88-4
3664э

МОСКОВСКИЙ ОБЛАСТНОЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ

им. Н.К.КРУПСКОЙ

Московское областное управление госпрофтехобразования

РАБОТА НА ДИАЛОГОВОМ ВЫЧИСЛИТЕЛЬНОМ КОМПЛЕКСЕ И ПЕРСОНАЛЬНОЙ ЭВМ "АГАТ"

Методическое пособие

(для учащихся и учителей средних ПТУ)

МОСКВА 1987 г.

87- 18184
ГОС. ПУБЛИЧНАЯ
БИБЛИО...
Ленин...
09 198 акт 484

О Г Л А В Л Е Н И Е

П Р Е Д И С Л О В И Е

Методическое пособие предназначено для учащихся, осваивающих работу на компьютерах ДВК-I, ДВК-2 и персональной ЭВМ "Агат". Предполагается, что обучающиеся имеют минимальный опыт программирования и знакомс версией языка Бейсика используемой в ДВК-I. Весь объем необходимых знаний содержится в методическом пособии "Программирование на Бейсике", и настоящее пособие следует рассматривать как продолжение, ориентированное на конкретное применение компьютеров.

Следует иметь в виду, что пособие не является справочником по язнку Бейсик или устройству и эксплуатации компьютеров. Материал изложен так, чтобы осветить наиболее сложные и принципиальные вопросы. Значительно меньше внимания уделяется вопросам, не представляющим затруднений при самостоятельной работе со справочниками и документацией, приложенными к компьютерам.

В первой главе излагается версия языка "Бейсик-ДВК-2". Основное внимание здесь уделено возможностям обработки текстовой информации. Последовательно проводится сравнение версий Бейсика для ДВК-2 и ДВК-I.

Во второй главе описана версия "Бейсик-Агат". Здесь впервые встречаются логические операции, позволяющие строить сложные логические выражения. Большое внимание в этой главе уделяется реализации графических возможностей ПЭВМ-"Агат".

В главе Ш рассматриваются вопросы взаимодействия ЭВМ

с внешними устройствами и излагаются необходимые сведения о программном обеспечении компьютеров ДВК-I, ДВК-2 и ПЭВМ "Агат".

Успешная работа с компьютерами предполагает знание основных правил эксплуатации. Следует также иметь представление о технических характеристиках компьютеров. Эти сведения также нашли отражение в главе Ш.

Изучив материал двух пособий "Программирование на Бейсике" и "Работа с микро-ЭВМ", учащиеся освоят три версии языка Бейсик. Это даст возможность получить правильное представление об алгоритмических языках и о возможностях различных версий языка Бейсик. Изучение только одной версии неизбежно влечет за собой некоторый консерватизм восприятия.

При наличии большого разнообразия вычислительной техники важное значение имеет и знакомство с тремя различными, с точки зрения эксплуатации, компьютерами.

Данное методическое пособие будет полезно не только учащимся, но и преподавателям при отборе материала и подготовке к занятиям.

Пособие создано на основе опыта преподавания, сложившегося на кафедре Информатики и вычислительной техники Московского педагогического института им. Н.К.Крупской.

Глава I. Бейсик ДВК-2

§ I.I. Сравнение версий Бейсик, используемых в ДВК-I и ДВК-2. Ввод и редактирование программы

Версия "Бейсик ДВК-2" отличается от версии "Бейсик ДВК-I" тем, что в ней

- предусмотрена возможность обработки текстовой информации;
- введены дополнительные операторы и расширены возможности операторов Бейсика ДВК-I;
- предусмотрена возможность работы с внешними устройствами;
- имеются различия в оформлении программ и команд диалога.

В этом параграфе рассматриваются основные команды диалога. Команда LIST распечатывает на экран текст программы, находящийся в оперативной памяти. Возможны пять вариантов написания команды:

LIST	распечатывает всю программу
LIST N	распечатывает указанную строку (с номером N)
LIST N1-N2	распечатывает программу, начиная со строки N1 и кончая N2
LIST N1-	распечатывает, начиная со строки N1, и до конца программы
LIST -N2	распечатывает с начала программы и до строки N2

В "Бейсике ДВК-I" используется не "-", а запятая и две последние команды невозможны. Программа в Бейсике ДВК-2 имеет ИМЯ (ЗАГОЛОВОК) и при работе оператора LIST имя

выводится на экран. Команда LISTNH работает так же, как и LIST, но не распечатывает заголовка. В версии "Бейсик ДВК-I" программа имени не имеет.

Команды RUN и RUNNH запускают программу для выполнения. Команда RUNNH не выводит заголовка программ на экран.

Существует пять вариантов команды, удаляющей строки программы:

DEL N	удаляет строку N
DEL N1-N2	-"- все строки с N1 по N2
DEL N1-	-"- с N1 и до конца
DEL -N2	-"- до N2
DEL	-"- всю программу

В версии "Бейсик ДВК-I" возможны только две первые операции, причем вместо DEL требуется полное имя команды DELETE, а вместо "-" запятая "," . Для того, чтобы удалить одну строку, достаточно просто набрать ее номер и нажать клавишу <ВК> (ввести пустую строку).

Команда NEW (отсутствующая в "Бейсике ДВК-I") очищает память (удаляет программу) и присваивает имя новой программе, которая будет вводиться. В этом случае имя указывается в командной строке:

NEW ПОИСК

Если имя не указано, то на экране появится запрос

NEW FILE NAME- (имя нового файла),

в ответ на который можно указать имя. Если это не сделать и просто нажать клавишу <ВК> , то программа получит имя

NONAME - безымянная.

Часто возникает ситуация, когда требуется остановить выполнение уже работающей программы. В этом случае следует одновременно нажать две клавиши - <СУ> и <С> (где С - латинская буква).

На ДВК-I эту же реакцию вызывает одновременное нажатие клавиш <СУ> и <Р> .

§ 2.I. Оформление программы. Константы.

Программа на Бейсике состоит из нумерованных строк. В версии "Бейсик ДВК-2" номера строк могут быть от I до 32767, а в "Бейсике ДВК-I" от I до 8I9I. Команды диалога вводятся без номера строки и считается, что строка с командой имеет нулевой номер.

В "Бейсике ДВК-2" в операторе присваивания ключевое слово LET можно не ставить (опустить), т.е. два следующих оператора эквивалентны

A = 5 LET A = 5

Следует помнить, что в Бейсике ДВК-2 разделителем между двумя операторами, стоящими в одной строке, является косая черта с наклоном влево " \ ", а не двоеточие ":" как в ДВК-I.

Вещественные константы в обоих версиях представляются с точностью в 8 знаков, но на ДВК-2 абсолютная величина числа может быть в диапазоне от 10^{-38} до 10^{+38}, в то время как ДВК-I допускает диапазон от 10^{-9309} до 10^{+9809}. Числа могут быть представлены в одной из двух форм: с фиксирован-

ной запятой или с плавающей запятой. В качестве разделителя целой и дробной частей используется точка:

		представление на Бейсике	
21.25	-0.43	3.4E-8	-0.445E18
		обычное написание	
21,25	-0,43	$3{,}4 \cdot 10^{-8}$	$-0{,}445 \cdot 10^{18}$

В Бейсике ДВК-2 допускается использование целых констант в диапазоне от -32768 до 32767.

В качестве признака целой константы используется символ %

320% 2% 0% - 74%

Следует помнить, что целые и вещественные константы есть разные объекты, даже если их числовые значения равны. Вещественное число без дробной части это не целое (в рамках языка Бейсик 12% не тождественно 12).

Текстовые (строковые или литерные) константы на ДВК-2 есть последовательность символов, заключенная в ограничители. Ограничителями могут быть кавычки ("А5ж") или апострофы ('А5ж'). Если есть необходимость внутри константы использовать кавычку или апостроф, то вся константа должна быть заключена в ограничители другого типа:

'СЛОВО "МАМА"'	- текстовая константа СЛОВО "МАМА"
"ФИЛЬМ 'ПОБЕДА' "	- константа ФИЛЬМ 'ПОБЕДА'

§ 3.1. Переменные и оператор присваивания

Бейсик ДВК-2 допускает использование вещественных, целых и текстовых переменных. Имена вещественных переменных должны состоять или из одной буквы или из буквы с цифрой:

A B H5 B9 C4

Значения вещественных переменных, так же как и вещественные константы представляют числа с точностью до 8 знаков в диапазоне от 10^{-38} до 10^{+38}. Значение вещественной переменной может быть определено или изменено в операторе присваивания, который имеет вид:

LET <имя переменной> = <арифметическое выражение>

Ключевой слово LET в Бейсике ДВК-2 может быть опущено (в скобках < > записываются названия той конструкции, которая должна стоять на месте этих скобок). Приведем примеры:

```
LET A = 5*B+C
A2 = SIN (5*X-3)
```

Целые значения можно хранить в целых переменных (на ДВК-1 целые переменные отсутствуют). Признаком того, что переменная является целой является символ "%" в имени переменной. Перед "%" может стоять или буква, или буква с цифрой:

A% B8% X% P4%

Значение целой переменной может быть определено и изменено в операторе присваивания:

LET A% = <арифмет. выраж.>

P4% = <арифмет. выраж.>

Если значение арифметического выражения в таком операторе оказалось нецелым числом, то дробная часть будет отброшена.

Значение вычислительной техники было бы не столь велико, как это есть на самом деле, если бы компьютеры умели только считать. Часто информация, подлежащая обработке, имеет вид текста, последовательности символов. Например, требуется найти фамилию в списке, слово в предложении, расшифровать условное обозначение (маркировку стали, команду станка с числовым программным управлением). Для того, чтобы эффективно обрабатывать текстовую информацию, в языке должны быть предусмотрены текстовые переменные, значения которых могут меняться в процессе работы программы. Можно дать следующее описание:

Текстовая переменная - это есть ячейка памяти, в которой записана последовательность символов. Программист может обращаться к ней по имени. В ячейку можно записать значение (текст) и можно прочитать (для использования) это значение.

Имя текстовой переменной в Бейсике ДВК-2 может состоять из буквы или буквы с цифрой и должно заканчиваться символом "$" :

A$ B1$ C9$

Значение текстовой переменной состоит не более, чем из 255 символов и может быть определено или изменено в опера-

торе присваивания:

A$ = "УЧЕБНИК"

B$ = "РУССКОГО"

C1$ = "ЯЗЫКА"

(Ключевое слово LET можно не использовать). Общий вид оператора присваивания, в левой части которого стоит текстовая переменная, есть:

LET <имя текстовой переменной> = <текстовое выражение>

Или

<имя текстовой переменной> = <текстовое выражение>

В общем случае можно дать следующее определение оператора присваивания

[LET] <имя переменной> = <выражение>

Квадратные скобки указывают, что конструкция, стоящая в них, необязательна, и ее можно опустить. Переменная и выражение имеют один из 3-х типов

- вещественный
- целый
- текстовый

Значение арифметического выражения целого типа можно присвоить вещественной переменной; переменная не изменит своего типа. Переменной целого типа можно присвоить значение арифметического выражения вещественного типа. При этом дробная часть значения выражения будет отброшена. Например, в результате выполнения оператора

```
LET A% = 25.44
```

Значение переменной A% будет равно целому числу 25%.

Арифметическим выражением являются:

1) Числовая (вещественная или целая) константа;
2) Переменная, принимающая числовое значение;
3) Стандартная функция, принимающая числовое значение;
4) функция, определяемая пользователем;
5) Два арифметических выражения, соединенные знаками арифметических действий - возведения в степень, умножения, деления, сложения, вычитания;
6) Арифметическое выражение, взятое в скобки.

По этим правилам можно построить сколь угодно сложное арифметическое выражение. Если все элементы арифметического выражения принимают значение целого типа, то выражение имеет целый тип. В противном случае получается выражение вещественного типа. Приведем примеры операторов присваивания:

```
A   = 5*X ↑ 7 + 9* X ↑ 278
B% = A% + 5
C   = (5*X+Y) - 3 + D %
F % = X+(2* (Y+3) * Z + 2 5)
```

Порядок выполнения арифметических операций определяется скобками и приоритетом операций. Приоритет установлен в следующем порядке: вначале проводится вычисление стандартной функции и функции, определяемой пользователем, затем возведение в степень, затем умножение и деление и, наконец,

сложение и вычитание. Напомним, что A*B/(C*H) соответствует $\frac{AB}{CH}$, и A → B → C соответствует (A → B) → C.

Если в левой части оператора присваивания фигурирует имя текстовой переменной, то в правой части должно стоять текстовое выражение. Значение текстового выражения есть текстовая величина. Оно может быть присвоено только текстовой переменной. Текстовым выражением являются:

1) текстовая константа
2) текстовая переменная
3) текстовая (строковая) функция
4) два текстовых выражения, соединенных знаком операции "объединение" (" & ")

Например:

```
A$  = "MA" & "MA"
B1$ = "M" & "AM" & "A"
C$  = "M"&"AMA"
C2$ = "MA" & C$
C3$ = C$ & C$
```

Значения переменных A$, B1$, C2$, C3$ после выполнения этих операторов будут равны "MAMA". Иногда вместо знака " & " используют знак "+", однако следует помнить, что это не сложение, а объединение, и A$ + B$ не совпадает с B$ + A$. Например, "M" + "B" есть "MB", а "B" + "M" есть "BM".

§ 4.I. Стандартные функции

Бейсик ДВК-2 содержит набор математических и строковых

функций. Математические функции принимают числовое значение, вычисляемое по значению аргумента (исключая функцию PI, принимающую значение, равное числу π = 3,I4I5926 и функцию RND, значение которой не зависит от аргумента и равно случайному числу в диапазоне от нуля до единицы). В качестве аргумента может стоять любое арифметическое выражение. Приведем список стандартных математических функций. В этом списке под X подразумевается аргумент - любое арифметическое выражение.

I.	SIN(X)	синус от X (аргумент в радианах)
2.	COS(X)	косинус от X (аргумент в радианах)
3.	ATN(X)	арктангенс (arctg) от X (значение в радианах)
4.	SQR(X)	корень квадратный из X
5.	EXP(X)	показательная функция от X
6.	LOG(X)	натуральный логарифм от X
7.	LOG10(X)	десятичный логарифм от X
8.	INT(X)	целая часть от X
9.	SGN(X)	знак X (если X> 0, то функция равна +I, если X< 0, то - -I, если X=0,то - 0)
I0.	ABS(X)	абсолютная величина X ($\lvert X \rvert$)
II.	PI	число π = 3,I4I5926
I2.	RND(X)	функция случайных чисел. От значения аргумента независит. Генерирует случайное число в интервале от 0 до I.

В Бейсике ДВК-I отсутствуют функции LOG-I0(X) и PI. Приведем несколько примеров операторов присваивания

```
A = COS(X+2*B)+SIN(X+3*B)
B = SIN(X+COS(Y))+LOG(Z)
C = SIN(PI/4+Z)
F = COS(X-3)-4
```

Отметим, что последнее арифметическое выражение соответствует $\cos^{4}(X^{3})$.

Рассмотрим две программы, дающие последовательность случайных чисел, в интервале от 0 до I00 (всегда меньше ста, но больше нуля).

```
I0  A = I00 * RND(5)        I0  RANDOMIZE
20  PRINT A                 20  A = I00 * RND (5)
30  GOTO I0                 30  PRINT A
                            40  GOTO I0
```

Программа, записанная слева, при каждом запуске будет давать одну и ту же последовательность чисел. Другая же программа каждый раз будет генерировать новую последовательность: Оператор RANDOMIZE меняет непредсказуемым образом начальное значение этой последовательности.

Рассмотрим теперь функции, обрабатывающие текстовые (отроковые) величины.

I. Функция LEN(X$) определяет длину (количество символов) строки. Под X подразумевается любое текстовое выражение. Рассмотрим примеры:

```
I) A = LEN ("БУКВА")          2) LET B$ = "СТЕНА"
3) LET B$ = "СТ"                 LET A = LEN (B$)
   LET A = LEN (B$ + "ЕНА")
```

Во всех трех случаях значение вещественной переменной A равно 5.

2. Функция TRM$(X$) имеет текстовое значение, получающееся из X$ после отбрасывания начальных и конечных пробелов. Например, программа

```
10 LET A$ = "␣␣␣ БЕЙСИК␣␣ "
20 PRINT LEN(A$)
30 LET B$=TRM$(A$)
40 PRINT LEN(B$)
50 PRINT B$
```

выдаст на экран

- значение длины строки A$, равное 11-ти,
- значение длины строки B$, равное 6-ти,
- значение строковой переменной B$ ("БЕЙСИК")

3. Функция SEG$(X$,N1,N2) выделяет из строки X$ подстроку (часть, сегмент), начиная с символа с номером N1 и кончая символом с номером N2. Например, следующая программа

```
10 A$ = "ТЕЛЕВИДЕНИЕ"
20 B$ = SEG$(A$, 3,5)
30 B1$ = SEG$(A$, 5,7)
40 N = LEN(A$)
50 B2$ = SEG$(A$, 5,N)
60 PRINT B$, B1$, B2$, N
```

выдаст три текстовых значения ("ЛЕВ" "ВИД" "ВИДЕНИЕ") и длину текстовой переменной A$, равную 11-ти. Для того, чтобы выделить первые M символов из строки, можно записать функцию SEG$(A$, 1, M). Для выделения последних

N символов следует сначала определить длину строки:

```
N1 = LEN(A$)
PRINT SEG$(A$,N1-N,N1)
```

Рассмотрим две возможные ситуации с параметрами функции SEG$(A$,N1,N2):

1)	N1 < 1	функция работает как с N1 = 1
2)	N1 > N2 или N1 больше длины строки	функция принимает значение, равное нуль-строке (пустой строке - "")

3. Функция POS(X$,Y$,N) осуществляет поиск в строке X$ подстроки (сегмента), совпадающего со строкой Y$. Обнаружив в строке X$ подстроку Y$, функция POS принимает значение, равное порядковому номеру в X$ первого символа подстроки Y$. Если же подстрока не найдена, то POS принимает значение, равное нулю. Рассмотрим задачу о поиске суффикса "ин" в словах русского языка:

```
10 A1$ = "ГУСИНЫЙ"
20 A2$ = "ПЧЕЛИНЫЙ"
30 A3$ = "ОЛОВЯННЫЙ"
40 PRINT POS(A1$ , "ИН", 1)
50 PRINT POS(A2$ , "ИН", 1)
60 PRINT POS(A3$ , "ИН", 1)
70 PRINT POS(A1$ , "ИН", 5)
```

Программа распечатает на экран четыре числа (4 , 5 , 0 , 0). Первые два соответствуют номеру символа "и" в переменных A1$ и A2$. В переменной A3$ суффикс отсут-

ствует и получилось значение ноль. Четвертый оператор PRINT дал значение ноль, т.к. в той части строки AI$, которая начинается с 5-го символа комбинации "ин" нет.

Рассмотрим еще одну задачу:

Пусть дана текстовая переменная A$, значение которой есть фамилия и имя человека, причем фамилия и имя разделены пробелом. Следует преобразовать эту переменную в переменную, значение которой есть фамилия и первая буква имени с точкой (из "ИВАНОВ ␣ СЕРІЕЙ" следует сделать "ИВАНОВ ␣ С."). Опишем последовательность действий:

I. Найти номер позиции N, в которой стоит пробел.
2. Выделить из A$ первые N+1 символов.
3. Присоединить к ним точку.

Программа будет выглядеть следующим образом:

```
I0  A$ = "ИВАНОВ ␣ СЕРІЕЙ"
20  N = POS(A$, "␣", 1)
30  B$ = SEG(A$, 1, N+1) & "."
40  PRINT B$
```

В заключении параграфа отметим, что функции, определяемые пользователем в рамках Бейсика ДВК-2, могут быть всех трех типов (целые, вещественные, текстовые). Эти функции должны иметь имена, первые две буквы которых есть FN. Функция может иметь от одного до пяти аргументов или не иметь их вообще. Прежде чем использовать функцию следует определить в операторе DEF.

Приведем примеры определения функций:


```
DEF FNB(X,Y) = 5*A*X + 6*Y
DEF FNC$(X$,Y$) = X$ & SEG(Y$, 5, 6)
```

Аргументы функции называют формальными параметрами. Слово "формальный" подчеркивает, что в определении функции фигурируют не конкретные значения, а просто указатели - как надо вычислять функцию. При использовании функций вместо формальных параметров следует ставить фактические параметры, значения которых будут использованы при вычислении функций.

Важной особенностью языка Бейсик является то, что в определении могут стоять не только формальные параметры, но и любые переменные. Единственное условие состоит в том, что эти переменные должны быть определены к коменту использования функции (оператор DEF - это не использование, а определение функции). Если в определении функции указано выражение, в котором не используются формальные параметры, то значение функции не зависит от параметров. Какими бы не были значения фактических параметров, значение функции будет одним и тем же.

§ 5.I. Определенные и неопределенные переменные. Операторы INPUT, READ, DATA.

В Бейсике ДВК-2 так же, как и в Бейсике ДВК-I, все переменные, используемые в программе, должны быть определены; в противном случае будет выдано сообщение об ошибке. Переменная считается определенной, если ее имя встретилось в одной из следующих ситуаций:

I) в левой части оператора присваивания LET

2) в списке оператора интерактивного ввода INPUT

3) в списке оператора чтения данных READ

Рассмотрим оператор INPUT, общий вид которого есть:

INPUT <СПИСОК>

Элементами списка должны являться имена переменных (любого типа), разделенные запятыми. Встретив оператор ввода, Бейсик печатает на экране знак "?" и ждет ввода значений для каждой переменной. Если число вводимых данных (констант) меньше, чем число переменных в операторе, то на экране вновь появится знак "?" и компьютер будет ждать недостающих значений. Избыточное число данных вызовет предупреждение об ошибке. Лишние данные игнорируются, но работа программы не прекращается. При попытке ввести текстовые значения вместо числовых будет выдано сообщение об ошибке. Текстовые значения можно вводить без кавычек (или апострофов), разделяя их запятыми. Для ввода текстовых строк можно воспользоваться оператором LINPUT который вводит строку, включая начальные и конечные пробелы, кавычки, символы пунктуации. Приведем примеры:

```
INPUT  A, B$, C
INPUT  B1, Z, F8$, G$
```

Первый оператор требует ввода двух чисел и одной строки (если будут введены три числа, то второе из них будет интерпретированно как текстовая строка). Второй оператор требует двух чисел и двух строк. Если пользователь заранее не знает, какие данные следует вводить, он может "заготовить" данные в операторе DATA и считывать их с помощью оператора READ.

Оператор READ имеет вид:

READ <СПИСОК>

Элементами списка опять являются имена переменных. Оператор DATA, содержащий данные для READ, записывается следующим образом:

DATA <СПИСОК>

Элементами списка должны являться константы, разделенные запятыми. Строковые константы могут быть в кавычках или без них. Если константа дается в кавычках, то запятая в ней воспринимается как часть константы, а не как разделитель элементов списка. Все операторы DATA независимо от их места положения в программе формулируют единый комплекс (блок) данных. Операторы READ последовательно считывают константы из этого блока. В случае недостатка констант (данных) будет выдано сообщение об ошибке и программа прекратит работу. При необходимости повторить ввод данных, начиная с первого оператора DATA, следует использовать оператор RESTORE.

В качестве примера рассмотрим задачу о поиске суффиксов в словах. Список слов будем задавать в операторах DATA, а интересующий нас суффикс будем вводить (каждый раз новый) в операторе INPUT. О наличии суффикса будем судить по значению функции POS (ноль, если суффикса нет и номер позиции, если суффикс найден). Приведем программу:

```
10 PRINT  "ВВЕДИТЕ СУФФИКС"
20 INPUT  B$
```

```
30  READ AX
40  PRINT AX, POS(AX, BX, 1)
50  GOTO 30
60  DATA  СТЕКЛЯННЫЙ, ГУСИНЫЙ
61  DATA  СМЕШНОЙ, СТРАШНЫЙ
62  DATA ...
```

Количество операторов DATA может быть очень большим. Мы воспользовались оператором безусловного перехода GOTO передающим управление на строку с номером, указанным в GOTO (в данном случае 30). Выполнение программы прекратится, и будет выдано сообщение об ошибке как только исчерпаются все данные в операторах DATA, однако программа к этому моменту свою задачу выполнит.

§ 6.1. Оператор печати PRINT.

Оператор PRINT выводит (распечатывает) данные на экран дисплея в процессе выполнения программы. Общий вид оператора есть

PRINT ⟨список⟩

Список оператора PRINT состоит из элементов, разделенных запятой или точкой с запятой. Элементами списка могут быть

1) константы
2) переменные
3) арифметические и текстовые выражения
4) функция TAB

Оператор без списка выводит пустую строку. Строка

экрана условно разбивается на 5 зон, по 14 позиций каждая. Если элементы списка разделены запятыми, то значение каждого следующего элемента печатается в новой зоне. Когда заполнится пятая зона, вывод результата продолжится на следующей строке экрана. Использование в качестве разделителя элементов списка точки с запятой приведет к уплотненной записи на экране. Числовые значения будут разделяться двумя пробелами, строковые - выводиться без пробелов. Текстовые величины выводятся без ограничивающих символов. Так, например, два следующих оператора

```
PRINT   "ХОРОШО"
PRINT   'Журнал "КВАНТ"'
```

дадут на экране следующие строки:

ХОРОШО
ЖУРНАЛ "КВАНТ"

Рассмотрим функцию TAB, используемую только в операторе PRINT и позволяющую пропустить определенное число позиций в строке:

PRINT TAB (⟨арифм. выраж.⟩) ; ⟨список⟩

Здесь элементы списка будут помещены в позицию, номер которой на единицу больше целого значения арифметического выражения в TAB. Например, следующие операторы:

```
PRINT TAB(5); X
PRINT TAB(PI*10); X
A=5 \ B=8 \ PRINT TAB (A*B); X
```

выдадут на экран значение X, начиная с 6-й, 32-й и 41-й

позиции соответственно. Обратим внимание на то, что после функции TAB должна стоять точка с запятой. Функция TAB позволяет организовать выдачу на экран простейших графиков и зависимостей.

Рассмотрим оператор PRINT A. Если значение A по абсолютной величине не превосходит 999999 и не меньше 0.01, то число будет выдано в форме с фиксированной запятой. В противном случае используется форма с плавающей запятой. Для того, чтобы вывести не все знаки числа, а определенное их количество, можно воспользоваться оператором PRINT USING. Например, операторы:

```
PRINT USING "###.##", 12.432
PRINT USING "###.#", 12.432
PRINT USING "###.####", 12432
```

выдадут на экран числа в виде:

```
12.43
12.4
12.4320
```

С помощью формата, состоящего из символов "#" и точки, можно определить количество выводимых знаков до десятичной точки и после нее (незначащие нули перед целой частью числа заменяются пробелами). Если после комбинации из "#" поставить символы "^^^^", то число будет выведено в форме с плавающей запятой. Например, операторы:

```
PRINT USING "##.# ^^^^", 1,346
PRINT USING "#.## ^^^^", 0.0325, 4521
```


дадут следующие изображения чисел на экране

```
 10.0E-01          34.6E+01
 3.25E-02          4.52E+03
```

Существует довольно много форматов, по которым могут выводиться числа и текстовые строки. За полной информацией о них отошлем читателя к справочнику.

§ 7.1. Условные операторы и организация циклов. Коды символов.

Условный оператор и оператор условного перехода в Бейсике ДВК-2 и Бейсике ДВК-1 выглядят одинаково:

IF ⟨логическое выражение⟩ THEN ⟨группа операторов до конца строки⟩

IF ⟨логическое выражение⟩ THEN ⟨номер строки⟩

IF ⟨логическое выражение⟩ GOTO ⟨номер строки⟩

Группа операторов после THEN выполняется, если логическое выражение имеет значение "истина". В противном случае выполняется оператор, стоящий в следующей строке. Аналогично в операторах условного перехода осуществится переход на строку, номер которой указан после THEN или GOTO только в случае, когда логическое выражение имеет значение "истина". Отметим, что вслед за THEN можно поставить еще один оператор IF :

```
IF SIN(X)>0 THEN IF ABS(X)<6 THEN 50
```

Переход к строке 50 будет осуществлен тогда, когда SIN(X)>0 и |X|<6 . Операторы, стоящие после THEN должны быть разделены символом " \ " :

Общий вид выражения отношения (единственный тип логического выражения, используемый в данной версии языка)есть:

⟨арифмет. или текстов. выражение⟩ ⟨знак отнош.⟩ ⟨арифмет.или текстов. выражение⟩

Выражения справа и слева от знака отношений должны быть одного типа - бессмысленно сравнивать число с текстом. Знаки отношений могут быть следующие:

№	Знак	Пример	Пояснение
I)	=	A=B	равенство
		A$=B$	совпадение двух строк (эквивалентность)
2)	<>	A<>B	неравенство
	><	A$<>B$	несовпадение строк (неэквивалентность)
3)	><	A> B	больше
		A$ > B%	строка A$ больше, чем строка B$
4)	<	A < B	меньше
		A$ < B$	строка A$ меньше, чем строка B$
5)	>=	A> = B	больше или равно
	=>	A$> = B$	строка A$ эквивалентна строке B$ или A$ больше B$
6)	<=	A <= B	меньше или равно
	=<	A%<=B%	строка A$ эквивалентна B$ или A$ меньше B$

Сравнение двух арифметических выражений не вызывает недоумения. Хорошо понятно, что означают условия

A>B SIN(X+5)<>0 A=LOG(2)

По-видимому, не могут вызвать недоумение и условия,устанавливающие эквивалентность (совпадение, одинаковость,равенство) двух строк. Пусть A$ = "BCDE", B$ = "EFGH" и даны следующие операторы:

```
IF  A$="BCDE" THEN 400
IF  B$="BCDE THEN 400
IF  A$=B$ THEN 400
IF  A$=" BCDE" THEN 400
```

Очевидно, что выражение отношения (логическое выражение) имеет значение "истина" только в первом операторе. Обратим внимание, что в последнем случае строка A$ не совпадает со строкой " BCDE ", в которой есть пробел.

Рассмотрим следующие примеры. Пусть в начале программы имеются такие строки:

```
10  A$ = "КЛЮЧ - 584.2"
20  PRINT "ВВЕДИТЕ ПАРОЛЬ"
30  INPUT B$
40  IF A$<>B$ THEN 20
```

С такой программой успешно может работать только тот, кто правильно введет пароль "КЛЮЧ - 584.2". Вернемся теперь к задаче о суффиксах. Пусть мы хотим в одном и том же списке искать слова с разными суффиксами. Исчерпав весь список,следует вернуться за новым суффиксом. В качестве признака конца списка в последнем операторе DATA поставим текстовую

константу " * КОНЕЦ * " . Будем выдавать номер слова в списке, если в нем обнаружен данный суффикс.

```
 10  PRINT   "ВВЕДИТЕ СУФФИКС"
 20  INPUT   B$
 30  I=1
 40  READ A$ \ IF  A$= "*КОНЦ *" THEN 80
 50  IF  POS(A$,B$,1)>0  THEN  PRINT I
 60 I = I + 1
 70 GOTO  40
 80  PRINT  "КОНЕЦ СПИСКА"
 90  GOTO  10
100  DATA    ГУСИНЫЙ, УТИНЫЙ
110  DATA    ОЛОВЯННЫЙ, ....
 .
 .
 .
400  DATA     " * КОНЕЦ * "
```

Рассмотрим теперь, какой смысл можно вложить в утверждение "строка А$ больше, чем строка В$". Прежде всего следует знать, что каждый используемый символ имеет свой числовой КОД. Этим кодом пользуется ЭВМ, обрабатывая символы. Приведем для примера коды некоторых символов, используемых на ДВК-2

СИМВОЛ	КОД
I	49
З	5I
9	57
А	65

Z	90
/	47
*	32

Код мивола легко узнать с помощью функции ASC (X$). Если Х$ - многосимвольное выражение (переменная), в качестве аргумента функции берется только первый символ. Результатом функции будет десятичное чисоло, равное коду аргумента. Символ можно восстановить по значению кода с помощью функции CHR$, аргументом которой должно являться арифметическое выражение. Целая часть этого выражения воспринимается как код. Значение функции CHR$ равно соответствующему символу. Отметим, что коды цифр упорядочены по возрастанию (код 0 меньше кода 9), а коды букв латинского алфавита так, что код А меньше, чем вод В; код В меньше, чем код С и т.д. После всего сказанного, ясно, что, сравнивая два символа, компьютер будет считать большим тот символ, у которого значение кода больше, т.е. "А" < "В" имеет значение "истина", а "С" > " Z " - "ложь".

При сравнении двух текстовых величин сравнение проводится посимвольно, слева направо.

Отметим, что наименьший код имеет символ "" (пустой символ, отсутствие чего-либо, нуль-символ).

Пусть дана некоторая текстовая величина. Как определить - цифра это или какой другой символ. Эту задачу легко решить, если знать, что коды чисел имеют значение от 48 (код нуля) до 57 (код девятки)

```
10  INPUT A$
20  B$=SEG$(A$,1,1)
30  IF B$>CHR$(57) THEN 70
40  IF B$<CHR$(57) THEN 70
50  PRINT "ЦИФРА"
60  GOTO 80
70  PRINT "НЕ ЦИФРА"
80  ....
```

Наиболее удобный способ организации циклов состоит в использовании операторов FOR и NEXT .

Общий вид заголовка цикла есть:

FOR ⟨имя переменной цикла⟩ $= i_1$ TO i_2 [STEP i_3]

Переменной цикла (счетчиком) может быть вещественная или целая переменная, значение которой не меняется в теле цикла.

i_1, i_2, i_3 арифметические выражения, значения которых соответствуют начальному (i_1), конечному (i_2) значению переменной цикла и величине шага, с которым изменяется переменная цикла (i_3). Если шаг равен единице, то его можно не указывать. Последним оператором цикла является NEXT , где указывается имя переменной цикла. Между версиями Бейсика для ДВК-1 и ДВК-2 нет различий в организации циклов. Приведем примеры использования циклов при обработке текстовой информации. Допустим требуется распечатать на экране латинский алфавит. Мы знаем, что минимальным кодом обладает буква A, а максимальным Z . Воспользовавшись функциями ASC и CHR$ можно легко

решить эту задачу:

```
10  FOR  I=ASC("A") TO ASC("Z")
20  PRINT  CHR$(I)
30  NEXT  I
```

Пусть нужно в некотором тексте (A0) подсчитать количество участков (сегментов), совпадающих с B0 . Приведем участок программы, решающий данную задачу

```
100  N=LEN(A$)
110  K=LEN(B$)
120  C=0
130  FOR I=1 TO N-K+1
140  IF SEG$(A$,I,I+K-1)=B$ THEN C=C+1
150  NEXT I
160  PRINT "C=";C
```

§ 8.I. Операторы ON GOTO и ON GOSUB.

Оператор ON GOTO (ON THEN) позволяет осуществить переход к одной из строк, указанных в списке оператора. Общий вид оператора есть

ON ⟨арифмет. выраж.⟩ GOTO ⟨список⟩

Вместо GOTO можно оставить THEN. Приведем пример:

```
ON C GOTO 200 , 350 , 400
```

Если C равно единице, то осуществляется переход к строке с номером двести (первой строке описка оператора), если (C = 2) равно двум, то - к строке 350, в случае C = 3 - к

ГОС. ПУБЛИЧНАЯ БИБЛИОТЕКА

строке 400. Если значение арифметического выражения меньше единицы или больше количества номеров строк в списке, то будет выдано сообщение об ошибке. В данном примере ошибка будет в случае C < I или C > 3.

Оператор ON GOTO в этом примере эквивалентен группе из 4 операторов:

```
IF  C=1   THEN  200
IF  C=2   THEN  350
IF  C=3   THEN  400
PRINT "ОШИБКА" \ STOP
```

С помощью этого оператора можно создать так называемое "меню", в котором пользователю предоставляется право выбрать дальнейший путь выполнения программы:

```
10  PRINT  "ВВЕДИТЕ НОМЕР ЗАДАНИЯ"
20  PRINT  " 1 - ОБУЧЕНИЕ"
30  PRINT  " 2 - КОНТРОЛЬ"
40  PRINT  " 3 - ДОПОЛНИТ.ИНФОРМАЦИЯ"
50  PRINT  " 4 - КОНЕЦ"
60  INPUT  C
70  ON C GOTO  100, 400, 700, 900
```

Аналогично оператору ON GOTO работает оператор ON GOSUB. Например:

```
ON C GOSUB 800, 900, 1200
```

Этот оператор передаст управление на одну из строк подпрограмм (800 при C = 1 ; 900 при C = 2 и 1200 при

C = 3). Встретив оператор RETURN (возврат из подпрограммы), компьютер передает управление на строку, следующую за оператором ON GOSUB.

Отметим здесь, что организация подпрограмм в рамках "Бейсик ДВК-2" осуществляется так же, как и в Бейсике ДВК-1: оператор GOSUB <номер строки> передает управление на строку с указанным номером и запоминает номер строки, с которой произошел уход в подпрограмму. Оператор RETURN возвращает управление в основную программу на строку, следующую за строкой, вызвавшей переход к подпрограмме.

§ 9.1. Использование массивов

Массивы переменных в "Бейсике ДВК-2" состоят из переменных с одним или двумя индексами. Переменные с индексами могут участвовать в операторах LET, PRINT, READ, INPUT, арифметических и логических выражениях на равных основаниях с простыми переменными (без индексов).

В Бейсике ДВК-2 максимальное значение индекса у одномерного массива (с одним индексом) может достигать 32767. Массивы могут состоять из переменных всех трех типов и должны быть определены в операторе DIM :

```
DIM  A1(45,52), B(18)
DIM  B2%(54), C$(32,4)
```

Недопустимо использование одного и того же имени для одномерного и двумерного массива. В качестве примера ис-

пользования массивов рассмотрим задачу об анализе строки XX . Пусть требуется определить, какие символы и в каком количестве встречаются в данной строке. В процессе анализа строки, заполним один массив теми символами, которые встречаются в строке, а в другом массиве в том же порядке будем указывать частоту появления этих символов. Приведем программу:

```
10 DIM A$(100), B(100)
20 INPUT X$
30 P=0
40 FOR M=1 TO 100 \ B(M)=0 \ A$(M)=" " \ NEXT M
50 N=LEN(X$)
60 FOR I=1 TO N
70 C$=SEG$(X$,I,I)
80 FOR K=1 TO P
90 IF A$(K)=C$ THEN B(K)=B(K)+1 \ GOTO 140
100 NEXT K
110 P=P+1
120 B(P)=1
130 A$(P)=C$
140 NEXT I
150 FOR N=1 TO P
160 PRINT A$(N), B(N)
170 NEXT N
```


Глава II. "Бейсик- АГАТ"

В этой главе излагаются сведения о языке "Бейсик-Агат", реализованном на персональной ЭВМ (ПЭВМ) "Агат". Кратко описаны команды диалога и операторы. Основное внимание уделено графическим возможностям машины. Следует обратить также внимание на возможность использования логических операций для построения сложных логических выражений. Команды диалога, осуществляющие взаимодействие с накопителем на гибких магнитных дисках, не рассматриваются, так как этим вопросам посвящена следующая глава.

§ I.II. Команды диалога и оформление программы

Опишем кратко основные команды диалога.

Команда RUN запускает программу для исполнения. Программу можно запустить с любой строки с помощью команды RUN ⟨номер строки⟩ .

Команда LIST распечатывает программу на экране видеоконтрольного устройства (ВКУ), заменяющего в Агате дисплей. Команда LIST N выведет на экран строку с номером N команда LIST N1,N2 выводит на экран строки с N1 по N2.

Команда DEL N1,N2 уничтожает из памяти строки с N1 по N2 включительно. Отметим, что в этой команде следует указывать два номера.

Команда NEW уничтожает из памяти всю программу. В рамках версии "Бейсик - Агат" - программа не имеет имени.

Остановить (прервать) выполнение любой программы или команды можно одновременным нажатием клавиши <УПР> и <СБР> . Одновременное нажатие клавиш <УПР> , <РЕГ>, <С> приостанавливает выполнение программы или команды; продолжить выполнение программы можно, дав команду CONT. Для того, чтобы осуществить ввод информации в ЭВМ,следует нажать клавишу < ↵ > . Признаком того, что компьютер готов к работе в режиме диалога, является знак "] "в начале строки и мигающий курсор.

При вводе программы следует понимать,что изображение строки на экране в момент ее формирования и после вывода по команде LIST , как правило, различаются - компьютер в некоторых случаях сам помещает нужные пробелы в строке.Следует помнить также, что версия "Бейсик - Агат" не допускает лишних пробелов в операторах. Например, операторы GO TO или GO SUB воспринимаются как ошибочные - следует ставить GOTO , GOSUB. Длина строки ВКУ составляет 32 символа. Однако одна строка программы (или команды) может быть больше, чем одна строка ВКУ. Следует помнить, что строка программы (или команды) - это не то же самое, что строка ВКУ.

Разделителем операторов, стоящих в одной строке,является двоеточие ":" .

Важная особенность АГАТА состоит в том, что он может формировать цветное изображение. Для управления цветом используется оператор RIBBON=N , где N - номер цвета. Приведем соответствие между номерами и цветами.

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

После отработки оператора RIBBON вся информация будет выводиться на экран указанным цветом. Цвет предыдущего изображения не изменится. Оператор RIBBON можно использовать и как команду диалога.

Рассмотрим, наконец, два оператора, позволяющих проследить выполнение программы. Если в программе стоит оператор TRACE, то при выполнении всех последующих строк на экране будут выводиться их номера. Оператор NOTRACE отменяет оператор TRACE. Рассмотрим программу

```
10  LET  A=0
20  TRACE
30  FOR  I=1  TO  4
40  PRINT  I
50  NEXT I
60  NOTRACE
70  PRINT A
```

При выполнении этой программы на экран будут выводиться не только значения переменной I , но и номера строк,которые в данный момент выполняются. Операторы TRACE и NOTRACE

могут использоваться как команды диалога. Если дать команду TRACE, то после команды RUN будут выводиться на экран номера выполняемых строк. Отменяется данный режим (режим трасировки) командой NOTRACE.

В конце этого параграфа отметим оператор REM позволяющий включать комментарии в программу. Все, что в строке помещается за оператором REM, не воспринимается ЭВМ и существует только для удобства программиста. В операторной строке могут использоваться любые символы языка Бейсик, в том числе буквы русского алфавита. Это позволяет делать в программе примечания и комментарии в привычной для пользования форме.

§ 2.II. Ввод-вывод информации

По характеру работы операторов READ и DATA версия "Бейсик - Агат" ничем не отличается от версий ДВК. Оператор INPUT выдавая информацию на экран, по-другому размещает ее - экран условно делится на 3 зоны и, если разделителем между элементами является запятая и предыдущее значение уместилось в своей зоне, новый элемент помещается в новую зону.

Оператор INPUT может содержать строку-подсказку.Его общий вид в данной версии есть:

INPUT "текст"; <список>

При наличии текста в кавычках на экран вместо знака вопроса, выдается этот текст-подсказка. Такой оператор почти эквивалентен двум следующим операторм:

PRINT "текст";
INPUT <список>,

но в этом случае за текстом будет выведен знак "?".(Обратите внимание на знак ";" в операторе вывода, блокирующий переход на новую строку.) Такая форма оператора ввода позволяет более компактно записывать программу.

В списке оператора INPUT можно указать несколько имен переменных. Если на запрос со стороны машины пользователь введет меньше констант, чем указано, на экране появятся два знака вопроса - это предложение продолжить ввод.

Рассмотрим теперь оператор GET , который осуществляет задержку выполнения программы до тех пор, пока с клавиатуры не будет введен любой символ. Этот символ воспринимается как значение текстовой переменной оператора GET A$ (к примеру A$). Особенность состоит в том, что на экране работа оператора никак не отображается: ни знак вопроса,ни значение нажатой клавиши не выводятся:

В следующем примере

```
1000  PRINT      "нажмите клавишу С"
1010  GET A$
1020  IF A$<>"С" THEN 1000
1030  PRINT A$
```

по команде RUN на экране будет выведено предложение "нажмите клавишу С",и машина как бы остановится на строке 1010. Выполнение строки 1030 произойдет только при нажатии указанной клавиши.

§ 3.П. Данные и оператор присваивания

"Бейсик - Агат" допускает три типа данных - вещественные, целые, текстовые (строковые).

Вещественные константы представляются с точностью 10 знаков в диапазоне от 10^{-38} до 10^{+38}. Обратим внимание на то, что при записи вещественной константы в форме с плавающей запятой необходим знак у порядка 1.2E+8 ; 14.35+17 ; запись 1.2E8 ; 14.3E17 воспринимается как ошибочная!

Целые константы могут быть в диапазоне от -32767 до +32767. Признаком целой константы является знак " % " :

17% 94% -756%

Текстовые константы должны заключаться в кавычки:

"ПРИШЛА ВЕСНА"

В составе текстовой константы знака "кавычки" быть не может.

Имя переменной может иметь длину до 255 символов, допускается использование русского алфавита. Если последний символ имени есть "%", то переменная считается целой. Имя текстовой переменной заканчивается символом "$". В "Бейсике-Агат" допускаются одномерные, двумерные и трехмерные массивы всех типов. Размер массива ограничен доступной памятью компьютера. Массивы перед их использованием должны быть определены в операторе.

Отметим важное отличие "Бейсика - Агат" от Бейсика-ДВК: "Бейсик - Агат" допускает использование неопределенных переменных, значение этих переменных считается равным нулю для вещественных и целых переменных и нуль - строке для текстовых. Такое обнуление неопределенных переменных часто приводит к ошибкам. Например, следующая программа:

```
10  LET  B = A
20  LET  C =1/B
```

даст ошибку деления на ноль в строке 20, т.к. значение неопределенной переменной равно нулю.

В данной версии языка Бейсик ключевое слово можно не ставить:

```
10  B = 8
20  C = B * 5
```

Приведем список стандартных математических функций:

SIN (X)	синус X в радианах
COS (X)	косинус -"-
TAN (X)	тангенс -"-
ATN (X)	арктангенс (значение в радианах)
INT (X)	целая часть X
SGN (X)	- 1 при X 0 0 при X = 0 +1 при X 0
ABS (X)	абсолютное значение X
SQR (X)	Корень квадратный из X
EXP (X)	показательная функция от X
LOG (X)	натуральний логарифм X
RND (X)	случайное число в интервале от 0 до 1

Использование этих функций обсуждалось ранее.

Выпишем также функции, обрабатывающие строки

LEN (A)	длина (количество символов) строки
MID$ (A$,N1,N2)	выделение под-строки, начиная с N1 – символа длиной N2 символа
LEFT$ (A$,N)	первые N символов строки
RIGHT (A$,N)	последние N символов строки
ASC (A$)	код символа
CHR$ (X)	символ, код которого равен X
VAL (A$)	преобразует текстовое изображение числа в число
STR$ (X)	преобразует число в строку символов

§ 4.П. Логические выражения и операторы передачи управления

Условний оператор и оператор условного перехода в "Бейсике-Агат" выглядит так же, как и в версиях языка, используемых на ДВК:

IF ⟨логические выражения⟩ THEN ⟨операторы⟩
-"- THEN ⟨номер строки⟩
-"- GOTO ⟨номер строки⟩

В простейших логических выражениях – выражениях отношения, допустимо сравнивать как числовые, так и текстовые переменные. Сравнение текстовых переменных проводится по кодам символов. Большим считается тот символ, код которого больше.

"Бейсик-Агат" позволяет из выражений отношения строить более сложные логические выражения с помощью логических операций

AND	логическое "и"
OR	логическое "или"
NOT	логическое "не"

Например:

A>0 AND B>0
A<0 OR B<0

Первое логическое выражение имеет значение "истина" тогда, когда выполняются оба условия (A > 0 и B < 0). Второе логическое выражение имеет значение "истина", если выполняется хотя бы одно из двух. Любое логическое выражение можно взять в скобки (A > 0 AND B > 0)OR(B < 0).

Данное выражение имеет значение "истина" в случае, когда A > 0 и B > 0 – или, когда B < 0. Иногда удобно воспользоваться логическим отрицанием. Следующий пример:

NOT (⟨логическое выражение⟩)

имеет значение "истина", если логическое выражение в скобках ложно. Наличие логических операций позволяет значительно упростить программу.

Оператор FOR (заголовок цикла) в версии "Бейсик-Агат" ничем не отличается от операторов цикла, использовавшихся в версиях ДВК-I, ДВК-2. В операторе NEXT (последний оператор цикла) можно не указывать имя переменной цикла. Телом цикла считается последовательность строк от заголовка до первого встретившегося оператора NEXT.

Так же, как в версиях ДВК-I, ДВК-2 в "Бейсике-Агат" допускается использование операторов:

ON GOTO
ON GOSUB

§ 5.П. Графические возможности ПЭВМ "Агат"

Возможность выдавать на экран не только текстовую, но и графическую информацию, составляет основное преимущество ПЭВМ-"Агат" перед ДВК-2. Способность ПЭВМ "Агат" оперировать различными цветами в значительной мере повышает ценность выдаваемой графической информации. Рассмотрим на примере Агата, как ЭВМ формирует графическое изображение.

Весь экран (в случае Агата - это экран видеоконтрольного устройства) разбит на позиции. На экране можно разместить определенное число строк, а в строке укладывается определенное число позиций. В графическом многоцветном режиме компьютер может поставить в произвольное место экрана (в любую позицию) точку заданного цвета или удалить ее. Поставить точку - означает закрасить позицию требуемым цветом, а удаление точки равносильно окраске позиции в цвет экрана.

Качество изображения зависит во многом от того, на сколько позиций разбит экран. Эта характеристика называется РАЗРЕШЕНИЕМ. В случае Агата возможно 3 режима воспроизведения графической информации. Для графики низкого разрешения (GR) экран разбивается на 64 строки по 64 позиции в каждой (64x64). Точка на экране имеет размер, составляющий 1/64 размера экрана. Для графики среднего разрешения (MGR) разбиение экрана увеличивается в два раза - 128 строк по 128

позиций в каждой. И, наконец, графика высокого разрешения (HGR) имеет 256 строк по 256 позиций. Цветное изображение можно строить только в первых двух режимах; графика высокого разрешения позволяет строить только черно-белые изображения.

В компьютере существует область памяти, где хранится информация о том, в каком состоянии находится каждая точка экрана. Каждой точке экрана соответствует своя ячейка памяти. Содержание памяти отображается на экран, и эту область называют ЭКРАННОЙ ПАМЯТЬЮ. Если нам нужно что-то изменить на экране, мы должны произвести изменения в экранной памяти. В случае Агата экранная память размещается в одной из областей оперативной памяти и, прежде чем формировать графическое изображение, следует "объявить", где мы собираемся разместить экранную память. Это следует делать с помощью одного из следующих операторов.

GR=N, где $2 \leqslant N \leqslant 31$
MGR=N, где $1 \leqslant N \leqslant 7$
HGR=N, где $1 \leqslant N \leqslant 7$

Первый оператор открывает возможность работать с графикой низкого разрешения, второй - с графикой среднего разрешения и третий - с графикой высокого разрешения. Номер указывает область, занятую экранной памятью.

Следующий шаг построения изображения состоит в объявлении цвета рисунка. Это осуществляет оператор

COLOR=X, где $0 \leqslant X \leqslant 7$

Соответствие цветов то же, что и при использовании опе-

ратора RIBBON

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

Следует иметь в виду, что, не объявив цвет изображения, вы начнете рисовать нулевым цветом, совпадающим с цветом экрана. В результате изображение будет неразличимо с фоном экрана. Это весьма распространенная ошибка. Прежде, чем рисовать какой-либо элемент, следует использовать оператор COLOR. Таким образом достигается многоцветность изображения. В случае графики высокого разрешения цвет изображения, разумеется, остается белым, но в операторе COLOR=X значение X не должно быть равно нулю.

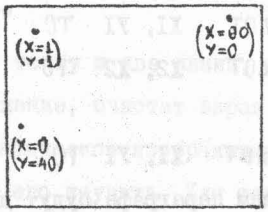
Существуют две основные операции, позволяющие формировать графическое изображение. Одна из них ставит точку в определенное место экрана, а другая проводит линию между двумя заданными точками. Оператор, размещающий точку на экране, выглядит так:

PLOT X,Y

Вместо X и Y могут стоять любые арифметические выражения, значения которых больше нуля и меньше 64 для графики низкого разрешения. Для графики среднего разрешения X и Y

должны быть меньше 128, и для графики высокого разрешения - меньше 256. Числовые значения X и Y задают положение точки на экране. Позиция с номерами X=0, Y=0 находится в левом верхнем углу. Позиция с номерами X=127, Y=128 для графики среднего разрешения совпадает с правым нижним углом. Для графики низкого разрешения этот же угол отмечается координатами X=63 и Y=63, для графики высокого разрешения - X=255, Y=255. Следует помнить, что выход значений X и Y за допустимый интервал приводит к искажению изображения. Обратим особое внимание на то, что с увеличением координаты Y (второго значения в операторе PLOT X, Y) точка будет перемещаться ВНИЗ. Приведем пример программы и получающегося на экране изображения. На рисунке отмечены координаты точек.

```
10  GR  = 3
20  COLOR  = 1
30  PLOT  1,1
40  PLOT  0,40
50  PLOT  60,0
```



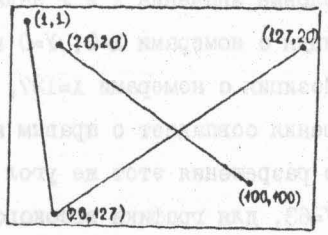
Для того, чтобы провести линию из одной позиции (x1,y1) в другую (X2, Y2), оператор PLOT должен выглядеть следующим образом:

PLOT X1, Y1 TO X2, Y2

Приведем программу и формируемое ею изображение:

```
10  MGR  = 3
20  COLOR  = 1
30  PLOT  1,1 TO  20, 127
40  PLOT  20, 127 TO  127, 20
```

```
50 PLOT  20, 20  TO  100, 100
```



Если в операторе PLOT опущены координаты первой точки

```
PLOT  TO  X2, Y2
```

то линия будет проведена из той позиции, в которой закончилось изображение, задаваемое предыдущим оператором PLOT. Два оператора PLOT можно объединить в один. Например:

```
PLOT  X1, Y1  TO  X2,  Y2
PLOT  X2, X2  TO  X3,  Y3
```

Равносильно

```
PLOT  X1, Y1  TO  X2, Y2 TO X3, Y3
```

В следующем параграфе будут даны примеры программ для ПЭВМ "Агат", дающие графические изображения.

С переходом в режим графики исчезает возможность обмениваться с компьютером текстовой информацией. Даже окончание программы не возвращает в режим диалога.

Для перехода в режим диалога требуется одновременно нажать клавиши <УПР> , <СБР> . В случае необходимости перехода в текстовой режим программным образом (например, чтобы выдать информацию на экран с помощью оператора PRINT) , пользуются оператором TEXT=N , где $2 \leq N \leq 31$

Этот оператор уничтожит с экрана график и предоставит возможность обмениваться с ЭВМ текстовой информацией. После работы оператора TEXT на экране может появиться разнообразный "мусор" - посторонние символы и тексты. Для того, чтобы стереть с экрана ненужную текстовую информацию, следует воспользоваться оператором HOME. Приведем пример:

```
10  CR = 5
20  PLOT  63, 32
30  PLOT  10, 10 TO  40, 15
40  PLOT  TO  12, 29
50  TEXT = 2
60  HOME
70  PRINT, "КОНЕЦ"
```

Нарисовав на экране точку и две линии, компьютер уничтожит графическое изображение, очистит экран и напишет на экране слово "КОНЕЦ". Графическое изображение исчезнет так быстро, что мы не сумеем его изучить. Для того, чтобы задержать его на экране обычно пользуются оператором GET. Если в 45 строке поставить оператор 45 GET A$, то компьютер будет ждать ввода текстового символа A$.

Пауза закончится только при вводе одного любого символа с клавиатуры. При этом значение символа никак не отобразится на экране. Работа программы будет продолжена.

§ 6.П. Примеры программ на построение графических изображений

Научимся рисовать простейшие геометрические фигуры.

Для этого рассмотрим следующую программу:

```
10 MGR  = 2
20 REM  КВАДРАТ КРАСНОГО ЦВЕТА
30 COLOR = 1
40 PLOT  20, 20 TO  20, 100 TO  100, 100
50 PLOT  100, 100 TO  100, 20 TO  20, 20
60 REM  ТРЕУГОЛЬНИК СИНЕГО ЦВЕТА
70 COLOR = 4
80 PLOT  50, 30 TO  70, 70 TO  30, 70
90 REM  ЖЕЛТАЯ ТОЧКА В ЦЕНТРЕ ЭКРАНА
100 COLOR = 3
100 PLOT  64, 64
```

Эта программа рисует квадрат, треугольник и точку разными цветами. Обратите внимание на использование операторов REM , дающих пояснения к тексту программы.

Нарисуем теперь квадрат,закрашенный красным цветом:

```
10  GR  = 5
20  COLOR  = 1
30  FOR  Y=10  TO  50
40  PLOT  10,Y  TO  50,Y
50  NEXT
```

Здесь рисуется на экране 41 линия красного цвета. Для организации программы мы воспользовались оператором цикла FOR.

Нарисуем теперь на экране синусоиду. Координата X будет соответствовать аргументу функции SIN(X) . Разместим синусоиду так, чтобы точка со значением $x = 2\pi = 6{,}28$

соответствовала 126 позиции по горизонтали экрана, а X=0 - нулевой позиции экрана. Точке Y=0 поставим в соответствие со строк экрана :

```
10  INPUT  "ВВЕДИТЕ ШАГ ПО X" ; DX
15  MGR  = 3
20  COLOR = 1
30  FOR  X=0  TO  6.28  STEP DX
40  Y= SIN(X)
50  I  = X * 127/6.28
60  J = 60 - 50 * Y
70  PLOT  I,J
80  NEXT
```

В строке 50 произведено преобразование координаты X так, чтобы растянуть график на весь экран. В строке 60 по значению Y вычисляется значение номера строки экрана, в которой должна стоять точка. В этой строке произведено растяжение по оси Y на 50 единиц; если этого не сделать, то график слился бы в прямую линию. Обратим внимание на то, что пришлось поставить знак "-" перед 50*Y для того, чтобы график не получался перевернутым - с возрастанием номера строки точка на экране смещается вниз. Изображение графика сформировано в виде отдельных точек, частота которых определяется шагом изменения аргумента X. Этот шаг DX вводится в 10 строке.

Глава Ш. Работа с внешними устройствами микро-ЭВМ

§ I.Ш. Память ЭВМ

Электронные вычислительные машины предназначены для обработки информации. Для того, чтобы информацию обрабатывать, ее нужно помнить и хранить. В составе ЭВМ существует два вида памяти основная и внешняя.

Основная память подразделяется на оперативное запоминающее устройство (ОЗУ) и постоянное запоминающее устройство (ПЗУ). Информация хранится в ячейках ОЗУ и ПЗУ. ЭВМ умеет быстро извлекать информацию из ячеек и передавать ее для обработки или для вывода на различные устройства, входящие в состав вычислительного комплекса. В ячейки ОЗУ можно (так же быстро) записать новую информацию, уничтожив при этом предыдущую. В ячейках ПЗУ этого сделать невозможно - информация в нем остается неизменной. Другая причина, по которой употребляется термин постоянное запоминающее устройство, состоит в том, что информация из ПЗУ не исчезает при выключении питания ЭВМ. Говорят, что ОЗУ есть энергозависимая память (есть подача энергии - есть информация, отключили энергию - информация исчезла), а ПЗУ - энергонезависимая память. В ПЗУ часто хранится информация, необходимая для работы ЭВМ и вычислительного комплекса и многократно используемая в течение длительного времени. ОЗУ предназначено для временного хранения входной, выходной информации и промежуточных результатов.

Основная память характеризуется высоким быстродействием - возможностью быстро считывать и записывать информацию в ячейки памяти. К сожалению объем внутренней памяти не может быть большим, и часто ее не хватает для хранения всей нужной информации. Тогда, часть информации приходится хранить во внешней памяти, объем которой практически неограничен, но время обращения к этой памяти (время поиска, считывания, записи информации) значительно больше, чем время обращения к основной памяти. ЭВМ может непосредственно работать только с основной памятью. Информацию из внешней памяти она сначала перекачивает (загружает) в ОЗУ и только потом обрабатывает. Внешней памятью ЭВМ могут быть магнитные ленты, магнитные диски, перфокарты, перфоленты и др. Для работы с внешней памятью в составе вычислительного комплекса должны быть соответствующие устройства, позволяющие организовать обмен информацией между ЭВМ и внешней памятью. Кроме указанных устройств нужно еще иметь соответствующие программы, обеспечивающие взаимодействие ЭВМ и устройства внешней памяти. Информация на носителях внешней памяти (на магнитных дисках, магнитных лентах и т.д.) энергонезависима и может храниться достаточно долго. Это обстоятельство увеличивает значение внешней памяти для организации работы с компьютерами, т.к., как правило, нежелательно, чтобы при выключении ЭВМ вся хранимая в ней информация исчезала.

Информация во внешней памяти (а иногда и в основной) организована в определенные блоки, имеющие название ФАЙЛЫ. Про каждый файл можно сказать следующее:

I) где находится его начало

2) в какую область (или в какие области) памяти записан этот файл

Наиболее удобный способ работы с внешней памятью предполагает:

I. Наличие у каждого файла своего имени .

2. Существование возможности быстро обратиться к любому месту памяти (организовать прямой доступ к памяти).

3. Существование каталога - списка файлов с указанием их имен, местонахождения начала каждого файла.

Такой способ обращения к памяти (к файлам) организован, как правило, с помощью дисководов (накопителей информации на магнитных дисках). Носителем внешней памяти в этом случае является магнитный диск.

Другой способ обращения (последовательный доступ) к памяти организуется с помощью магнитофонов (информация хранится на магнитных лентах), когда файлы выстроены в одну цепочку друг за другом. В этом случае можно обойтись и без имени файла - файл фиксируется по его порядковому номеру на ленте.

В файле может быть записана самая разнообразная информация. Это может быть программа на алгоритмическом языке (например, на Бейсике), или данные для обработки, или программы, осуществляющие управление работой ЭВМ.

Объем памяти измеряется в единицах, имеющих название байты. Не вдаваясь в подробности, дадим для оценки следующие

ориентиры:

I. Для того, чтобы запомнить один символ (буква латинского или русского алфавита, знак арифметической операции) нужна память объемом в один байт;

2. для того, чтобы запомнить вещественное восьмизначное число, нужно память в 4 байта.

§ 2.Ш. Основное и периферийное оборудование ЭВМ

Все оборудование вычислительного комплекса принято делить на основное и периферийное. Основное оборудование составляет ядро комплекса, в котором собственно и осуществляется обработка информации. Периферийное оборудование позволяет ядру связаться с внешним миром - передать и принять информацию, обменяться информацией с внешней памятью.

В состав основного оборудования входят:

I. Основная память (ОЗУ и ПЗУ)

2. Процессор

3. Узлы, организующие связь с периферийным оборудованием.

4. Узлы, обеспечивающие работу основного оборудования.

Процессор управляет работой ЭВМ и осуществляет переработку информации, хранящейся в оперативной памяти ЭВМ, распределяет информацию по различным блокам ЭВМ. Работа процессора протекает в соответствии с программами, заложенными в основной памяти. Конечным результатом работы процессора является выдача информации в узлы, организующие связь основного оборудования с периферийными устройствами.

Основной характеристикой процессора является его быстродействие - время, затрачиваемое на одну операцию. Процессоры в ДВК-I, ДВК-2, ПЭВМ "Агат" выполняют по несколько сот тысяч операций в секунду. Следует иметь в виду, что речь идет об элементарных операциях; умножение, например, не является элементарной операцией и требует несколько большего времени.

Важнейшим периферийным устройством современных компьютеров является дисплей, позволяющий отображать вводимую и выводимую информацию на экране и вводить информацию с помощью клавиатуры. В ПЭВМ "Агат" устройство отображения информации на экран выполнено на основе бытового телевизора. Для использования внешней памяти в составе ЭВМ должны быть соответствующие устройства (накопители информации на магнитных лентах или дисках, устройства ввода-вывода перфокарт или перфолент). Для того, чтобы получить информацию в удобном для анализа виде, используют печатающие устройства и графопостроители. Важной характеристикой вычислительного комплекса является скорость обмена информацией между основным оборудованием и внешними устройствами.

§ 3.Ш. Диалоговые вычислительные комплексы.

Работа с ДВК-I

Диалоговые вычислительные комплексы (ДВК) существуют в различных модификациях. Наиболее распространенными являются ДВК-I (ДВК-IM) и ДВК-2 (ДВК-2М). Основу ДВК составляет микро-ЭВМ "Электроника МС II00.I" (Электроника МС I20I"),

выполненная в виде одной платы с комплектом больших и сверхбольших интегральных схем (БИС и СБИС). На базе этих микросхем выполнены основные узлы микро-ЭВМ:

- процессор
- оперативное запоминающее устройство
- устройства связи с внешними устройствами (дисплей I5ИЭ-00-I3, накопитель на гибких магнитных дисках "ГМД 70" или "ГМ 70I2", печатающее устройство).

В состав микро-ЭВМ входит выполненное на СБИС системное постоянное запоминающее устройство (СПЗУ), в котором записаны программы, обеспечивающие взаимодействие ЭВМ со штатными (предусмотренными) внешними устройствами. СПЗУ берет на себя управление ДВК в момент включения. В нем записана также программа "Резидентный проверяющий тест", позволяющая проверить техническое состояние вычислительного комплекса. В составе микро-ЭВМ предусмотрено место (контактирующее устройство), куда можно вставить еще одно ПЗУ (так называемое ПЗУ пользователя), содержащее необходимую данному пользователю информацию.

Микро-ЭВМ обладает малыми габаритными размерами (приблизительно 250 х 300 х I5 мм), потребляет небольшую (менее 20 вт при напряжении 5-I2 в) мощность и широко используется не только в ДВК, но и в составе технологического оборудования.

Прежде, чем приступать в описанию ДВК-I и ДВК-2, рассмотрим, как осуществить контроль технического состояния микро-ЭВМ и комплекса с помощью "Резидентного проверяющего

теста"(РПТ).

После пуска ДВК в соответствии с инструкцией по эксплуатации, на экране появится символ " ∂ " (приглашение к работе в пультовом режиме). Вслед за этим символом следует ввести (сформировав на экране) команду ТО - пуск последовательности тестов. При работе тестов на экран выводятся сообщения.

ТЕСТ <номер теста>

в случае удачного прохождения теста и

ДЕФЕКТ номер дефекта

при обнаружении неисправности. Программа РПТ осуществляет проверку следующих узлов:

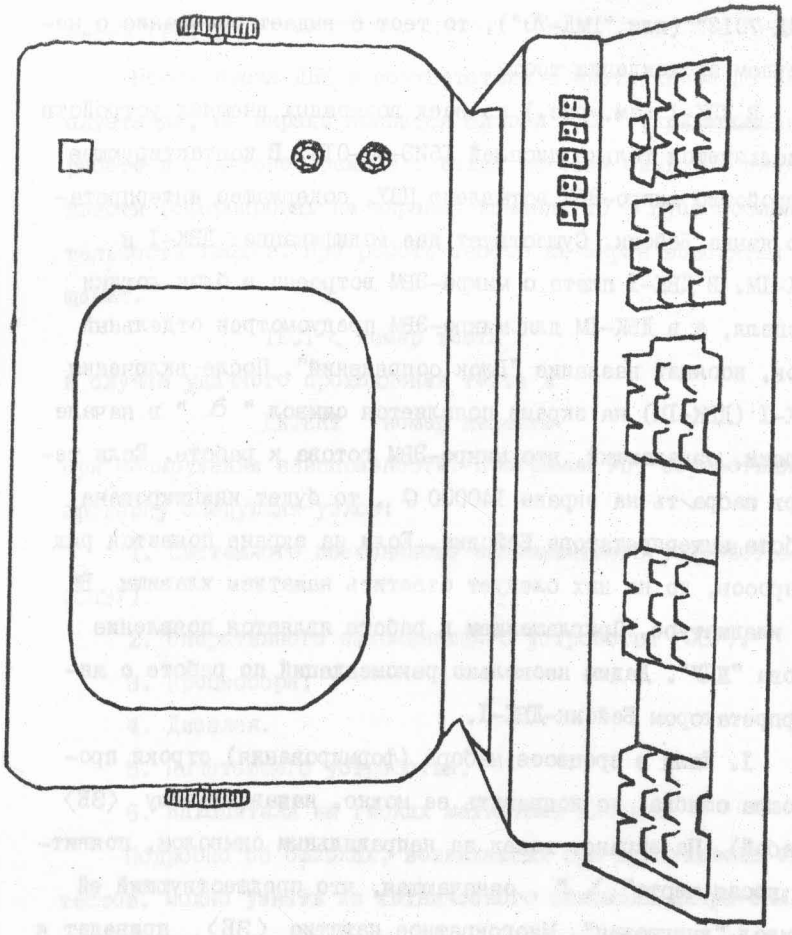
1. Системного постоянного запоминающего устройства (СПЗУ).
2. Оперативного запоминающего устройства (ОЗУ).
3. Процессора.
4. Дисплея.
5. Печатающего устройства.
6. Накопителя на гибких магнитных дисках.

Подробно об ошибках, возникающих при прохождении этих тестов, можно узнать из технического описания микро-ЭВМ. Обратим здесь внимание на тот факт, что тесты 5 и 6 проверяют наличие штатных устройств. Для ДВК-2М в настоящее время поставляется не "ГМД 7012", а более совершенный накопитель "Электроника НГМД-6022". В состав комплекса входит и устройство управления (контролер) для этого дисковода. Но, поскольку РПТ анализирует работу предусмотренного дисковода

"ГМД-7012" (или "ГМД-70"), то тест 6 выдает сообщение о неудачном прохождении теста.

В ДВК-I (см. рис.) из всех возможных внешних устройств используется только дисплей I5ИЭ-00-0I3. В контактирующее устройство микро-ЭВМ вставлено ПЗУ, содержащее интерпретатор языка Бейсик. Существует две модификации: ДВК-I и ДВК-IМ. В ДВК-I плата с микро-ЭВМ встроена в блок логики дисплея, а в ДВК-IМ для микро-ЭВМ предусмотрен отдельный блок, носящий название "Блок сопряжений". После включения ДВК-I (ДВК-IМ) на экране появляется символ " ∂ " в начале строки, означающий, что микро-ЭВМ готова к работе. Если теперь набрать на экране I40000 С , то будет инициирована работа интерпретатора Бейсика. Если на экране появится ряд вопросов, то на них следует ответить нажатием клавиши ВК на клавиатуре. Приглашением к работе является появление слова "ЮЗУ". Дадим несколько рекомендаций по работе с интерпретатором Бейсик-ДВК-I.

I. Если в процессе набора (формирования) строки произошла ошибка, то исправить ее можно, нажав клавишу <ЗБ> (забой). На экране, вслед за неправильным символом, появится косая черта" \ " , означающая, что предшествующий ей символ "уничтожен". Многократное нажатие <ЗБ> приведет к появлению нескольких символов " \ " и "уничтожению" соответствующего числа предшествующих символов, включая пробелы. Любые другие попытки отредактировать (исправить) строку оказываются неудачными, несмотря на то, что иногда у пользователей создается впечатление будто бы им удалось провести



Внешний вид ДВК-I

редактирование более простым способом.

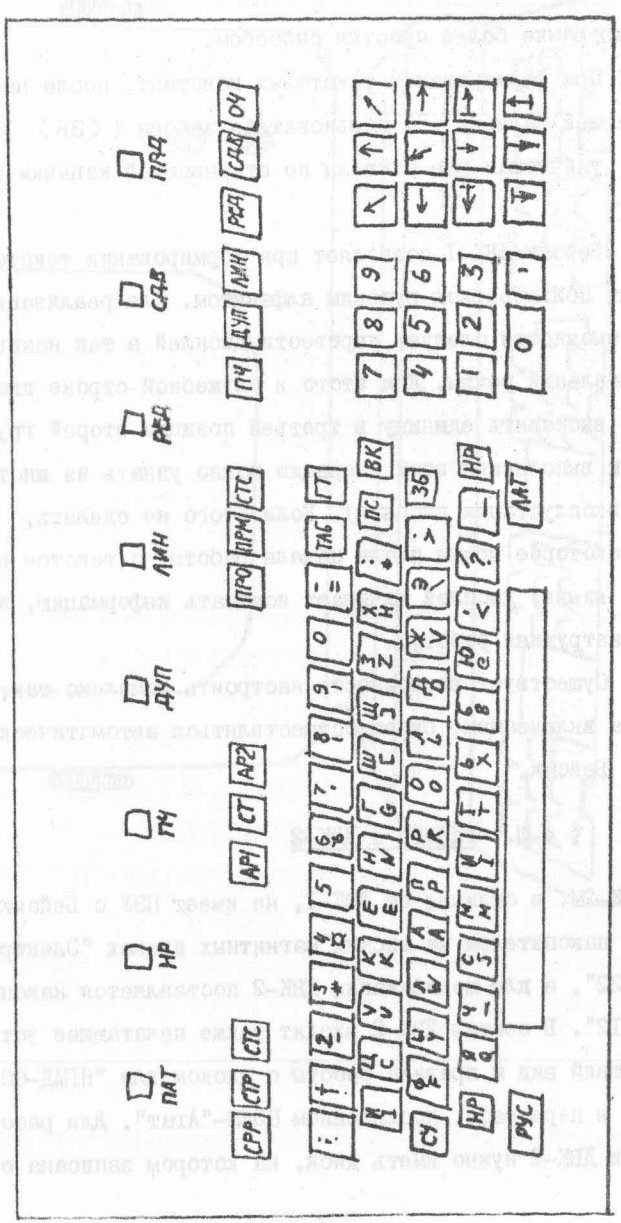
2. При формировании текстовых констант, после набора открывающей кавычки (") пользоваться забоем (<ЗБ>) можно только, уничтожая все символы до открывающей кавычки включительно.

3. Бейсик-ДВК-I позволяет при формировании текстовых констант пользоваться русским алфавитом. Для реализации этой возможности следует перевести дисплей в так называемый, дополнительный режим. Для этого в служебной строке дисплея следует выставить единицу в третьей позиции второй группы. (Порядок выполнения этой операции можно узнать из инструкции по эксплуатации дисплея). Если этого не сделать, то через некоторое время после начала работы, с текстом на русском языке, дисплей начинает искажать информацию, значительно затрудняя работу.

4. Существует возможность настроить комплекс так, что волед за включением, будет осуществляться автоматический выход в Бейсик.

§ 4.Ш. Работа с ДВК-2

ДВК-2М, в отличие от ДВК-I, не имеет ПЗУ с Бейсиком,но оснащен накопителем на гибких магнитных дисках "Электроника НГМД-6022", а для модификации ДВК-2 поставляется накопитель "ГМД-7012". В состав ДВК-2 входит также печатающее устройство. Внешний вид и правила работы с диском для "НГМД-6022" описаны в параграфе, посвященном ПЭВМ-"Агат". Для работы с ДВК-2М и ДВК-2 нужно иметь диск, на котором записана опера-



Клавиатура дисплея ІЭИЭ-00-013

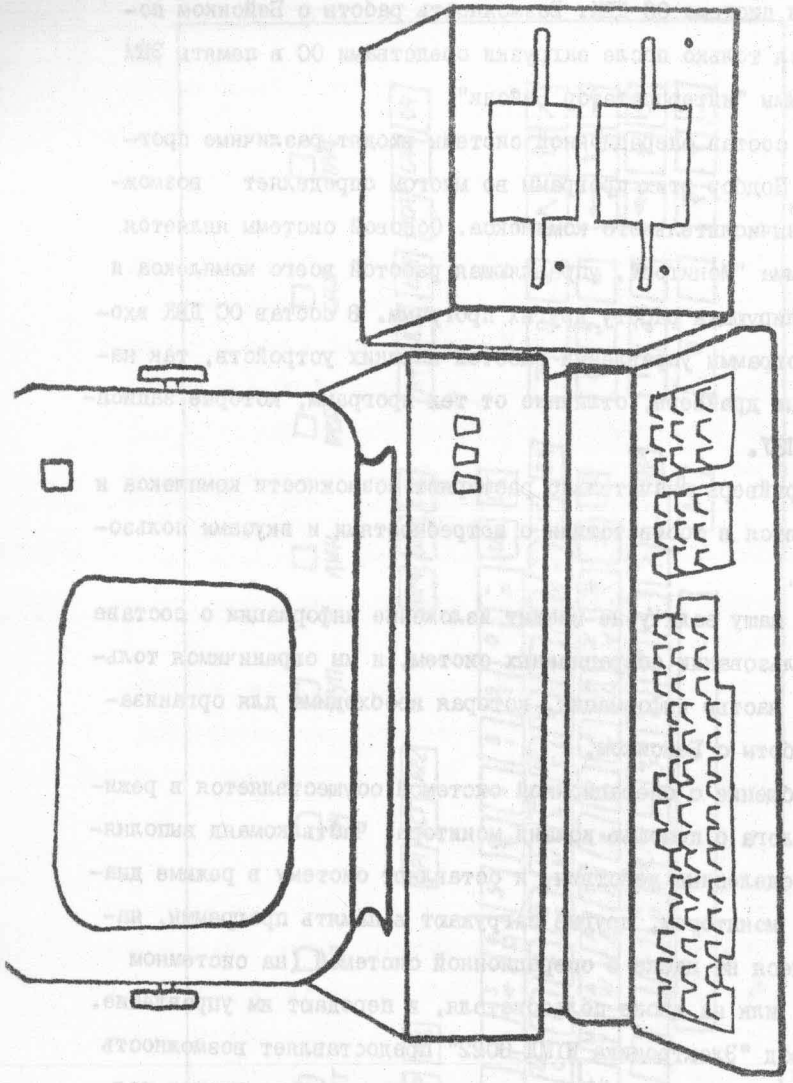
ционная система ОС ДВК. Возможность работы с Бейсиком появляется только после загрузки средствами ОС в память ЭВМ программы "интерпретатор Бейсик".

В состав операционной системы входят различные программы. Подбор этих программ во многом определяет возможности вычислительного комплекса. Основой системы является программа "Монитор", управляющая работой всего комплекса и контролирующая работу других программ. В состав ОС ДВК входят программы управления работой внешних устройств, так называемые драйверы, отличные от тех программ, которые записаны в СПЗУ.

Драйверы значительно расширяют возможности комплекса и выбираются в соответствии с потребностями и вкусами пользователя.

В нашу задачу не входит изложение информации о составе и использовании операционных систем, и мы ограничимся только той частью информации, которая необходима для организации работы с Бейсиком.

Общение с операционной системой осуществляется в режиме диалога с помощью команд монитора. Часть команд выполняют определенные действия, и оставляют систему в режиме диалога с монитором, другие загружают в память программы, находящиеся на диске с операционной системой (на системном диске) или на диске пользователя, и передают им управление. Дисковод "Электроника НГМД-6022" предоставляет возможность работать одновременно с двумя дисками; один из дисков должен быть системным.



Внешний вид ДВК-2М (без печатающего устройства)

Среди большого числа команд первого типа нас интересует только команда DIR, позволяющая просмотреть каталог (список) файлов,хранящихся на диске. В имени файла выделяют часть, определяющую тип файла. Эта часть расположена после точки в имени файла и называется расширением:

PROG.BAS
PIP.SYS
BASIC.SAV

Файлы, сформированные в рамках Бейсика, имеют расширение BAS и именно они представляют для нас интерес, так как в них содержатся программы на Бейсике. Приглашением к диалогу с ОС ДВК является точка в начале строки, после которой следует формировать команду:

.DIR

Просмотрев каталог файлов, мы выясняем, какие файлы, записанные в рамках Бейсика, имеются в нашем распоряжении.

Для работы с Бейсиком следует дать команду (второго типа):

.BASIC

которая загрузает программу "Интерпретатор Бейсик". После этой команды на экране появляется вопрос, на который имеет смысл ответить нажатием клавиши <ВК> . Появление на экране слова READY означает готовность Бейсика к работе.

Перечислим основные команды диалога, осуществляющие взаимодействие с внешними устройствами:

I. OLD <имя файла> - позволяет загрузить с диска программу, записанную в данном файле. Следует отметить, что в Бейсике в имени файла нельзя указывать расширение (вместо PROG.BAS следует писать просто PROG). Если в команде не указать имя файла, то компьютер задаст вопрос, уточняющий, в какой файл вы хотите записать программу:

OLD FILE NAME (имя старого файла)

2. SAVE <имя файла> записать (сохранить) текст программы в виде файла с указанным именем (и расширением BAS). Если имя не указать, то файлу будет присвоено имя программы.

Напомним, что программа в Бейсике ДВК-I всегда имеет имя. Если файл с указанным именем уже существует, то выдается сообщение об ошибке.

Имя программы в памяти ЭВМ устанавливается командой NEW и может быть изменено командой RENAME <новое имя>.

3. REPLACE <имя файла> заменяет содержимое файла с указанным именем на программу, имеющуюся в памяти.

4. UNSAVE <имя> уничтожает файл с указанным именем на диске.

5. SAVE LP: распечатывает текст программы на печатающем устройстве.

6. BYE заканчивает работу Бейсика и передает управление монитору ОС.

Приведем пример работы с Бейсиком ДВК-2:

- Загрузим ОС и вызовем Бейсик командой .BASIC

- После появления слова READY дадим команду

NEW PROBA

и введем небольшую программу

```
10 A=5
20 B=5*A+4
30 PRINT A, B
```

убедимся, что программа есть в памяти ЭВМ и правильно работает.

- Запишем программу на диск с помощью команды

SAVE

- Распечатаем ее на печатающем устройстве с помощью команды

SAVE LP:

-Выйдем из Бейсика по команде BYE, в режиме операционной системы убедимся, что в каталоге диска появился новый файл PROBA.BAS и вновь войдем в Бейсик.

- Загрузим файл PROBA.BAS с помощью комнды OLD PROBA

Изменим ее и запишем новый вариант с помощью команды

REPLACE

Изготовителем ДВК-2 поставляется два диска - один содержащий ОС ДВК и второй - тест-мониторную систему (ТМОС), позволяющую провести детальную проверку всех узлов ДВК-2. Следует иметь в виду, что поставляемый набор программ на диске с ОС не всегда позволяет сразу же организовать работу с комплексом. Информацию на этом диске имеет смысл рассматривать, как заготовку для операционной системы. Пользователь по своему усмотрению может придать ОС ту конфигурацию, которая наиболее соответствует его интересам. Помимо организа-

ции работы с Бейсиком, часто формируют операционную систему так, чтобы она позволяла работать с другими языками высокого уровня (Паскаль, Фортран и др.). Компоновку ОС на диске имеет смысл поручать только квалифицированному программисту. Системный диск может быть организован, так, что при загрузке системы управление сразу же передается на "интерпретатор-Бейсик".

Каталог диска ОС ДВК

```
SWAP   .SYS.  26      MXMUSJ .SYS  70
TT     .SYS.  2       OX     .SYS  2
MX     .SYS   9       LP     .SYS  2
LT     .SYS   5       LL     .SYS  3
NL     .SYS   2       STARTS .COM  1
DIR    .SYS   17      DUP    .SAV  21
PIP    .SAV   16      EDIT   .SAV  19
LINK   .SAV   29      MACRO  .SAV  15
FORMAT .SAV   13      CREF   .SAV  6
DUMP   .SAV   7       PATCH  .SAV  9
ODT    .OBJ   10      SYSMAC .SML  54
SRCCOM .SAV   11      VRF    .SAV  3
PAT    .SAV   7       SYSLIB .OBJ  43
  26 FILES, 432 BLOCKS
  0 FREE BLOCKS
```


§ 5.Ш. Персональная электронно-вычислительная машина (ПЭВМ) "Агат"

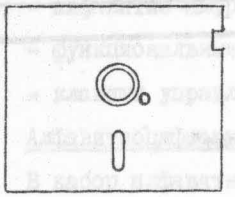
Конструктивно состоит из следующих блоков:

- системный блок;
- блок клавиатуры;
- видеоконтрольное устройство (ВКУ).

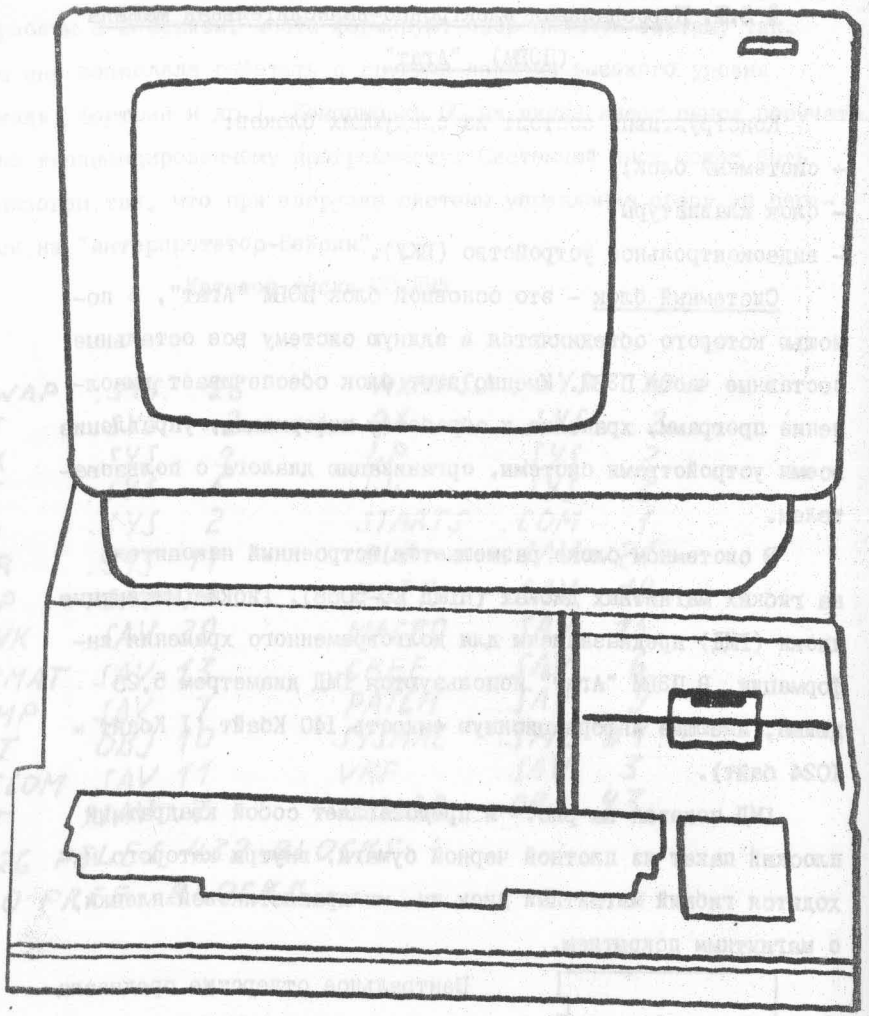
Системный блок - это основной блок ПЭВМ "Агат", с помощью которого объединяются в единую систему все остальные составные части ПЭВМ. Именно этот блок обеспечивает выполнение программ, хранение и обработку информации, управление всеми устройствами системы, организацию диалога с пользователем.

В системном блоке размещается встроенный накопитель на гибких магнитных дисках (НГМД ЕС-5088). Гибкие магнитные диски (ГМД) предназначены для долговременного хранения информации. В ПЭВМ "Агат" используются ГМД диаметром 5,25 дюйма, имеющие информационную емкость 140 Кбайт (1 Кбайт = 1024 байт).

ГМД показан на рис. и представляет собой квадратный плоский пакет из плотной черной бумаги, внутри которого находится гибкий магнитный диск из майларлавсановой пленки, с магнитным покрытием.



Центральное отверстие предназначено для того, чтобы НГМД обеспечил вращение ГМД внутри пакета. Через окно в бумажном пакете,



Персональная ЭВМ "Агат"

магнитная головка НГМД контактирует с ГМД, осуществляя запись или считывание информации. Запись производится на концентрические дорожки (треки). На используемых ГДМ имеется 35 информационных дорожек.

Использование ГМД предполагает неукоснительное выполнение следующих правил:

1. Не касаться магнитного покрытия диска.

2. Не хранить ГМД вблизи электромоторов: генераторов и других источников магнитных полей. Не подносить близко к ГМД магниты и намагниченные предметы.

3. Не сгибать и не складывать диск.

4. Не допускать деформации диска.

5. Не допускать нагрева диска до высокой температуры.

6. Защищать ГМД от прямых солнечных лучей.

7. Правильно вставлять диск в НГМД:

этикеткой - вверх, прорезью для головки - вперед, внутрь НГМД.

Блок клавиатуры является основным устройством, с помощью которого осуществляется общение с ПЭВМ. Всю совокупность клавиш можно разбить на три группы:

- алфавитно-цифровые клавиши;
- функциональные клавиши;
- клавиши управления.

Алфавитно-цифровые клавиши

В набор алфавитно-цифровых клавиш входят 32 буквы рус-

ского алфавита, 26 букв латинского алфавита, 10 арабских цифр, 27 специальных знаков (арифметических действий, скобок и т.д.). Переход от латиноких букв к русским осуществляется путем одновременного нажатия клавиш "РЕГ" и "РУС" либо "РЕГ" и "ЛАТ" при обратном переходе. При этом загорается соответствующий светодиод.

Функциональные клавиши

В правой части клавиатуры имеется набор из 15 особо выделенных клавиш, назначение которых заранее неопределенно. За ними могут быть закреплены некоторые функции, определяемые пользователем и выполняемые после нажатия этих клавиш.

Клавиши управления 1 Сброс – "СБР" + "УПР"

Для того, чтобы прервать выполнение любой программы и установить ПЭВМ в начальное состояние, необходимо одновременно нажать клавиши "СБР" и "УПР". При этом ПЭВМ переходит в состояние приглашения к диалогу, слева на экране высвечивается знак "] " и мигающий курсор.

2. Перевод строки, исполнение – " ↵ "

Информация, набранная на экране, воспринимается ПЭВМ только после нажатия на клавишу " ↵ ". При этом ПЭВМ отвечает тем или иным образом. Для того, чтобы ПЭВМ выполнила какую-либо директиву (команду), набранную на клавиатуре, в конце необходимо нажать клавишу " ↵ ".

3. Повторение – "ПВТ".

Нажатие клавиши "ПВТ" обеспечивает многократную печать на экране символа, нажатого вместе с клавишей "ПВТ". Печать прекращается, когда вы отпускаете либо клавишу символа или

клавишу "ПВТ".

4. Регистр – "РЕГ".

Клавиша "РЕГ" сама по себе никаких символов не генерирует. Она только изменяет символы, клавиши которых нажаты одновременно с клавишей "РЕГ". Например, при нажатии клавиш с изображением цифр вместе с клавишей "РЕГ" вместо цифр, будут высвечиваться те символы, которые изображены на соответствующих кнопках снизу.

5. Управление – "УПР"

Клавиша "УПР" используется только совместно с другими клавишами.

Одновременное нажатие клавиш	действие
"УПР" – " ← "	очистка экрана
"УПР" – "] "	все, что набрано справа за курсором на данной строке – стирается.

6. Клавиши управления курсором. " → ", " ↑ ", " ↓ ", " ← ". Нажатие этих клавиш вызывает соответствующее стрелке перемещение курсора. В случае неправильного набора строки можно с помощью клавиш " ← " вернуться назад в нужное место, исправить символ, затем, перейдя с помощью клавиши " → " к концу строки, продолжить набор строки.

Видеоконтрольное устройство (ВКУ) выполнено на базе унифицированного переносного цветного телевизора "Шилялис" и предназначено для отображения на экран символьной и графической информации, поступающей с ПЭВМ "Агат".

§ 6.Ш. Программное обеспечение и эксплуатация ПЭВМ "Агат"

Программное обеспечение ПЭВМ "Агат" состоит из программы "Системный монитор", дисковой операционной системы (ДОС) и интерпретатора языка "Бейсик-Агат".

Программа "Системный монитор" является одной из основных программ системного программного обеспечения ПЭВМ "Агат", хранится в постоянном запоминающем устройстве. Программа обеспечивает запуск, обмен информацией между ВКУ и клавиатурой. Программа начинает работу сразу после включения ПЭВМ. При этом осуществляется загрузка с НГМД.

Дисковая операционная система (ДОС) записана на ГМД (системний ГМД) и вводится в ПЭВМ с НГМД. ДОС обеспечивает запись и стирание информации на ГМД. ДОС позволяет работать с файлами трех типов:

A – Бейсик – программа;

B – двоичный;

T – текстовой.

Интерпретатор "Бейсик-Агат" служит для выполнения программы и директив языка "Бейсик-Агат". Интерпретатор "Бейсик-Агат" вводится в ПЭВМ одновременно с ДОС и системного ГМД.

Появление на экране ВКУ знака "] " является признаком наличия в ПЭВМ интерпретатора языка "Бейсик-Агат" и является приглашением к диалогу в операторах этого языка.

Приведем список команд диалога, обеспечивающих взаимо-

действие ЭВМ с накопителем на ГМД:

- CATALOG – распечатка каталога диска
- RENAME Ф1, Ф2 – переименование файлов.

Здесь Ф1 - старое имя файла, Ф2 - новое имя файла.

- DELETE <имя файла> – уничтожение файла.
- LOAD <имя файла> – загрузка файла с ГМД в ОЗУ.
- SAVE <имя файла> – записывает программу из ОЗУ на ГМД.
- RUN <имя файла> – запускает программу сразу же после загрузки.

По команде CATALOG на экран ВКУ выводится каталог ГМД, т.е. имена файлов, записанных на ГМД. (Имя файла должно начинаться с буквы и может содержать до 30 любых символов, включая специальные знаки, кроме ","). Наберем команду CATALOG и заставим ПЭВМ выполнить ее, нажав клавишу "↵". На экране ВКУ высветится каталог – системного ГМД, т.к. именно он находится сейчас в НГМД.

B	0 60	HELLO
B	0 II	ТЕСТ.ДАТА
A	0 4I	ТЕСТ
тип файла	размер файла в секторах I сектор-256 байт	имя файла

Первый файл содержит дисковую операционную систему и именно этот файл загружается вначале работы машины. Два других файла содержат тестовую программу, проверяющую работу Агата. Запустить выполнение этой программы можно с по-

мощью команды RUN

RUN TEST

Другой способ запустить эту программу состоит в том, что сначала проводится ее загрузка с помощью команды LOAD, а затем эта программа, находящаяся в этот момент уже в памяти ЭВМ, запускается с помощью команды

RUN

Рассмотрим пример:

I. Очистим память компьютера с помощью команды NEW

Наберем программу:

```
10  A=8
20  B=A*A+A
30  PRINT  A,B
```

2. Убедимся, что программа работает и выдает значения A и B (8,72 соответственно).

3. Запишем эту программу на ГМД под именем ПРОВЕРКА. Для этого дадим команду:

SAVE ПРОВЕРКА

Убедимся, с помощью команды CATALOG , что в каталоге диска появился новый файл с именем ПРОВЕРКА. Обратите внимание на то, что этот файл имеет тип B (Бейсик). Удалим из памяти программу с помощью команды NEW . Убедимся, что программа исчезла; для этого попытаемся распечатать текст программы с помощью команды LIST .

4. Загрузим теперь файл ПРОВЕРКА с магнитного диска. Это делается с помощью команды

LOAD ПРОВЕРКА

В оперативной памяти ЭВМ появилась программа (строки 10,20, 30). Ее можно просмотреть с помощью команды LIST и запустить для выполнения с помощью команды RUN . Если вместо LOAD использовать команду

RUN ПРОВЕРКА

то произойдет не только загрузка, но и запуск программы на выполнение.

5. Удалим с ГМД созданный файл с помощью команды

DELETE ПРОВЕРКА

Включение ПЭВМ "Агат" и порядок работы. В начале работы с компьютером необходимо:

I. Вставить в щель лицевой панели системного блока системный ГМД.

2. Включить ВКУ, нажав до упора кнопку выключателя сети на передней панели ВКУ, при этом на ней загорается красный светодиод.

3. Включить ПЭВМ, тумблером на задней панели системного блока.

При этом:

а) на блоке клавиатуры должен загореться один из индикаторов "РУС" или "ЛАТ";

б) на экране ВКУ появится надпись ** АГАТ ** с характерным звуком "бип";

в) производится загрузка ДОС и интерпретатора "Бейсик-Агат" с системного диска, что подтверждается загоранием красного светодиода на НГМД;

г) после окончания загрузки с системнргр ГМД лампа на ГМД гаснет на экране исчезает надпись * * АГАТ * * и появляется знак "] " и курсор в виде мигающего квадрата. Это означает,что ПЭВМ готова к работе на языке "Бейсик-Агат".

4. Отключение ПЭВМ производится в обратном порядке (I-ым ПЭВМ, 2-ым,ВКУ).

ПЭВМ может работать в двух режимах: программном и непосредственном. В программном режиме любая команда, набираемая на экране ВКУ, должна начинаться с номера и набираемая на экране ВКУ, должна начинаться с номера и завершаться нажатием клавиши " ↵ ". До нажатия клавиши " ↵ " набранная на клавиатуре команда находится в специальной памяти ВКУ и может быть изменена, то есть отредактирована произвольным образом. При нажатии клавиши " ↵ " набранная на экране команда записывается в оперативную память ПЭВМ, но не исполняется. Значит, если перед командой стоит номер, то это означает, что вы работаете в программном режиме, в котором клавиши " ↵ " означает "запись в память" и переносит курсор в начало следующей строки.

Программа собирается из строк,которые последовательно записываются в память ПЭВМ. Номер строки с командой определяет порядок их исполнения, поэтому при составлении программы целесообразно нумеровать строки номерами кратными I0, чтобы иметь возможность вставлять забытые или дополнительные строки. Любая команда (оператор) языка "Бейсик-Агат" должна размещаться на одной строке.

В непосредственном режиме в ЭВМ вводятся команды диалога. Многие операторы языка Бейсик можно использовать и в непосредственном режиме как команды диалога.

Если команда набрана на экране ВКУ без предшествующего ей номера, то при нажатии клавиши " ↵ " она не запоминается в памяти, а сразу исполняется - следовательно, при вводе строки

I0 PRINT "ЭТО ПЭВМ АГАТ"

на экране ВКУ не будет никакого сообщения, тогда как ввод команды

PRINT "ЭТО ПЭВМ АГАТ"

вызывает ее непосредственное выполнение и на экране появится

ЭТО ПЭВМ АГАТ

Значит отсутствие номера строки означает, что Вы работаете в непосредственном режиме и нажатие клавиши " ↵ " вызывает немедленное выполнение набранной на экране команды. В непосредственном режиме можно использовать Агат как мощный калькулятор. Например,команда PRINT SIN(5), COS(5), SIN(5)/COS(5)

выдаст на экран значения тригонометрических функций.

Научный руководитель

зав.кафедрой информатики и вычислительной техники

проф. Ваграменко Я.А.

Авторский коллектив (МОПИ кафедра информатики и вычислительной техники)

Колыхалов П.И.

Шепелева Т.Е.

Плужникова Н.Г.

Устинов В.М.

Оформление (МОПИ каф.информатики и вычислительной техники)

Дружкова Н.Т.

Дорохина Е.В.

Катков С.А.

Консультант (МОУПТО) Киселева Л.А.

Методическое пособие подготовлено по плану темы I4/86 хоздоговора с МОУПТО

Подписано к печати 30.09.86 г. Л-47345
Бумага офсетная №2 Офсетная Формат бумаги 60x84/I6
Тираж 250 экз. Цена 30 коп. Усл.п.л. 2,5 Зак. 13

Адрес редакции:
I03062, Москва, К-62, Подсосенский пер., 20, НИИВШ
Телефон 297-74-99
Отпечатано на ротапринте НИИВШ