АННОТАЦИЯ

 Настоящее пособие содержит описание основных средств языка

Турбо-Бейсик, а также необходимые сведения по среде Турбо-Бейсика.

 Пособие в основном ориентировано на использование при выпол-

нении практикума по курсу "Основы прикладной информатики". Может

также использоваться в качестве справочника по языку Турбо-Бейсик.

 С Московский государственный

 институт стали и сплавов

 1994 г.

 - 3 -

 С О Д Е Р Ж А Н И Е

 1. Основные средства языка.............................. 4

 1.1. Структура программы............................. 4

 1.2. Типы данных..................................... 4

 1.3. Массивы......................................... 5

 1.4. Выражения....................................... 5

 1.5. Стандартные функции............................. 6

 1.6. Основные операторы.............................. 8

 1.7. Операторы цикла................................. 10

 1.8. Подпрограммы, функции и процедуры............... 12

 2. Цвет и графика....................................... 16

 3. Файлы. Работа с файлами данных последовательного и

 прямого доступа...................................... 20

 3.1. Файлы последовательного доступа................. 20

 3.2. Файлы прямого доступа........................... 23

 4. Движущиеся изображения, звуковые эффекты и нестандар-

 тный ввод с клавиатуры............................... 26

 4.2. Оператор DRAW................................... 26

 4.3. Операторы GET и PUT............................. 27

 4.4. Оператор BEEP................................... 28

 4.5. Оператор SOUND.................................. 28

 4.6. Оператор PLAY................................... 28

 4.7 Функция INKEY$.................................. 30

 5. Библиотека стандартных процедур в Турбо-Бейсике...... 31

 6. Рекомендуемый порядок работы на ПЭВМ при создании и

 выполнении программ на языке Турбо-Бейсик............ 32

 - 4 -

 1. ОСНОВНЫЕ СРЕДСТВА ЯЗЫКА

 1.1. Структура программы

 Программа состоит из строк. Одна строка содержит один или

несколько операторов, разделенных двоеточием. Например,

 a=1:b=1

 Перенос оператора на другую строку допускается. В этом слу-

чае используется знак переноса \_. Например,

 a=1:b\_

 =1

 Строки могут нумероваться (но это не обязательно). Ссылка в

нужное место программы может осуществляться также по метке.

  \_Метка . размещается в отдельной строке, начинается с буквы и

содержит любое количество букв и цифр. После метки ставится двое-

точие. Например,

 aaa:

 x=x+1

 ...

 GOTO aaa

 Если первым символом в строке является апостроф ('), то

строка воспринимается как комментарий.

  \_Имя . образуется из букв от A до Z (или от a до z), знака под-

черкивания \_ и цифр от 0 до 9, начинается с буквы. Имя использу-

ется для обозначения переменных, меток, процедур и т.п.

 1.2. Типы данных

 Целый (значения от -32767 до 32767, занимает 2 байта); длин-

ный целый (от 2 5-31  0до 2 531 0, 4 байта); вещественный (4 байта); ве-

щественный двойной точности (8 байт); символьный ( последователь-

ность любых символов, каждый символ занимает 1 байт).

  \_Тип переменной  .определяется типом данных, которые она предс-

тавляет. Указание типа переменной задается соответствующим знаком

после имени.

 a$ - символьный

 a% - целый

 a& - длинный целый

 - 5 -

 a! - вещественный обычной точности

 а# - вещественный двойной точности

 Если знак после имени отсутствует, то эта переменная счита-

ется по умолчанию вещественной обычной точности. То есть

 a!=1.1

и

 а=1.1

одно и то же.

 1.3. Массивы

  \_Описание массивов . осуществляется с помощью оператора DIM с

указанием размеров. Например, оператор

 DIM a(10), b(10:20, 25:45)

описывает одномерный массив a, элементы которого имеют индексы от

0 до 10, и двухмерный массив b, элементы которого имеют индексы :

первый от 10 до 20, второй от 25 до 45.

 Если нижняя граница индексов в описании не указана, то она

считается равной 0.

 Используя оператор  \_OPTION BASE n ., можно задать нижнюю грани-

цу индексов равной n. Например,

 OPTION BASE 1

 DIM a(10)

В этом случае индекс элементов массива a 2  0будет принимать значе-

ния, начиная с 1. Если описание массива отсутствует, то по умол-

чанию верхний индекс равен 10 ( нижний - 0). В описании массива

вместо константы может использоваться переменная. Например,

 DIM a(n)

Значение n должно быть предварительно определено.

 Используя  \_оператор ERASE ., можно освободить память, выделен-

ную под массив, и вернуть ее для дальнейшего использования. Нап-

ример, оператор

 ERASE a

возвращает память, выделенную под массив a.

 1.4. Выражения

  \_Символьные выражения . состоят из символьных констант (после-

довательность символов, заключенная в кавычки), символьных пере-

 - 6 -

менных, символьных функций, соединенных знаком + (конкатенация,

или объединение). Например,

 a$="КАТЯ"

 b$="ЛЕНА"

 f$=a$+" И "+b$

В результате f$ будет иметь значение "КАТЯ И ЛЕНА". При помощи

функций LEFT$, RIGHT$, MID$ можно выбрать часть символьной пере-

менной слева, справа или из середины. Например,

 F1$=LEFT$(f$,4)

 F2$=RIGHT(f$,4)

 F3$=MID$(f$,6,1)

В результате будет F1$ = "КАТЯ", F2$ = "ЛЕНА" и F3$ = "И".

  \_Числовые выражения . включают в себя константы, переменные,

функции, соединенные знаками арифметических операций, при этом

могут использоваться величины разных типов.

  \_Приоритеты . арифметических операций:

 1. Возведение в степень (^)

 2. Умножение, деление (\*,/ )

 3. Деление нацело (\) (например, 5\2 равно 2)

 4. Вычисление остатка от деления (MOD) (например, 45 MOD 19

равно 7)

 5. Сложение, вычитание (+,-)

 1.5. Стандартные функции

 Запись на Бейсике Математическая запись

 ABS (x) │x│

 SIN (x) sin x

 COS (x) cos x

 TAN (x) tg x

 ATN (x) arctg x

 EXP (x) e 5x

 LOG (x) ln x

 LOG2(x) log 42  0x

 LOG10(x) lg x

 INT (x) целая часть х

 SGN (x) знак х (+1 при х > 0,

 0 при х=0, -1 при x < 0)

 SQR (x) квадратный корень из х

 - 7 -

 LEN (а$) количество символов а$

 LEFT$(a$,n) выбирает из a$ n символов,

 начиная с первого

 MID$ (a$,m,n) выбирает из a$ n символов,

 начиная с m-го

 RIGHT$ (a$,n) выбирает n символов а$,

 начиная с последнего

 STR$ (x) преобразует число к

 символьному виду

 VAL (a$) определяет числовое значение а$

 ASC (a$) определяет код первого

 символа а$

 TAB (n) указывает номер позиции для

 вывода следующего элемента в

 списке вывода оператора PRINT

 SPC (n) вывод n пробелов, используется

 в списке вывода оператора PRINT

 RND [(x)] выдает случайное число из инте-

 рвала (0,1)

 Замечания. 1. [] обозначают необязательный параметр.

2. Если х=0, то повторяется последнее сгенерированное число; если

х < 0, то для каждого x 2  0генерируется новая последовательность

случайных чисел; если x > 0 или отсутствует, то генерируется оче-

редное случайное число, но при повторном запуске последователь-

ность повторяется. Можно использовать оператор RANDOMIZE (перед

первым использованием функции RND), чтобы при новом запуске прог-

раммы получать новую последовательность случайных чисел.

  \_Операции отношения .: <, <=, >, >=, <>.

 Левая и правая части отношения - это числовое или символьное

выражение (в последнем случае сравниваются числовые коды симво-

лов). Если отношение удовлетворяется (является истинным), то его

значение равно -1 , если не удовлетворяется (является ложным), то

его значение 0. Отношения могут быть использованы также в число-

вых выражениях. Например, при выполнении оператора

 PRINT 5 > 6, 5 < 6, (5 < 6) \* 15

будет напечатано 0, -1 и -15.

  \_Логические операции .: AND (логическое и ), OR (логическое

или), XOR (исключающее или ), EQV (эквивалентность), IMP (импли-

кация).

 - 8 -

 Логические операции осуществляются над целыми числами -1

(TRUE) и 0 (FALSE).

 -1, если A = -1, B = -1

 A AND B 0, в остальных случаях

 0, если A = 0, B = 0

 A OR B -1, в остальных случаях

 -1, если A и B не совпадают

 A XOR B 0, если A и B совпадают

 -1, если A и B совпадают

 A EQV B 0, если A и B не совпадают

 0, если A = -1, B = 0

 A IMP B -1, в остальных случаях

 1.6. Основные операторы

  \_Оператор присваивания . имеет вид

  1переменная 0 = 1 выражение

  \_Оператор безусловного перехода . имеет вид

 GOTO  1метка

  \_Условный оператор . имеет вид

 IF  1условие 0 THEN  1операторы 0 [ELSE  1операторы 0]

Например,

 IF a < b THEN t=15 : V=16 ELSE t=17

В качестве условия может использоваться целое выражение, которое

интерпретируется как FALSE, если его значение равно 0, и TRUE,

если не равно 0.

 Если после THEN или после ELSE располагается целая группа

операторов, то можно использовать  \_IF блок ., который имеет следую-

щую структуру

 IF  1условие 0 THEN

  1операторы

 ELSE

  1операторы

 END IF

При этом ELSE и операторы за ним могут отсутствовать, т.е. воз-

 - 9 -

можна конструкция

 IF  1условие 0 THEN

  1операторы

 END IF

Если после ELSE необходима проверка условия, то используется опе-

ратор ELSEIF

 IF  1условие 0 THEN

  1операторы

 ELSEIF  1условие 0 THEN

  1операторы

 ELSE

  1операторы

 END IF

  \_Оператор конца END . используется для окончания выполнения

программы. Может использоваться в программе несколько раз. Ис-

пользуется также с IF, SUB, DEF, SELEСT (см.ниже)

  \_Оператор очистки экрана CLS  .обычно располагается в начале

программы для очистки экрана перед выводом результатов программы.

  \_Оператор ввода INPUT . имеет вид

 INPUT [;] [ 1поясняющий текст 0],  1список переменных

Поясняющий текст заключается в кавычки. Переменные в списке отде-

ляются друг от друга запятыми. ";" после INPUT означает, что кур-

сор останется на той же строке, когда будет нажат "Ввод".

  \_Оператор вывода PRINT . имеет вид

 PRINT [ 1список вывода 0]

Элементы списка разделяются "," или ";" или пробелом. В первом

случае вывод следующего элемента начинается в новую зону (строка

экрана условно разделена на 5 зон по 14 позиций каждая), во вто-

ром - через пробел (для чисел) или непосредственно вслед (для

символьных строк) за предыдущим элементом.

 Элементом списка может быть имя переменной, константа (сим-

вольная заключается в кавычки) или выражение (перед выводом вы-

числяется).

  \_Оператор вывода PRINT USING . имеет вид

PRINT USING  1формат 0;  1список вывода

 Формат задается в виде символьной строки (заключенной в ка-

вычки), задающей форму вывода каждого элемента списка по следую-

щим правилам для чисел : # - для каждой цифры, ^^^^ - для вывода

в показательной форме. Остальные символы обозначают сами себя.

 - 10 -

Например 1,

 ##.## - будет выведено 2 цифры в качестве целой части и две

- дробной. Если целая часть содержит более двух цифр, то перед

числом будет напечатан знак % .

 #.## ^^^^ - вывод будет осуществляться в показательной фор-

ме.

 Для символьных строк :

 "\ 1  0 \" - (два пробела) - будет выведено четыре символа

 "\\" 1  0- (без пробела) - будет выведено два символа

Вообще количество выведенных символов будет равно числу пробелов

между двумя наклонными линиями + 2.

 "!"  1  0- будет выведен один символ

 "&"  1  0- будут выведены все 1  0символы

Например,

 A$= "КОТ" : GA= 6.5

 B$= "СОБАКА" : GB= 15.3

 PRINT USING "& ВЕСИТ #.#,\ \ ВЕСИТ ##.# КГ";A$,GA,B$,GB

При выполнении этого оператора на экране появится

 КОТ ВЕСИТ 6.5, СОБАКА ВЕСИТ 15.3 КГ

  \_Оператор определения положения курсора

 LOCATE i,j

устанавливает координаты (номер строки i и номер столбца j), в ко-

торые начнется вывод данных оператором PRINT.

 1.7. Операторы цикла

1. FOR i=i 4нач 0 to i 4кон 0 [STEP ih]

  1операторы

 NEXT i

 i, i 4нач 0, i 4кон 0, ih - соответственно, управляющая переменная

цикла, ее начальное, конечное значения и шаг изменения. Если

ih=1, то шаг можно опустить. Например,

 FOR i=1 to 10

 ....

 NEXT i

 FOR i=1 to 10

 FOR j=1 to 20

 ....

 NEXT j: NEXT i

 - 11 -

или

 FOR i=1 to 10

 FOR j=1 to 20

 ....

 NEXT j,i

2. DO WHILE  1условие

  1операторы

 LOOP

или

 WHILE  1условие

  1операторы

 WEND

(операторы выполняются 1, 0 пока выполняется условие)

3. DO UNTIL  1условие

  1операторы

 LOOP

(операторы выполняются до тех пор, пока не будет выполнено усло-

вие. Как только условие будет выполнено, осуществляется выход из

цикла).

4. DO

  1операторы

 LOOP

Выход из цикла в этом случае осуществляется оператором EXIT.

  \_Оператор EXIT  . 1( 0выход) используется с SELECT, DEF, FOR, IF,

LOOP, SUB, WHILE. Например,

 EXIT FOR осуществляет выход из цикла, организованного опера-

торами FOR/NEXT до его завершения

 EXIT IF - выход из любого места IF блока

 EXIT LOOP - выход из цикла, заканчивающегося оператором LOOP

или WEND.

Остальные случаи использования EXIT будут рассмотрены вместе с

соответствующими операторами.

  \_Операторы выбора . ON/GOTO и ON/GOSUB

 ON n GOTO  1метка 0 [,  1метка 0...]

n может принимать значения от 0 до 255. Переход осуществляется на

метку, определяемую значением n. Например, при n = 2 переход осу-

ществляется на вторую метку в списке. Если n = 0 или n больше ко-

личества меток в списке, то выполняется следующий по порядку опе-

 - 12 -

ратор.

 ON n GOSUB  1метка 0 [, 1 метка 0...]

n принимает значения от 0 до 255. Переход осуществляется к одной

из возможных подпрограмм (см.ниже), определяемых метками в спис-

ке, в зависимости от значения n.

 После выполнения подпрограммы возврат осуществляется к опе-

ратору, следующему за ON/GOSUB.

 1.8. Подпрограммы, функции и процедуры

  \_Подпрограмма . - помеченная последовательность операторов, за-

канчивающаяся оператором RETURN. Выполняется, когда достигнут

GOSUB.

 ....

 GOSUB aa

 ....

 END

 aa:

  1операторы

 RETURN

Оператор RETURN осуществляет возврат к оператору, непосредственно

следующему за GOSUB.

  \_Функция . может быть определена одной строкой (оператор-функ-

ция) или несколькими строками (подпрограмма-функция).

 DEF FN  1имя 0 [( 1список параметров 0)] =  1выражение

(это определение оператора-функции) 1.

Здесь список параметров может включать 1  0не более 16 параметров.

 DEF FN  1имя 0 [( 1список параметров 0)]

  1описание переменных

  1операторы

 [EXIT DEF]

 ....

 [FN  1имя 0 =  1выражение 0]

 END DEF

(это определение подпрограммы-функции) 1.

Обращение к функции записывается там, где нужно получить ее зна-

чение. При этом параметры заменяются их фактическими значениями.

Например 1,

 DEF FNF (x,y) = x \* x + y \* y

 - 13 -

 a = 3 : b = 4

 P = FNF (a,b)

или

 P = FNF (3,4)

Функция возвращает одно значение, тип которого определяется име-

нем функции. Определение функции может располагаться в любом месте

программы (обычно в начале или в конце). Например,

 F = FNa (1.5)

 PRINT F

 DEF FNa (x)

 FNa = x\*x

 END DEF

 END

или

 F = FNa (1.5)

 PRINT F

 END

 DEF FNa (x)

 FNa = x\*x

 END DEF

В результате будет напечатано 2.25.

 Список параметров оператора-функции может включать только

имена переменных.

  \_Процедура . - это последовательность строк программы, располо-

женная между операторами SUB и END SUB и выполняемая при обраще-

нии к ней при помощи оператора CALL.

 SUB  1имя 0 [( 1список параметров 0)]

  1описание переменных

  1операторы

 [EXIT SUB]

 END SUB

Имя может содержать до 31 символа. Список параметров (не более

16) может включать имена переменных и массивов. Для массивов в

скобках указывается размерность. Например, 1 - для одномерного

массива, 2 - для двухмерного и т.д.

 В обращении к процедуре за именем фактического массива сле-

дуют пустые скобки. Например,

 SUB SUM (x (1), n, S)

 S = 0

 - 14 -

 FOR I=1 to n

 S = S + x (I)

 NEXT I

 END SUB

 DIM A (3)

 A (1) = 1 : A (2) = 2 : A (3) = 4

 CALL SUM ( A (), 3, S)

 S = S/3

 PRINT S

 END

Эта программа вычисляет среднее значение элементов массива A,

процедура SUM суммирует элементы массива.

 Основные отличия процедуры от подпрограммы-функции :

1 - процедура не обязана возвращать значение, она не имеет типа;

2 - вызывается оператором CALL, а не из выражения.

 Отдельные переменные, используемые в процедуре (или подпрог-

рамме-функции), могут быть объявлены локальными, т.е. действующи-

ми только в данной подпрограмме (оператор LOCAL), общими, т.е.

действующими как в подпрограмме, так и в основной программе (опе-

ратор SHARED) и статическими, т.е. действующими в данной подпрог-

рамме как LOCAL, но сохраняющими полученные значения до следующе-

го обращения (оператор STATIC). По умолчанию в подпрограмме-функ-

ции переменная является SHARED, а в процедуре STATIC.

 DEF FNF1

 LOCAL i, S

 FOR i =1 to 10

 S = S + i

 NEXT i

 FNF1 = S

 END DEF

После выхода из подпрограммы-функции переменные i и S теряют свои

значения (при входе в подпрограмму всегда S = 0). Эти же имена

могут использоваться в программе для обозначения других величин.

 DEF FNF2

 LOCAL i

 SHARED S

 FOR i = 1 to 10

 S = S + i

 NEXT i

 - 15 -

 FNF2 = S

 END DEF

В этом случае S сохраняет значение после выхода из подпрограммы и

имеет один и тот же смысл во всех частях программы.

 DEF FNF3

 LOCAL i

 STATIC S

 FOR i = 1 to 10

 S = S + i

 NEXT i

 FNF3 = S

 END DEF

В этом случае S сохраняет значение после выхода из подпрограммы

до следующего обращения к ней, т.е. после первого обращения зна-

чение функции FNF3 будет 55, после второго - 110 и т.д. (S нельзя

изменять в основной программе).

 - 16 -

 2. ЦВЕТ И ГРАФИКА

 В графическом режиме экран рассматривается как решетка, точ-

ки в узлах которой могут быть включены (белым или другим возмож-

ным цветом) или выключены. Точка идентифицируется координатами x,

y; x - номер позиции по горизонтали (от 0 до 319 или до 639 в за-

висимости от режима), y - номер позиции по вертикали (от 0 до 199

или до 349 в зависимости от режима). Точка с координатами 0,0 на-

ходится в верхнем левом углу экрана.

 Графический режим устанавливается (или отменяется)  \_операто-

 \_ром SCREEN ..

 SCREEN i[,j]

i = 0, j = 0 черно-белый текстовый режим

i = 0, j = 1 цветной текстовый режим

i = 1, j = 0 цветная графика при средней разрешающей способнос-

 ти (320 позиций по горизонтали, 200 по вертикали)

i = 1, j = 1 черно-белая графика при средней разрешающей спосо-

 бности

i = 2 черно-белая графика при высокой разрешающей спосо-

 бности (640 позиций по горизонтали, 350 по верти-

 кали)

 После выполнения оператора SCREEN точка устанавливается в

середину экрана.

  \_Оператор COLOR . 2  0устанавливает цвет фона экрана и палитру (из

трех цветов) для окрашивания точек (для текстового режима - еще и

цвет рамки).

 Существует две палитры цветов: 0 и 1, следующего состава:

  \_Палитра 0 .  \_Палитра 1

 1 - зеленый 1 - голубой

 2 - красный 2 - сиреневый

 3 - желтый 3 - белый

Для фона можно выбрать один из 16 цветов (0 - черный, 1 - синий,

2 - зеленый, 3 - бирюзовый, 4 - красный, 5 - лиловый, 6 - корич-

невый, 7 - белый, 8 - серый, 9 - голубой, 10 - ярко-зеленый, 11 -

ярко-бирюзовый, 12 - ярко-красный, 13 - ярко-лиловый, 14 - жел-

тый, 15 - ярко-белый)

 Оператор COLOR 2  0имеет вид

 COLOR 2  0i, 2  0j[,k]

 - 17 -

где i - цвет фона, j - номер палитры, k - цвет рамки.

Например, операторы

 SCREEN 1,0

 COLOR 9,0

устанавливают цветной графический режим при средней разрешающей

способности; цвет фона - голубой; последующие операторы должны

использовать цвета из 0-й палитры.

 В графическом режиме (для получения на экране неподвижных

изображений) могут использоваться следующие операторы:

 PSET 2  0(PRESET) - включает (или выключает) точку; LINE 2  0- чер-

тит линию; CIRCLE 2  0- чертит окружность или ее часть; PAINT 2  0- зак-

рашивает указанную часть экрана.

 Более сложные операторы для получения движущихся изображений

(DRAW, GET, PUT) рассматриваются в п.4.

 Оператор

 PSET (x, y)

включает точку с координатами x,y.

 Оператор

 PSET STEP (x, y)

включает точку, отстоящую на x позиций по горизонтали и на y по-

зиций по вертикали от текущей точки. Например, если в предыдущем

положении точка имеет координаты (60, 75), то оператор

 PSET (10, 20) определяет точку с координатами (10, 20), а

оператор

 PSET STEP (10, 20) определяет точку с координатами (70, 95).

Операторы

 PSET (x, y), 0

или

 PRESET 2  0(x, y)

стирают точку (x, y) с экрана.

 Оператор LINE 2  0позволяет начертить отрезок прямой, указав его

начало (x1, y1), конец (x2, y2) и цвет k

 LINE 2  0[(x1, y1)] 2  0- 2  0(x2, y2), 2  0k

Если начальная точка отсутствует, то линия будет прочерчена, на-

чиная с точки, в которой остановился предыдущий оператор.

 С помощью оператора LINE 2  0можно также чертить и закрашивать

прямоугольники

 LINE (x1, y1) - (x2, y2), 2  0[k], 2  0B[F]

Точки (x1, y1) и (x2, y2) трактуются при этом как левая нижняя и

 - 18 -

правая верхняя вершины прямоугольника. Если цвет k отсутствует,

то прямоугольник будет начерчен 3-м цветом палитры. При наличии

параметра F площадь внутри прямоугольника будет закрашена. Напри-

мер, оператор

 LINE (80, 175) - (100, 25), 2, BF закрашивает красным цветом

прямоугольник, расположенный между 80-ой и 100-ой позициями по

горизонтали и 175-ой и 25-ой позициями по вертикали.

  \_Оператор CIRCLE . 2  0позволяет строить окружности, дуги, секторы

и эллипсы.

 Оператор CIRCLE 2  0(x, 2  0y), 2  0r, k

чертит окружность радиуса r с центром в точке (x, y) цветом k.

После вычерчивания окружности последней использованной точкой

считается ее центр.

 Оператор CIRCLE (x, y), r, k, ALPHA1,ALPHA2

вычерчивает дугу, имеющую начало под углом ALPHA1 и конец - под

углом ALPHA2.

Например, оператор (при PI = 3.1416)

CIRCLE (160, 100), 80, 2, PI/2, PI строит дугу от PI/2 до PI ок-

ружности радиуса 80 с центром в точке (160, 100) вторым цветом

палитры.

 Если использовать знак " - " перед начальной и конечной точ-

ками дуги, то будет нарисован сектор, т.е. конечные точки дуги

будут соединены с центром. Например,

 CIRCLE (160, 100), 80, 2, - PI/2, - PI

 Можно изобразить сплюснутую (или наоборот вытянутую) окруж-

ность, добавив в оператор CIRCLE еще один параметр. Если значение

этого параметра меньше 1, то окружность будет сплюснута по верти-

кали, если больше 1, то - по горизонтали. Например,

 CIRCLE (160, 100), 80, 2, , , 3/4

Все параметры оператора CIRCLE задаются в определенном порядке и

разделяются запятыми. Обязательными являются только параметры,

указывающие положение центра и радиус окружности. Остальные пара-

метры необязательны, однако, запятые должны быть оставлены, если

пропущены параметры в середине списка.

  \_Оператор PAINT . 2  0позволяет закрашивать большие области экрана.

Для закрашивания необходимо указать произвольную точку внутри

закрашиваемой области (x, y), цвет, которым нужно закрасить об-

ласть (k1) и цвет границы, которую нельзя пересекать при закраши-

вании (k2), k1 и k2 могут совпадать. Оператор PAINT имеет вид

 - 19 -

 PAINT (x, y), k1, k2

Например, оператор

 PAINT (160, 100), 2, 2

закрасит окружность нарисованную последним оператором CIRCLE,

красным цветом, как и цвет окружности, использованный в операторе

CIRCLE.

 - 20 -

 3. ФАЙЛЫ. РАБОТА В БЕЙСИКЕ С ФАЙЛАМИ ПОСЛЕДОВАТЕЛЬНОГО

 И ПРЯМОГО ДОСТУПА

 Если необходимо использовать набор данных или программу мно-

гократно, то необходимо сохранить их на внешнем устройстве памяти

(например, диске или дискете). Для этого создается файл. Файл

должен иметь имя. Имя состоит из двух частей: собственно имя и

расширение

  1имя 0. 1расширение

Имя может содержать до 8 символов, расширение  1-  0до трех.

 Могут использоваться следующие символы

 A - z 0 - 9 ( ) { } @ # $ 1  0% ^ & ! - \_ / ' ~

 Полное имя файла в MS DOS содержит весь путь по дереву от

корневой директории к нужному файлу. Например,

 C:\ST\LAB\work.bas,

где C - имя диска, ST - имя директории на диске, LAB - имя подди-

ректории директории ST, work.bas - имя программы на Бейсике.

 Далее речь пойдет только о файлах данных.

 3.1. Файлы последовательного доступа

 Последовательный файл состоит из записей, доступ к которым

осуществляется последовательно, т.е. n-я запись следует за (n -

1)-ой как при формировании файла, так и при считывании из него.

 Создание последовательного файла включает следующие шаги:

открытие файла (оператор OPEN), вывод данных в файл (операторы

PRINT#, WRITE#, или PRINT# USING) и закрытие файла (оператор

CLOSE).

 Считывание из последовательного файла включает следующие ша-

ги: открытие файла, ввод данных из файла (операторы INPUT#,

INPUT$, LINE INPUT#), закрытие файла.

  \_Оператор OPEN . 2  0подготавливает файл для чтения из файла

(INPUT) или записи в файл (OUTPUT, если создается новый файл,

APPEND, если добавляются записи в уже существующий файл).

 INPUT

 OPEN  1имя файла  0FOR OUTPUT  1  0 AS # 1номер файла

 APPEND

  \_Оператор CLOSE  .закрывает файл, т.е. делает невозможным чте-

 - 21 -

ние или запись.

 CLOSE [# 1номер файла 0]

 Оператор CLOSE без параметров закрывает все открытые файлы.

 Единицей информации, которая передается при записи в файл

или чтении из файла 1,  0является запись (логическая запись) 2.  0Запись

делится на поля. Каждое поле связано с одним элементом данных.

  0Могут быть созданы два типа последовательных файлов:

 1) с разделителями полей, когда каждое поле автоматически

отделяется специальным символом (например, запятой). Для создания

такого файла используется оператор WRITE#, для чтения из него -

оператор INPUT#.

 Данные в файле хранятся в таком виде, как если бы они наби-

рались на клавиатуре при выполнении оператора INPUT, т.е. разде-

лены запятыми и каждая запись заканчивается переводом строки.

 \_Оператор INPUT# . 2  0имеет вид

 INPUT # 1номер файла 0, 2  1список переменных

где переменные в списке по типу должны соответствовать данным в

файле.

 2) поля не разделены 1,  0и каждая запись точно такая же, как

если бы эти данные отображались на экране или печатались на прин-

тере. Для создания такого файла используется оператор PRINT#, для

чтения из него - операторы INPUT$ или LINE INPUT#.  \_Оператор

 \_INPUT$ . передает из файла сразу всю строку (запись) указанной дли-

ны n в символьную переменную, например, ST$

 ST$ 2 =  0INPUT$ (n, # 1номер файла 0)

 \_Оператор LINE INPUT# . 2  0передает в указанную переменную ST$ всю

строку независимо от ее длины

 LINE INPUT # 1номер файла 0, 2  0ST$

 Пример (с разделителями полей):

 OPEN "SEQU.DAT" FOR OUTPUT AS #1

 ST$ = " ИВАНОВ "

 INT% = 1972

 FLOT! = 75.3

'теперь записываем строку в файл

 WRITE #1, ST$, INT%, FLOT!

 ST$ = " ПЕТРОВ "

 INT% = 1975

 FLOT! = 62.5

'записываем другую строку в файл

 - 22 -

 WRITE #1, INT%, ST$, FLOT!

 CLOSE #1

 END

Созданный файл будет следующим

 " ИВАНОВ ", 1972, 75.3

 1975, " ПЕТРОВ ", 62.5

Следующая программа читает из этого файла:

 OPEN "SEQU.DAT" FOR INPUT AS #1

 ST$ = " "

 INT% = 0

 FLOT! = 0

'теперь читаем строку текста из файла

 INPUT #1, ST$, INT%, FLOT!

 PRINT ST$, INT%, FLOT!

 ST$ = " "

 INT% = 0

 FLOT! = 0

'читаем другую строку

 INPUT #1, INT%, ST$, FLOT!

 PRINT INT%, ST$, FLOT!

 CLOSE #1

 END

 Пример (без разделителей полей):

 OPEN "SEQUEN.DAT" FOR OUTPUT AS #1

 ST$ = " ИВАНОВ "

 INT% = 1972

 FlOT! = 73.1

'записываем строку текста в файл

 PRINT #1, ST$, INT%, FLOT!

 ST$ = " ПЕТРОВ "

 INT% = 1975

 FLOT! = 64.3

'запишем другую строку, используя оператор

 PRINT #1, USING "+####$#.##^^^^"; INT%, ST$, FLOT!

 CLOSE #1

 END

Содержание созданного файла будет таким:

 ИВАНОВ 1972 73.1

 1975 ПЕТРОВ 6.43Е+01

 - 23 -

Следующая программа читает из этого файла:

 OPEN "SEQUEN.DAT" FOR INPUT AS #1

 ST$ = " "

 ST$ = INPUT$ (80,#1)

 PRINT ST$

 LINE INPUT #1, ST$

 PRINT ST$

 CLOSE #1

 END

 3.2. Файлы прямого доступа

 Файл прямого доступа состоит из записей, доступ к которым

возможен в произвольном порядке по номеру записи.

 Создание файла прямого доступа включает следующие шаги:

открытие файла (оператор OPEN); задание структуры записи файла,

или разметка буфера (оператор FIELD); передача записи в буфер

(операторы LSET 2  0или RSET); пересылка содержимого буфера в файл

(оператор PUT), при этом если номер записи в операторе PUT 2  0не

указан явно, буфер помещается на место текущей записи, при откры-

тии файла указатель текущей записи равен 1, после каждой операции

ввода или вывода указатель сдвигается на следующую запись; закры-

тие файла (оператор CLOSE).

 Записи передаются в виде строк символов. Поэтому необходимо

использовать соответствующие функции преобразования данных в

строку символов и обратно.

 Ниже перечислены функции и осуществляемые ими преобразова-

ния.

 MKI$ (MKL$) - целое (длинное целое) в 2-х (4-х) байтовую

строку;

 MKS$ (MKD$) - вещественное (вещественное двойной точности) в

4-х (8-ми) байтовую строку.

 CVI(CVL) - 2-х (4-х) байтовую строку в целое (длинное це-

лое);

 CVS(CVD) - 4-x (8-ми) байтовую строку в вещественное (ве-

щественное двойной точности).

 Считывание из файла прямого доступа включает также пять ша-

гов: открытие файла; разметка буфера; передача записи из файла в

буфер (оператор GET); обработка данных буфера (переменные буфера

 - 24 -

доступны из программы); закрытие файла.

 Если вывод в файл и ввод из файла осуществляются в одной

программе, то открытие и закрытие файла осуществляется один раз

перед началом и после окончания всех операций с файлом.

  \_Оператор OPEN . 2  0имеет вид

 OPEN 2  1имя файла 2  0AS # 1номер файла 2  0LEN 2  0=  1длина записи

(отсутствие указания режима ввода или вывода означает, что файл

прямого доступа).

  \_Оператор FIELD . 2  0имеет вид

 FIELD # 2  1номер файла 0, 2  0n1 2  0AS 2  1имя1 0 [ 2, 0n2 2  0AS 2  1имя2 0, 2  0...] 2  0,

где n1, n2, ... - длина поля в байтах под переменные  1имя1 0,  1имя2 0,

... ( 1имя1 0,  1имя2 0, - символьные переменные ).

  \_Операторы .  \_LSET, RSET . 2  0имеют вид

 LSET 2  1имя 2  0= 2  1символьное выражение

 RSET 2  1имя 2  0= 2  1символьное выражение

 Оператор LSET (RSET) заполняет одно поле записи в буфере

(оператор LSET - с начала, RSET - с конца, оставшиеся позиции за-

полняются пробелами).

  \_Оператор PUT . имеет вид

 PUT 2  0# 2  1номер файла 2  0[ 2,  1номер записи 0]

 Этот оператор передает содержимое буфера в файл на место за-

писи с указанным номером (или текущей, если номер не указан).

  \_Оператор GET . 2  0имеет вид

 GET 2  0# 2  1номер файла 2  0[, 2  1номер записи 2и 0]

 Этот оператор передает содержимое записи с указанным но-

мером в буфер (если номера нет, передается текущая запись).

 В качестве примера приводится программа, которая формирует

файл прямого доступа из записей следующей структуры:

 N п/п Фамилия Год рождения

В программе переменные обозначены n%, nam$, dat%; для соответс-

твующих полей буфера используются обозначения np$ (2 байта), nm$

(10 байт), dt$ (6 байт).

 OPEN "GSR.DAT" AS #1 LEN = 18

 FIELD #1, 2 AS np$, 10 AS nm$, 6 AS dt$

 FOR i%=1 to 3

 INPUT n%, nam$, dat%

 LSET np$ = MKI$ (n%)

 LSET nm$ = nam$

 LSET dt$ = MKI$ (dat%)

 - 25 -

 PUT #1,i%

 NEXT i%

 CLOSE #1

 END

Следующая программа осуществляет ввод данных из файла GSR.DAT и

их печать в обратном порядке.

 OPEN "GSR.DAT" AS #1 LEN = 18

 FIELD #1, 2 AS np$, 10 AS nm$, 6 AS dt$

 FOR i% = 3 to 1 STEP -1

 GET #1, i%

 n% = CVI (np$)

 dat%=CVI (dt$)

 PRINT n%, nm$, dat%

 NEXT i%

 CLOSE #1

 END

 - 26 -

 4. ДВИЖУЩИЕСЯ ИЗОБРАЖЕНИЯ, ЗВУКОВЫЕ ЭФФЕКТЫ

 И НЕСТАНДАРТНЫЙ ВВОД С КЛАВИАТУРЫ

 4.1. Оператор DRAW

 При помощи оператора DRAW можно чертить (рисовать) на экране

различные фигуры (предметы), используя параметры Un, Dn, Ln, Rn

для перемещения на n позиций от текущей точки вверх (U), вниз

(D), влево (L) или вправо (R); En, Fn, Gn, Hn для перемещения по

диагонали на n точек вверх и вправо (E), вниз и вправо (F), вниз

и влево (G), вверх и влево (H); M h,V для перемещения в точку с

координатами h и V; M  7+ 0h, 7+ 0V для относительного перемещения на h

позиций вправо (+) или влево (-) и на V позиций вниз (+) или

вверх (-). Кроме того, параметр An устанавливает угол n, где n=0

(0 градусов), 1 (90 градусов), 2 (180 градусов) или 3 (270 граду-

сов); TAn - поворачивает на угол n, где n меняется от -360 до 360

градусов. При n>0 поворот осуществляется против часовой стрелки,

при n<0 - по часовой стрелке; Cn - устанавливает цвет n из теку-

щей палитры; Sn - определяет масштаб n/4, n может меняться от 1

до 255 (по умолчанию n=4); B - подавляет высвечивание точек по

пути перемещения в соответствии с командой С (по умолчанию путь

перемещения высвечивается в виде линий, С содержит какие-либо из

перечисленных параметров); N означает возврат в исходную позицию

после выполнения команды перемещения С.

 Последовательность параметров, определяющая режим и порядок

перемещений, заключается в кавычки и указывается в операторе DRAW.

Например, оператор

 DRAW "M 160, 100"

осуществляет перемещение от текущей позиции в позицию (160, 100),

оставляя при этом след в виде закрашенных точек. Оператор

 DRAW "BM160, 100"

осуществляет то же перемещение, но не оставляя следа. Оператор

 DRAW "L10 U10 R10 D10"

прочерчивает линию на 10 позиций влево, на 10 позиций вверх, на

10 позиций вправо и на 10 позиций вниз, т.е. образует квадрат.

Пробелы, разделяющие параметры, необязательны и используются для

наглядности. Оператор

 DRAW "BM 180, 20 NU10 ND10 NL10 NR 10"

 - 27 -

осуществляет перемещение без обозначения пути в точку (180, 20),

а затем чертит крест (знак +).

 В операторе DRAW можно использовать не только символьные

константы (см.выше), но и символьные переменные, в которых зара-

нее заготавливаются отдельные фрагменты рисунка (или весь рису-

нок), что позволяет воспроизводить их многократно различными опе-

раторами DRAW. Например,

 SQAR$ = "L10U10R10D10"

 DRAW SQAR$

Аргументы перемещения n, k, V могут быть заданы константами, как

в приведенных выше примерах операторов DRAW, или переменными. В

последнем случае нужно использовать функцию VARPTR$. Например,

операторы

 DRAW " E15 "

и

 a = 15

 DRAW " E = " + VARPTR$ (a)

выполняют одно и то же действие. Это же относится и к операторам

 I% = 1

 DRAW " M = " + VARPTR$ (I%) + " , 40"

и

 DRAW " M = 1, 40"

Но в первом случае (при использовании переменной I%) оператор

DRAW может выполняться при различных ее значениях, например, в

цикле.

 Замечание. Выполнение оператора DRAW возможно только в гра-

фическом режиме (переход при помощи оператора SCREEN).

 4.2. Операторы GET и PUT

 Операторы GET и PUT позволяют создавать движущиеся изображе-

ния.

 При выполнении оператора GET текущее состояние части экрана

запоминается. При выполнении оператора PUT изображение, сохранен-

ное оператором GET, воспроизводится в другом указанном месте эк-

рана, т.е. изображение как бы движется.

 Оператор GET запоминает изображение заданной области экрана в

массиве, который должен быть заранее описан. Оператор GET имеет

вид

 - 28 -

 GET (x1, y1) - (x2, y2), B

(x1, y1), (x2, y2) определяют верхнюю левую и нижнюю правую гра-

ницы запоминаемой области, B - имя числового массива, в котором

сохраняется копия этой части экрана.

 Существует формула, определяющая требуемую длину массива (в

байтах), исходя из размера запоминаемой области и режима работы

экрана. Так при средней разрешающей способности для сохранения n1

точек по горизонтали и n2 точек по вертикали необходимый размер

массива N вычисляется по формуле

 N = 4 + INT((n1\*2+7)/8) \* n2

при высокой разрешающей способности

 N = 4 + INT((n1 + 7)/8) \* n2

 При использовании массива целого типа (один элемент равен

двум байтам) необходимо предусмотреть размер INT((N + 1)/2).

 Оператор PUT помещает копию части экрана, сохраненную опера-

тором GET в любую часть экрана, левый верхний угол которой задан

точкой (x, y). Этот оператор имеет вид

 PUT (x, y), B

 4.3. Оператор BEEP

 Этот оператор вызывает звуковой сигнал определенной частоты

и длительности 0,25 секунды. Используется, когда нужно привлечь

внимание при выполнении какой-либо части программы.

 4.4. Оператор SOUND

 Этот оператор вызывает звуковой сигнал частоты x (от 37 до

32767 Гц) и длительности в "тиках" (1/18 секунды) y

 SOUND x, y

 Оператор SOUND с нулевой длительностью прерывает работу пре-

дыдущего оператора SOUND, даже если тот не отзвучал до конца. Ис-

пользуется для специальных звуковых эффектов (сирена и пр.).

 4.5. Оператор PLAY

 Этот оператор позволяет создавать музыку. Общий вид операто-

ра

 PLAY  1символьное выражение

 - 29 -

где символьное выражение 1  0задает порядок звучания нот, их длитель-

ность, номер октавы и т.д., используя следующие параметры: буквы

C, D, E, F, G, A, B (ноты от "до" до "си"), A 1# 0, или A+, обознача-

ет A диез, A- обозначает A бемоль, аналогично - для остальных

нот; On означает играть в октаве n (n изменяется от 0 до 6, окта-

ва начинается с ноты "до" (буква C), по умолчанию n = 4). Напри-

мер,

 PLAY " O3 FGA O4 ABC "

В этом примере в 3-ей октаве звучат ноты F, G, A, в четвертой ок-

таве ноты A, B, C. Если перед нотой используется знак > или < ,

например, > A, < C то это означает, что номер текущей октавы уве-

личивается или уменьшается на 1.

 Можно вместо номера октавы и символа ноты задать ее порядко-

вый номер (от 1 до 84), указав перед ним букву N. Например,

 PLAY " O4G "

аналогично оператору

 PLAY " N44 "

 Длительность ноты устанавливается параметром Ln, где n может

принимать значения от 1 до 64 и обозначает долю полной ноты, т.е.

L1 обозначает полную ноту, L2 обозначает половинную, L4 - чет-

вертную ноту и т.д.

 Пауза задается параметром Pn (n - длина паузы, измеряется в

длительностях, установленных параметром L).

 Темп задается параметром Tn, где n - число четвертных нот в

минуту, может принимать значения от 32 до 255 (по умолчанию n =

120). Параметры MS, MN или ML определяют долю времени звучания

каждой ноты от длительности, определенной параметром L:

 MS - в 3/4 этой длительности (стаккато),

 MN - в 7/8 (обычное, нормальное звучание),

 ML - полное время (легато).

 Режим звучания может быть  \_основным ., когда следующий оператор

программы не будет выполняться, пока оператор PLAY не завершен

(параметр MF), или  \_фоновым ., когда выполнение оператора PLAY не

препятствует выполнению остальных операторов программы (параметр

MB). Например,

 PLAY "L8 MF MS O3 GG L4 AG O4 C O3 L2 B"

 Отдельные музыкальные отрезки можно описать предварительно в

виде символьных строк и использовать их затем в операторе PLAY.

Например,

 - 30 -

 A$="EDE" : B$ = "CDE" : PLAY A$ + B$

 4.6. Функция INKEY$

 Функция INKEY$ действует аналогично оператору INPUT, однако

при вводе на экране не высвечивается вводимый символ и не требу-

ется нажатие клавиши "Ввод". Программа при этом не прерывает ра-

боту, как при выполнении оператора INPUT. При выполнении операто-

ра

 X$ = INKEY$

символ, соответствующий нажатой клавише, присваивается перемен-

ной X$.

 Рассмотрим в качестве примера программу:

 PRINT "Задайте направление: 1-север, 2-юг,\_

 3-запад, 4-восток"

 aa:

 D$ = INKEY$

 IF D$ = "" THEN aa

 ON VAL (D$) GOTO nn, ww, ee, ss

 PRINT "Неправильная клавиша" : GOTO aa

 nn:

 ...

 ww:

 ...

 Здесь, пока не будет нажата какая-либо клавиша, 3-й и 4-й

операторы выполняются в цикле, обеспечивая режим ожидания. Когда

клавиша "1", "2", "3", или "4" нажата, соответствующее значение

присваивается переменной D$. Далее в операторе ON осуществляется

ее преобразование к числовому значению и переход к указанной мет-

ке для соответствующей обработки.

 - 31 -

 5. БИБЛИОТЕКА СТАНДАРТНЫХ ПРОЦЕДУР В ТУРБО-БЕЙСИКЕ

1. Решение систем линейных алгебраических уравнений (SIMQ.BAS)

2. Интерполирование функций (ALI.BAS)

3. Обращение матрицы и вычисление определителя (MINV.BAS)

4. Умножение матриц (GMPRD.BAS)

5. Вычисление корня уравнения (RTMI.BAS)

6. Решение систем нелинейных уравнений (SNEN.BAS)

7. Вычисление корней полинома (POLRT.BAS)

8. Вычисление производной (DCAR.BAS)

9. Вычисление интеграла (QATR.BAS)

10. Экстремум функции (FMCG.BAS)

11. Решение систем обыкновенных дифференциальных

 уравнений (RKGS.BAS)

12. Сглаживание функций (SG13.BAS)

13. Аппроксимация функций (APFS.BAS)

14. Упорядочивание значений (используется при

 интерполировании функций) (ATSG.BAS)

15. Вычисление полиномов Чебышева (используется

 при аппроксимации функций) (APCH.BAS)

Подключение библиотечной процедуры осуществляется оператором

 $INCLUDE " 1имя библиотечной процедуры 0"

Пример использования библиотечной процедуры QATR.BAS

 $INCLUDE "QATR.BAS"

 CLS

 PRINT "ВЫЧИСЛЕНИЕ ИНТЕГРАЛА"

 CALL QATR(0, 1, .0001, 24, INTEGRAL, IER%)

 PRINT "ИНТЕГРАЛ =";

 PRINT USING "###.####"; INTEGRAL

 PRINT "КОД ОШИБКИ ="; IER%

 DEF FNFCTN (X) = LOG ( 1 + X) / (1 + X \* X)

 END

 - 32 -

 6. РЕКОМЕНДУЕМЫЙ ПОРЯДОК РАБОТЫ НА ПЭВМ ПРИ СОЗДАНИИ

 И ВЫПОЛНЕНИИ ПРОГРАММ НА ЯЗЫКЕ ТУРБО-БЕЙСИК

 1. Войти в Турбо-Бейсик. Для этого набрать tb в командной

строке и нажать клавишу "Ввод". Еще раз нажать "Ввод".

 В верхней строке экрана высвечивается главное меню.

 2. Для первоначального ввода или исправления программы войти

в редактор (пункт меню EDIT).

 3. После того, как программа составлена (или внесены исправ-

ления), ее необходимо записать в память на диск или дискету

(пункт главного меню FILE, подпункт SAVE или клавиша F2).

 Замечание. Рекомендуется после набора каждых 10 строк запи-

сывать программу на диск (простым нажатием клавиши F2), чтобы в

случае сбоя ПЭВМ программа не пропала.

 4. Выполнить программу (пункт RUN главного меню), переход на

большой экран для анализа результатов - команда ALT/F5.

 5. Выйти из Турбо-Бейсика (пункт FILE, подпункт EXIT или ко-

манда ALT/X).

 Замечание. В каждом режиме в нижней строке экрана указано

назначение функциональных клавиш. Например, если курсор находится

в главном меню, то нажатие клавиши F1 (Help) вызывает появление

на экране сведений по использованию соответствующего режима, пе-

реход к следующей странице - "Ввод". Нажатие F5 (Zoom) вызывает

раскрытие текущего окна на весь экран, повторное нажатие F5 -

возврат в исходное положение; F6 (Next) - переход к следующему

окну; F7 (Goto) - переход в режим редактирования; ALT/X (Exit) -

выход из Турбо-Бейсика.

 В режиме редактирования: F1 - Help (см. выше), F2 - Save (за-

пись на диск или дискету); F3 - New (загрузка новой программы,

имя вводится в появившееся окно) и т.д.

 Галина Ивановна Светозарова

 ОСНОВЫ ПРИКЛАДНОЙ ИНФОРМАТИКИ

 Раздел: Описание языка Турбо-Бейсик

 Справочное пособие

 ──────────────────────────────────────────────────────────────

 Подписано в печать

 Уч.-изд.л. 1,5 Тираж 1000 экз.

 Заказ Цена

 ──────────────────────────────────────────────────────────────

 Московский институт стали и сплавов, Ленинский проспект, 4

 Типография ЭОЗ МИСиС, Орджоникидзе, 8/9

\_