-##",A;B

25-25+25-25 25-25
```

При выводе цифровых значений по формату целых чисел соблюдаются правила:

если число содержит целую и дробную часть, то целая часть числа индицируется, а дробная - отбрасывается;

если число короче заданного формата, то свободные позиции впереди числа заполняются пробелами;

если число больше формата, то вместо числа индицируется формат.

Пример:

```
10 L=23.63:M=8:N=133
20 PRINTUSING "##",L,M,N
23
 8
##
```

При выводе цифровой информации по формату чисел с фиксированной точкой выполняются следующие правила:

если целая часть числа меньше формата, то при выводе числа впереди ставятся пробелы;

если целая часть числа больше формата, то печатается формат;

если дробная часть числа меньше формата, то оставшиеся позиции дополняются нулями;

если дробная часть числа больше формата, то лишние разряды дробной части отбрасываются.

Пример:

```
10 A=1.27:B=127.65:C=12.7:D=25.6824
20 PRINTUSING "##.##",A,B,C,D
 1.27
##.##
12.70
25.68
```

При выводе цифровой информации по формату чисел с плавающей точкой выполняется правило:

число приводится к показательной форме и индицируется по заданному формату, если показатель степени не превышает 99 (в противном случае индицируется формат)

Пример:

```
10 A=375.874:B=.0015:C=65.4E-3:D=1301
20 PRINTUSING "#.#####",A,B,C,D
3.75E+02
1.50E-03
6.54E-02
1.30E+03
```

При индикации символьной информации каждый знак # в формате заменяется на соответствующий символ. Если значение символьной константы короче формата, то при печати она дополняется пробелами. Если значение символьной константы длиннее формата, то лишние символы отбрасываются, например:

```
10 A$="ABC":B$="DEFLM":C$="ROMBINOS"
20 PRINTUSING "######",A$,B$,C$
ABC
DEFLM
ROMBI
```

Обращение формата в символьной переменной.

Форма записи:

PRINTUSING «символьная переменная»,«список выводимых переменных» .

В одной символьной переменной можно задавать несколько различных форматов, например:

```
10 DIM A$60
20 B$="ALFA":B=-375:C=12.247:D=.0015
30 A$="####  ###  ##.###  #.#^^^^"
40 PRINTUSING A$,B$,B,C,D
ALFA  -375  12.247  1.5E-03
```

При объявлении форматов в символьной переменной длина символьной переменной (если форматы занимают более 16 символов) должна быть также объявлена.

Если количество описаний форматов в символьной переменной меньше, чем число переменных, то форматы используются повторно до тех пор, пока все переменные не будут отпечатаны,например:

```
10 DIM A$60
20 B$="ALFA":B=-1.2:C=12:D=-16.4
30 A$="####  ##.# "
40 PRINTUSING A$,B$,B,C,D
ALFA  - 1.2
  12  -16.4
```

Если количество описаний форматов в символьной переменной больше, чем число переменных, то избыточные форматы игнорируются, например:

```
10 DIM A$60
20 B$="ALFA":B=-375
30 A$="####  ###  ##.###  #.#^^^^"
40 PRINTUSING A$,B$,B
ALFA  -375
```

Символьная переменная кроме описаний форматов может содержать текстовую информацию:

```
10 DIM A$60
20 G=1938:M$="ФЕВРАЛЬ":N=24
30 A$="ГОД ####  МЕСЯЦ ##########  ЧИСЛО ##"
40 PRINTUSING A$,G,M$,N

ГОД 1938  МЕСЯЦ ФЕВРАЛЬ  ЧИСЛО 24
```

Объявление формата в операторе IMAGE

Форма записи:

PRINTUSING <номер строки> , <список выводимых переменных> ,

где: <номер строки> - строка программы с оператором IMAGE.

Пример:

```
10 B$="ALFA":B=-375:C=12.274:D=.0015
20 PRINTUSING 30,B$,B,C,D
30 % ####  ###  ##.###  #.##^^^^
 ALFA -375  12.274  1.5E-03
```

Оператор IMAGE изображается символом % и используется совместно с оператором PRINTUSING , задавая ему форматы печати числовой и символьной информации.

Форма записи оператора:

% <цепочка символов> .

Оператор IMAGE может включать в себя текст, предназначенный для печати, и форматы, в соответствии с которыми осуществляется печать переменных, перечисленных в операторе PRINTUSING , например:

```
10 G=1930:M$="ФЕВРАЛЬ":N=24
20 PRINTUSING 30,G,M$,N
30 % ГОД ####  МЕСЯЦ #########  ЧИСЛО ##

 ГОД 1930  МЕСЯЦ ФЕВРАЛЬ  ЧИСЛО 24
```

4.24. Оператор печати кодов символов HEXPRINT

О п е р а т о р HEXPRINT предназначен для вывода символьной информации в шестнадцатиричном (HEX) коде.

Форма записи оператора:

HEXPRINT «устройство» «список символьных переменных».

Наличие запятой в операторе HEXPRINT вызывает печать следующего символа с начала новой строки.

Пример:

```
5 DIM A$*1,B$*1,C$*2
10 A$="L":B$="M":C$="LM"
20 PRINT A$,B$,C$
25 PRINT
30 HEXPRINT A$,B$,C$

L               M               LM

4C
4D
4C4D
```

Точка с запятой между элементами HEXPRINT вызывает печать символов в уплотненном формате:

```
5 DIM A$*1,B$*1,C$*2
10 A$="L":B$="M":C$="LM"
20 PRINT A$,B$,C$
25 PRINT
30 HEXPRINT A$;B$;C$
L               M               LM

4C4D4C4D
```

Символы, не поместившиеся в одной строке, печатаются с начала новой строки.

Основные операторы языка БЕЙСИК даны в приложении

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

1. Машина вычислительная электронная клавишная программуправляемая "ИСКРА-226". Инструкция по программированию. Базовый объем. I.320.136 Д14-1.-Курск.: э-д Счетмаш, 1983, ч.1, 135 с.

2. Кетков Ю.Л. Программирование на БЭЙСИКЕ.-М.: Статистика, 1979, 159 с.

3. Уорт Т. Программирование на языке БЭЙСИК.-М.:Машиностроение, 1981, 255 с.

4. Трэктон К. Программы на БЭЙСИКЕ для инженерно-технических расчетов.-М.: Радио и связь, 1985, 96 с.

5. Чагинаев В.П., Чернухин С.И., Шахунянц Т.Г. Программирование на алгоритмическом языке БЭЙСИК ЭВМ "ИСКРА-226". Методические указания к практическим и лабораторным работам.-М.: МИИТ, 1985, 52 с.

ПРИЛОЖЕНИЕ

ОСНОВНЫЕ ОПЕРАТОРЫ ЯЗЫКА БЭЙСИК

ТАБЛИЦА

NN П/П	ИМЯ ОПЕРАТОРА	НАЗНАЧЕНИЕ ОПЕРАТОРА	ПРИМЕР ЗАПИСИ ОПЕРАТОРА
1	REM	КОММЕНТАРИЙ	10 REM НАЧАЛО РАСЧЕТА
2	LET	ПРИСВАИВАНИЯ	30 LET A=1.2:B%=2 40 C=X+Z 2:D=2.8E-4
3	INPUT	ВВОД ДАННЫХ	60 INPUT A,B%,C$
4	PRINT	ПЕЧАТЬ ДАННЫХ	50 PRINT A;B%;C$ 55 PRINT A;B%;C$
5	STOP	ОСТАНОВ	90 STOP 100 STOP"2"
6	END	КОНЕЦ	9999 END
7	GOTO	БЕЗУСЛОВНЫЙ ПЕРЕХОД	80 GOTO 100
8	IF-THEN	УСЛОВНЫЙ ПЕРЕХОД	60 IF A>B THEN 300
9	ON	ВЫЧИСЛЯЕМЫЙ ПЕРЕХОД	40 ON K GOTO 90,20, 60,110
10	ON ERROR	ПЕРЕХОД ПО ОШИБКЕ	80 ON ERROR A$,B$ GOTO 300
11	FOR-TO	ЗАГОЛОВОК ЦИКЛА	30 FOR X=1.2 TO 9.6 STEP 0.4 35 FOR L=1 TO 16
12	NEXT	КОНЕЦ ЦИКЛА	120 NEXT X
13	DIM	ОБЬЯВЛЕНИЕ РАЗМЕРНОСТИ	20 DIM A(20),B(5,6), C$(10)12
14	COM	ОБЬЯВЛЕНИЕ ОБЩЕЙ ОБЛАСТИ ПАМЯТИ	30 COM B(10),L(2,4)
15	DATA	БЛОК ДАННЫХ	10 DATA 1,2,3,4,5,6 20 DATA "A","B","C"

16	READ	ЧТЕНИЕ ДАННЫХ ИЗ БЛОКА DATA	30 READ X(I),R(K) 40 READ L(K,J)
17	RESTORE	ВОССТАНОВЛЕНИЕ ДАННЫХ	100 RESTORE
18	DEF FNT	ОБЪЯВЛЕНИЕ ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ	70 DEF FNT(X)=SIN(X)/X
19	GOSUB	ОБРАЩЕНИЕ К ПОДПРОГРАММЕ	150 GOSUB 200
20	RETURN	ВОЗВРАТ ИЗ ПОДПРОГРАММЫ К ОСНОВНОЙ ПРОГРАММЕ	360 RETURN
21	RETURN CLEAR	ВЫХОД ИЗ ПОДПРОГРАММЫ СО СТИРАНИЕМ АДРЕСА ВОЗВРАТА	240 RETURN CLEAR
22	DEFFN '	ОБЪЯВЛЕНИЕ ПОМЕЧЕННОЙ ПОДПРОГРАММЫ	100 DEFFN' 15(L)
23	GOSUB '	ОБРАЩЕНИЕ К ПОМЕЧЕННОЙ ПОДПРОГРАММЕ	60 GOSUB ' 15(N)
24	PRINTUSING	ФОРМАТНАЯ ПЕЧАТЬ	80 PRINTUSING "###",A,B,C 190 PRINTUSING 110,A%,B,L%,M
25	IMAGE	ФОРМАТ	110 % ### #.## ## ##.###
26	HEXPRINT	ПЕЧАТЬ ШЕСТНАДЦАТИРИЧНЫХ (HEX) КОДОВ	80 HEXPRINT A%,B%,

С О Д Е Р Ж А Н И Е

ОСНОВЫ ЯЗЫКА БЕЙСИК ЭВМ ИСКРА-226

Методические указания по дисциплине
"Программирование на ЭВМ ИСКРА-226"
Ч а с т ь I

Редактор Т.Н.Тихомирова
Технический редактор Н.Н.Васильева
Корректор М.Б.Останович

Подписано в печать 4.01.85 формат 60×90 1/16
Уч.-изд.л. 2,6 Усл.-печ.л. 3,25 Тираж 500 экз.
Изд. № 414 Заказ 24 Бесплатно

Редакционно-издательский отдел МИЭТ.
103473, Москва, К-55, ул. Образцова, 15.
Типография МИЭТ.