Министерство образования и науки Республики Казахстан

Павлодарский государственный университет им. С. Торайгырова

Кафедра Информатики и информационных систем

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам

по дисциплине «Алгоритмизация и программирование»

для студентов специальности 050716 - Приборостроение

Павлодар

УТВЕРЖДАЮ
Декан ФФМиИТ
_____С.К.Тлеукенов
«__»_____20__ г.

Составитель**:** преподаватель Нургазина Б.К.

Кафедра Информатика и информационные системы

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам

по дисциплине «Алгоритмизация и программирование»

для студентов специальности 050716 - Приборостроение

Рекомендовано на заседании кафедры от «__»_____20__ г. Протокол №____.

Заведующий кафедрой _____ Нурбекова Ж.К.
(подпись)

Одобрена методическим советом факультета ФМиИТ
«___»_____20__г. Протокол №_____

Председатель МС_____ А.З.Даутова
(подпись)

Лабораторная работа №1

Тема: Арифметические выражения. Линейное программирование.

Программирование линейных алгоритмов. Отладка и тестирование программ

В рабочем окне редактора интегрированной среды наберем текст первой программы.

```
Program Myl_l ;
Var a, b, rez : Integer;
Begin
   WriteLn ('Введите два числа через пробел');
   ReadLn (a, b);
   rez :=a*b;
   WriteLn ('Их произведение равно ', rez);
   WriteLn ('Нажмите <Enter>');
   ReadLn
End.
```

Краткий разбор примера. Структура программы

Программа начинается с заголовка, который имеет следующий вид:

Program <имя программы>;

За заголовком идет раздел описаний, в котором должны быть описаны все идентификаторы (константы, переменные, типы, процедуры, функции, метки), которые используются в программе. После раздела описаний идет раздел операторов, который начинается со служебного слова Begin и заканчивается служебным словом End. В этом разделе задаются действия над объектами программы, введенными в разделе описаний. Операторы, посредством которых эти действия производятся, разделяются точкой с запятой. После последнего слова End ставится точка.

Для того чтобы сохранить программу, необходимо:

1 выйти в главное меню и выбрать режим File;

2 нажать <Enter>, в вертикальном меню выбрать пункт Save as… и нажать клавишу <Enter>;

3 в появившемся окне ввести имя файла (например, a:\prim1_1.pas) и нажать клавишу <Enter>. В нашем примере файл с именем PRIM1_1.PAS сохраняется в корневом каталоге диска A; PAS — расширение, сообщающее о том, что файл содержит программу, написанную на языке Паскаль.

Поскольку мы работаем в режиме DOS, имя файла не может содержать более 8 символов.

Примечания.

1. В именах файлов нельзя употреблять следующие символы: *, =, +, [,], \, |, :, ., <, >, /, ?, символ пробела и буквы русского алфавита.

2. Для быстрого сохранения файла можно воспользоваться командами Save <F2> или Save all меню File.

Для того чтобы закончить работу, необходимо:

1 выйти в главное меню и выбрать пункт File;

2 нажать <Enter> и в вертикальном меню выбрать пункт Exit, после чего нажать либо <Enter>, либо просто нажать комбинацию клавиш <Alt> + <X>.

Эксперименты с программой

1. Введите в качестве исходных данных достаточно большие числа, например, 4567 и 789. Убедитесь, что у вас получается неправдоподобный результат — отрицательное число (—1117). Найдите экспериментальным путем тот интервал значений переменных a и b, когда результат умножения правильный.

2. Вместо числа введите какой-нибудь символ. Убедитесь, что компьютер

выдает сообщение об ошибке "Error 106: Invalid numeric format".

3. Добавьте лишний знак апострофа в операторе WriteLn. Убедитесь, что

программа не проходит компиляцию, а система сообщает об ошибке "Error 8: String constant exceeds line".

4. Измените оператор

rez :=a*b на rez:=a-(a Div b)*b

Выясните результат операции Div над переменными целого типа. Измените текстовую часть следующего за оператором присваивания оператора WriteLn, отразив в ней результат вашего исследования.

5. Добавьте в программу переменную с именем ost, оператор присваивания ost:=a Mod b; и оператор вывода WriteLn (' ???????? ', ost). Выясните, какая операция обозначается словом Mod. Замените знаки вопроса вашими пояснениями.

6. Наберите следующую программу:

```
Program My1_2;
Var a : Integer;
Begin
     WriteLn ('Введите целое число');
     ReadLn (a) ;
     WriteLn ( ' ???????? ', Abs (a) );
     WriteLn ( ' Нажмите <Enter> ' );
     ReadLn
End.
```

Выясните, что вычисляет Abs. Выполните аналогичное исследование для выражений `sqr (a), Ord (a) , Succ (a) , pred (a).`

Примечание.

Рекомендуется следующий порядок работы: текст программы My1_1 сохраняется под новым именем, например, PRlM1_2.PAS, а затем изменяется и вновь сохраняется. Это позволяет сократить время на набор программы.

Задания для самостоятельной работы

1. Измените программу таким образом, чтобы в ней находилась сумма двух чисел.

2. Измените программу таким образом, чтобы в ней находилась сумма четырех чисел.

3. Напишите программу для вычисления выражения: (a + (d - 12) *3) * (c - 5*k), значения переменных a, d, c и k вводятся с клавиатуры.

Лабораторная работа №2

Тема: Условный оператор

Полные и неполные формы операторов ветвления и выбора

Условный оператор позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить то или иное действие. Таким образом, условный оператор - это средство ветвления вычислительного процесса.

Структура условного оператора имеет следующий вид:

IF <условие> THEN <оператор1> ELSE <оператор2>,

где IF, THEN, ELSE - зарезервированные слова (если, то, иначе); <условие> - произвольное выражение логического типа; <оператор1>, <оператор2> - любые операторы языка Турбо Паскаль.

Условный оператор работает по следующему алгоритму. Вначале вычисляется условное выражение <условие>. Если результат есть TRUE (истина), то выполняется <оператор1>, а <оператор2> пропускается; если результат есть FALSE (ложь), наоборот, <оператор1> пропускается, а выполняется <оператор2>. Часть ELSE <оператор2> условного

оператора может быть опущена. Тогда при значении TRUE условного выражения выполняется <оператор1>, в противном случае этот оператор пропускается.

Оператор выбора позволяет выбрать одно из нескольких возможных продолжений программы. Параметром, по которому осуществляется выбор, служит ключ выбора -выражение любого порядкового типа (любого из рассмотренных, кроме типов REAL и STRING).

Структура оператора выбора такова:

CASE <ключ_выбора> OF <список_выбора> [ELSE <операторы>] END

Здесь CASE, OF, ELSE, END - зарезервированные слова (случай, из, иначе, конец);

<ключ_выбора> - ключ выбора;

<список_выбора> - одна или более конструкций вида:

<константа_выбора> : <оператор>;

<константа_выбора> - константа того же типа, что и выражение<ключ_выбора> ;

<операторы> - произвольные операторы Турбо Паскаля.

Оператор выбора работает следующим образом. Вначале вычисляется значение выражения <ключ_выбора>, а затем в последовательности операторов <список_выбора> отыскивается такой, которому предшествует константа, равная вычисленному значению. Найденный оператор выполняется, после чего оператор выбора завершает свою работу. Если в списке выбора не будет найдена константа, соответствующая вычисленному значению ключа выбора, управление передается операторам, стоящим за словом ELSE. Часть ELSE <оператор> можно опускать. Тогда при отсутствии в списке выбора нужной константы ничего не произойдет и оператор выбора просто завершит свою работу. Любому из операторов списка выбора может предшествовать не одна, а несколько констант выбора, разделенных запятыми.

```
Program Hex;
{Программа считывает введенное с клавиатуры целое число в диапазоне от 0 до 15, преобразует его к шестнадцатеричной системе счисления и выводит результат на экран}
var
  n : Integer; {Вводимое число}
  ch : Char; {Результат}
begin
  Write ( ',n = ' ) ;
   ReadLn(n); {Вводим число. Проверяем число на принадлежность к диапазону 0...15}
  if (n >= 0) and (n <= 15) then
  begin {Да, принадлежит диапазону}
    if n < 10 then
      ch := chr(ord('0') + n)
    else
      ch := chr(ord('A') + n- 10);
    WriteLn('n = ',ch)
  end
  else {Не принадлежит диапазону}
    WriteLn('Ошибка')
end.
```

В шестнадцатеричной системе счисления используется 16 цифр в каждом разряде: цифры 0...9 обозначают первые 10 возможных значений разряда, буквы A...F - остальные шесть.

В программе учитывается непрерывность и упорядоченность множеств цифр 0...9,

букв A...F и их кодов.

Следующая программа при вводе одного из символов: y или Y выведет на экран слово «Да», а при вводе n или N - слово «Нет»:

```
var
ch : Char ;
begin
ReadLn (ch) ;
case ch of
'n','N' : WriteLn ('Нет' );
'y','Y' : WriteLn ('Да')
end
end.
```

Задания для самостоятельной работы

1. Пользователь вводит 3 числа. Программа выводит наибольшее из них.

2. Написать программу, которая находит корни квадратного уравнения с заданными коэффициентами A, B, C

3. Пользователь вводит натуральное однозначное число (цифру). Программа должна в ответ печатать название цифры в текстовом видею

Лабораторная работа №3

Тема: Операторы цикла

Алгоритмы арифметики, вычисление многочленов. Использование перечисляемых типов в организации циклов с параметром

Счетный оператор цикла FOR имеет такую структуру:

FOR <пар_цик> := <нач_знач> TO <кон_знач> DO <оператор>.

Здесь FOR, TO, DO - зарезервированные слова (для, до, выполнить);

<пар_цик> - параметр цикла - переменная типа INTEGER (точнее, любого порядкового типа, см. гл.4);

<нач_знач> - начальное значение - выражение того же типа;

<кон_знач> - конечное значение - выражение того же типа;

<оператор> - произвольный оператор Турбо Паскаля.

При выполнении оператора FOR вначале вычисляется выражение <нач_знач> и осуществляется присваивание <пар_цик> : = <нач_знач>. После этого циклически повторяется:

1 проверка условия <пар_цик> <= <кон_знач>; если условие не выполнено, оператор FOR завершает свою работу;
2 выполнение оператора <оператор>;
3 наращивание переменной <пар_цик> на единицу.

Отметим два обстоятельства. Во-первых, условие, управляющее работой оператора FOR, проверяется перед выполнением оператора <оператор>: если условие не выполняется в самом начале работы оператора FOR, исполняемый оператор не будет выполнен ни разу. Другое обстоятельство - шаг наращивания параметра цикла строго постоянен и равен (+1). Существует другая форма оператора:

FOR<пар_цик>: = <нач_знач> DOWNTO <кон_знач> DO <оператор>

Замена зарезервированного слова TO на DOWNTO означает, что шаг наращивания параметра цикла равен (-1), а управляющее условие приобретает вид <пар_цик> = <кон_знач>.

```
Program Summ_of_Integer;
```

```
{Программа вводит целое положительное число N и подсчитывает сумму всех целых чисел от 1 до N}
var
i, n, s : Integer;
begin
Write('N = ');
ReadLn(n); . {Вводим N}
s := 0; {Начальное значение суммы}
for i : = 1 to n do {Цикл подсчета суммы}
s : = s + i;
writeln('Сумма = ',s) {Выводим результат}
end.
```

Задания для самостоятельной работы

1. Подсчитать сумму кубов нечетных чисел от 1 до 10.
2. Подсчитать сумму всех чисел, которые делятся на 3 без остатка в интервале от 1 до 100
3. Составить программу, которая выводит таблицу степеней числа 2 от 1-й до 16-й.

Лабораторная работа №4-5

Тема: Одномерные массивы

Одномерные и двумерные массивы. Алгоритмы ввода массива

Массивы представляют собой ограниченную упорядоченную совокупность однотипных величин. Каждая отдельная величина называется компонентой массива. Тип компонент может быть любым, принятым в языке ПАСКАЛЬ, кроме файлового типа. Тип компонент называется базовым типом.

Вся совокупность компонент определяется одним именем. Для обозначения отдельных компонент используется конструкция, называемая переменной с индексом или с индексами:

A[5] S[k+1] B[3,5].

В качестве индекса может быть использовано выражение. Тип индексов может быть только интервальным или перечислимым. Действительный и целый типы недопустимы. Индексы интервального типа, для которого базовым является целый тип, могут принимать отрицательные, нулевое и положительные значения.{} В операторной части программы один массив может быть присвоен другому, если их типы идентичны, например:

R1:=Z.

Для ввода или вывода массива в список ввода или вывода помещается переменная с индексом, а операторы ввода или вывода выполняются в цикле. Первый индекс определяет номер строки, второй - номер столбца. Двумерные массивы хранятся в памяти ЭВМ по строкам. Инициализация массивов (присвоение начальных значений всем компонентам массивов) осуществляется двумя способами.

Первый способ - с использованием типизованных констант, например:

```
type Dim10= Array[1..10] of Real;
const
 raM10: Dim10 = ( 0, 2.1, 4, 5.65, 6.1, 6.7, 7.2, 8, 8.7, 9.3 );
```

При инициализации двумерных массивов значения компонент каждого из входящих в него одномерных массивов записывается в скобках:

```
type Dim3x2= Array[1..3,1..2] of Integer;
const
 iaM3x2: Dim3x2= ( (1, 2),
```

```
            (3, 4),
            (5, 6) );
```
Второй способ инициализации - использование процедуры FillChar, которая заполняет заданное число следующих друг за другом байт указанным значением.

FillChar(var V; NBytes: Word; B: Byte);

Эта процедура заполняет участок памяти однобайтовым значением. Например, для обнуления массива A[1..10] of Real можно записать:

FillChar(A, 40, 0);

или

FillChar(A, SizeOf(A), 0);

Второй способ предпочтительнее, т.к. при помощи функции SizeOf Паскаль определяет размер массива в байтах самостоятельно.

Пример 1:

Следующая программа напечатает 10 раз число 257. Почему?

```
Program ExDim1;
Var Dim10:Array[1..10] of integer;
    I:integer;
Begin
     FillChar(Dim10,SizeOf(Dim10),1);
     for i:=1 to 10 do Writeln(Dim10[I]);
End.
```

Пример 2:

```
 Program ExDim2;
 Type Dim5x5=Array[1..5,1..5] of Byte;
 Const Diag:Dim5x5=( (1,0,0,0,0),
                     (0,1,0,0,0),
                     (0,0,1,0,0),
                     (0,0,0,1,0),
                     (0,0,0,0,1) );
 Var Dim:Dim5x5;
     I,J:integer;
 procedure PrintDim;
 Begin
      for i:=1 to 5 do begin
          for J:=1 to 5 do Write(Dim[i,j],' ');
          Writeln;
      end;
      Writeln('_____________________________');
 end;
 Begin
      FillChar(Dim,SizeOf(Dim),0);
      PrintDim;
      Dim:=Diag;PrintDim;
      J:=5;
       for i:=1 to 5 do begin
            Dim[i,j]:=1;Dec(J);
      end;
      PrintDim;
      Readln;
```

```
  End.
```

Пример 3. Вычислить сумму элементов числового массива A = (a1 , a2 , ... , aN).

```
Program Summa;
  Uses Crt;
  Type Mas = Array [1..20] of Real;
  Var A    : Mas;
      i, N : Integer;
      S    : Real;
BEGIN
  ClrScr;    {очистка экрана }
  Write('Введите N = ');
  ReadLn(N); {ввод значения N}
  For i := 1 to N do {цикл по элементам массива}
    begin
      Write('A [ ', i , ' ] = ');
      ReadLn(A[i])   {ввод элементов массива}
    end;
  S := 0; {присваивание начального значения}
  For i := 1 to N do S := S+A[i]; {суммирование}
  WriteLn;
  WriteLn('Сумма равна ', S : 5 : 1);
  ReadLn
END.
```

Задания для самостоятельной работы

1. Подсчитать число и сумму положительных, число и произведение отрицательных элементов заданного массива A(N).

2. Элементы заданного массива B(N) переписать в новый массив A(N) в обратном порядке

3. Вычислить сумму квадратов всех элементов заданного массива X(N), за исключением элементов, кратных пяти.

Сортировка массивов

Алгоритмы поиска максимального (минимального). Алгоритмы сортировки массивов: сортировка выборкой, простым обменом и др.

Попробуем решить нашу задачу, поставленную в параграфе 2.7.1. Для начала нам надо описать массив чисел, который вы будете вводить с клавиатуры. Как мы говорили числа целые и их не более 40. Выделим память для хранения переменных: счетчика, конкретного количества чисел которое вы будете вводить и значения минимального элемента.

```
   Program    Primer_8; {программа нахождение минимального по величине числа в массиве}
   var
   Chislo :array [1..40] of integer;
   i,n, resultat: integer;
```

Использование массива позволяет упростить текст программы для ввода данных

```
   procedure init;
   begin
```

```
        write ('input n -');
        readln (n);  {ввод количества чисел}
        for i:=1 to n do
           begin
                write ('input Chislo[',i,']=');
                readln (Chislo[i]);
           end;
   end;
```

Обратите внимание на процедуру write('input Chislo[',i,']=') , использование такой конструкции позволяет выводить не только приглашение ввода нового элемента, но и за счет изменения значения счетчика помогает не запутаться с вводом значения конкретного элемента.

После ввода исходных данных, приступим к основной работе: поиску элемента с минимальным значением. Для этого воспользуемся искусственным приемом: Будем считать, что первый элемент и является элементом с минимальным значением. Действительно, если в массиве нет других элементов, то так оно и есть. В переменную с именем resultat поместим значение первого элемента. Если элементов в массиве несколько сравним значение переменной resultat со значением второго элемента массива. Если значение второго элемента меньше значения переменной resultat , то поместим в переменную resultat значение второго элемента, иначе оставим значение resultat без изменения. Будем повторять эти действия, пока не просмотрим все элементы массива:

```
     procedure work;
     begin
      resultat:= Chislo[1];
      for i:=2 to n do
              if Chislo[i]< resultat then
                          resultat:= Chislo[i];
     end;
```

Нам остается вывести результат на экран, в этом затруднений, наверное, не будет. Текст модуля такой же, как и для программы Primer_7. Текст тела программы так же не изменился.

Для решения многих задач необходимо упорядочить данные по определенному признаку. Этот процесс называют сортировкой. Для простоты изложения мы будем рассматривать одномерный массив из целых чисел:

```
Const NMax=... ;
Type MyArray=Array[l..NMax] Of Integer;
Var A:MyArray;
```

Суть большинства алгоритмов сортировки от такого упрощения не изменяется. Алгоритмы сортировки отличаются друг от друга степенью эффективности, под которой понимается количество сравнений и количество обменов, произведенных в процессе сортировки. В основном мы будем оценивать эффективность количеством операций сравнения (порядком этого значения). Заметим, что элементы массива можно сортировать:

• по возрастанию — каждый следующий элемент больше предыдущего — A [1] < A [2] < ... < A[N];

• по неубыванию — каждый следующий элемент не меньше предыдущего A[1] <= A [2] <= ... <= A[N];

• по убыванию — каждый следующий элемент меньше предыдущего A[1] > A [2] > ... > A[N];

• по невозрастанию — каждый следующий элемент не больше предыдущего A [1] >= A [2]

>= ... >= A [N].

Научившись выполнять одну сортировку, изменить ее, чтобы получить другую, не составляет особого труда.

Рассмотрим идею метода простым выбором на примере. Пусть исходный массив состоит из 10 элементов:

5 13 7 9 1 8 16 4 10 2

После сортировки массив должен выглядеть так:

1 2 4 5 7 8 9 10 13 16

Процесс сортировки представлен ниже. Максимальный элемент текущей части массива заключен в кружок, а элемент, с которым происходит обмен, — в квадратик. Скобкой помечена рассматриваемая часть массива, т.е. еще не упорядоченная.

1-й шаг. Рассмотрим весь массив и найдем в нем максимальный элемент — 16 (стоит на седьмом месте), поменяем его местами с последним элементом — с числом 2.

Максимальный элемент помещен на свое место.

2-й шаг. Рассмотрим часть массива — с первого до девятого элемента. Максимальный элемент этой части — 13, стоящий на втором месте. Поменяем его местами с последним элементом неупорядоченной части — с числом 10.

Отсортированная часть массива состоит теперь уже из двух элементов.

3-й шаг. Уменьшим рассматриваемую часть массива на один элемент. Здесь нужно поменять местами второй элемент (его значение — 10) и последний элемент этой части — число 4.

В отсортированной части массива — 3 элемента.

4-й шаг.

5-й шаг. Максимальный элемент этой части массива является последним в ней, поэтому его нужно оставить на месте.

Фрагмент программной реализации:

```
Procedure Sort;
  Var i, j, k: Integer;
     m:Integer; {Значение максимального элемента рассматриваемой части
массива.}
  Begin
      For i:=N DownTo 2 Go Begin
      {Цикл по длине рассматриваемой части массива.}
      {Поиск максимального элемента и его номера в текущей части массива.}
      k: =i; m: =A [ i ] ;
      {Начальные значения максимального элемента и его индекса в
рассматриваемой части массива.}
      For j:=l To i-1 Do If A[j]>m Then
         Begin :k:=j; m:=A[k] End;
      If k<>i Then
     {Перестановка элементов.)
       Begin A[k] :=A[i],;A[i] :=m;End;
                              End;
End;
```

Оценим количество сравнений. При первом просмотре N — 1, при втором — N — 2, при последнем — 1. Общее количество C =

N — 1+N — 2 + ...+ 1= N• (N- l)/2, или C=O(N2).

Задания для самостоятельной работы

1. Что надо изменить в предыдущей программе текст для того, что бы программа находила максимальное по значению число?

2. Напишите программу, которая не только находит минимальное по значению число, но и определяет номер (индекс) этого числа.

Варианты:

№	**Задание (для таблицы из 5 строк и 4 столбцов)**
1.	Подсчитать количество нулей в таблице и заменить на это значение все нечетные целые элементы таблицы
2.	Поменять местами максимальный и минимальный элементы таблицы, если сумма значений таблицы равна нулю
3.	Найти сумму элементов, расположенных перед максимальным элементом таблицы, и заменить на это значение все нулевые элементы таблицы
4.	Подсчитать количество отрицательных элементов в таблице и увеличить на это значение минимальный и максимальный элементы таблицы
5.	Определить, имеется ли в таблице хотя бы один нулевой элемент. Если такой элемент есть, то заменить все вещественные значения таблицы единицами
6.	Если максимальный элемент в таблице расположен после минимального, то поменять эти элементы местами
7.	Подсчитать количество четных целых элементов, расположенных перед максимальным элементом таблицы и увеличить на это значение максимальный элемент
8.	Если после максимального элемента таблицы расположена хотя бы одна единица, то увеличить все положительные элементы таблицы в два раза
9.	Если в таблице сумма значений равна нулю, то уменьшить максимальный элемент таблицы в два раза
10.	Если перед максимальным элементом таблицы расположены все единицы, то заменить максимальный элемент таблицы на количество этих единиц
11.	Если сумма элементов первой строки равна максимальному элементу таблицы, то увеличить этот максимальный элемент в два раза
12.	Если во втором столбце стоят две единицы, то уменьшить максимальный элемент первой строки в два раза.

Лабораторная работа №6

Тема: Двумерные массивы

Напишите программу, в которой двумерный массив вводится с клавиатуры и выводится в виде матрицы

Напишите программу, которая находит минимальное по значению число в двухмерном массиве и определяет номер (индекс) этого числа. Максимальная размерность массива [1..10,1..10].

Напишите программу, которая выводит на экран суммы элементов каждого столбца двумерного массива.

Напишите программу, проверяющую, является ли данная квадратная матрица магическим квадратом (магической называется матрица, у которой суммы элементов в каждом столбце, каждой строке и по каждой диагонали одинаковы).

Лабораторная работа №7

Тема: Процедуры и функции

Процедуры с параметрами. Формальные и фактические параметры. Вызов процедур. Передача параметров. Вызов функций.

Описание подпрограммы состоит из заголовка и тела подпрограммы.

Заголовок процедуры имеет вид:

PROCEDURE <имя> [(<сп. ф. п . >)] ;

Заголовок функции:

FUNCTION <имя> [(<сп.ф.п.>)] : <тил>;

Здесь <имя> - имя подпрограммы (правильный идентификатор);

<сп.ф.п.> - список формальных параметров;

<тип> - тип возвращаемого функцией результата.

Список формальных параметров необязателен и может отсутствовать. Если же он есть, то в нем должны быть перечислены имена формальных параметров и их типы, например:

```
Procedure SB(a: Real; b: Integer; c: Char);
```

Как видно из примера, параметры в списке отделяются друг от друга точками с запятой. Несколько следующих подряд однотипных параметров можно объединять в подсписки, например, вместо

```
Function F(a: Real; b: Real): Real;
```

можно написать проще:

```
Function F(a,b: Real): Real;
```

Операторы тела подпрограммы рассматривают список формальных параметров как своеобразное расширение раздела описаний: все переменные из этого списка могут использоваться в любых выражениях внутри подпрограммы. Таким способом осуществляется настройка алгоритма подпрограммы на конкретную задачу.

Рассмотрим следующий пример. В языке Турбо Паскаль нет операции возведения в степень, однако с помощью встроенных функций LN(X) и EXP(X) нетрудно реализовать новую функцию с именем, например, POWER, осуществляющую возведение любого вещественного числа в любую вещественную степень. В программе вводится пара чисел X и Y и выводится на экран дисплея результат возведения X сначала в степень +Y, а затем - в степень -Y. Для выхода из программы нужно ввести Ctrl-Z и Enter.

```
var
x,y:Real;
Function Power (a, b : Real):
Real;
begin {Power}
if a > 0 then
Power := exp(b * In (a))
else if a < 0 then
Power := exp(b * ln(abs(a))
else if b = 0 then
Power := 1
else
Power := 0
end {Power} ;
{--------------------}
begin {main}
repeat
```

```
readln(x,y) ;
writeln (Power (x,y) :12:10, Power (x, -y) : 15 : 10)
until EOF
end {main} .
```

Для вызова функции POWER мы просто указали ее в качестве параметра при обращении к встроенной процедуре WRITELN. Параметры X и Y в момент обращения к функции - это фактические параметры. Они подставляются вместо формальных параметров A и B в заголовке функции и затем над ними осуществляются нужные действия. Полученный результат присваивается идентификатору функции - именно он и будет возвращен как значение функции при выходе из нее. В программе функция POWER вызывается дважды - сначала с параметрами X и Y, а затем X и -Y, поэтому будут получены два разных результата.

Механизм замены формальных параметров на фактические позволяет нужным образом настроить алгоритм, реализованный в подпрограмме. Турбо Паскаль следит за тем, чтобы количество и тип формальных параметров строго соответствовали количеству и типам фактических параметров в момент обращения к подпрограмме. Смысл используемых фактических параметров зависит от того, в каком порядке они перечислены при вызове подпрограммы. В примере первый по порядку фактический параметр будет возводиться в степень, задаваемую вторым параметром, а не наоборот. Пользователь должен сам следить за правильным порядком перечисления фактических параметров при обращении к подпрограмме.

Любой из формальных параметров подпрограммы может быть либо параметром-значением, либо параметром-переменной, либо, наконец, параметром-константой.

В предыдущем примере параметры A и B определены как параметры-значения. Если параметры определяются как параметры-переменные, перед ними необходимо ставить зарезервированное слово VAR, а если это параметры-константы,- слово CONST, например:

Procedure MyProcedure (var a: Real; b: Real; const c: String);

Здесь A - параметр-переменная, B -параметр-значение, а C - параметр-константа.

Определение формального параметра тем или иным способом существенно, в основном, только для вызывающей программы: если формальный параметр объявлен как параметр-переменная, то при вызове подпрограммы ему должен соответствовать фактический параметр в виде переменной нужного типа; если формальный параметр объявлен как параметр-значение или параметр-константа, то при вызове ему может соответствовать произвольное выражение. Контроль за неукоснительным соблюдением этого правила осуществляется компилятором Турбо Паскаля. Если бы для предыдущего примера был использован такой заголовок функции:

```
Function Power (var a, b : Real) : Real;
```

то при втором обращении к функции компилятор указал бы на несоответствие типа фактических и формальных параметров (параметр -Уесть выражение, в то время как соответствующий ему формальный параметр описан как параметр-переменная).

Для того чтобы понять, в каких случаях использовать тот или иной тип параметров, рассмотрим, как осуществляется замена формальных параметров на фактические в момент обращения к подпрограмме.

Если параметр определен как параметр-значение, то перед вызовом подпрограммы это значение вычисляется, полученный результат копируется во временную память и передается подпрограмме. Важно учесть, что даже если в качестве фактического параметра указано простейшее выражение в виде переменной или константы, все равно подпрограмме будет передана лишь копия переменной (константы). Любые возможные изменения в подпрограмме параметра-значения никак не воспринимаются вызывающей программой, так как в этом случае изменяется копия фактического параметра.

Если параметр определен как параметр-переменная, то при вызове подпрограммы передается сама переменная, а не ее копия (фактически в этом случае подпрограмме передается адрес переменной). Изменение параметра-переменной приводит к изменению самого фактического параметра в вызывающей программе.

В случае параметра-константы в подпрограмму также передается адрес области памяти, в которой располагается переменная или вычисленное значение. Однако компилятор блокирует любые присваивания параметру-константе нового значения в теле подпрограммы.

Существует еще одно обстоятельство, которое следует учитывать при выборе вида формальных параметров. Как уже говорилось, при объявлении параметра-значения осуществляется копирование фактического параметра во временную память. Если этим параметром будет массив большой размерности, то существенные затраты времени и памяти на копирование при многократных обращениях к подпрограмме можно минимизировать, объявив этот параметр параметром-константой. Параметр-константа не копируется во временную область памяти, что сокращает затраты времени на вызов подпрограммы, однако любые его изменения в теле подпрограммы невозможны - за этим строго следит компилятор.

```
{Программа перевода десятичного числа в двоичное }
var a : longint;
function DEC_BIN(x:longint):string;
const digits:array [0..1] of char = ('0','1');
var res:string; d:0..1;
begin
  res:='';
  while (x<>0) do begin
    d:=x mod 2; res:=digits[d]+res;
    x:=x div 2;
  end;
  DEC_BIN:=res;
end;
begin { основная программа }
  readln( a );
  writeln( DEC_BIN(a) );
end.
```

Задания для самостоятельной работы

1. Написать подпрограмму, переводящую двоичное число в десятичное
2. Написать подпрограмму, вычисляющего третью сторону треугольника по двум другим сторонам и углу между ними.
3. Написать подпрограмму, которая выясняет, существует ли файл с указанным именем.

Лабораторная работа №8

Тема: Обработка символьных и строковых данных

Символьные и строковые данные

Алгоритмы обработки строк. Алгоритмы поиска подстроки в строке. Встроенные функции и процедуры для работы со строками.

При сравнении двух строковых переменных меньшей считается та символьная переменная, текст которой стоит раньше, если бы эти строковые переменные были упорядочены в алфавитном порядке. Например:

'abc' <'abd' 'abc' <'abca' '12'<'4'

Строковые переменные можно складывать. Например:

Stroka1:='ffff';

Stroka2:='aaaa';

Stroka3:= Stroka1+ Stroka2;

В результате значением переменной Stroka3 будет значение - 'ffffaaaa'.

В среде BP для работы со строковыми переменными имеются готовые процедуры и функции. Познакомимся с некоторыми из них:

Функция **length** (s:String) : Integer;

Возвращает длину строки. В скобках должно находится переменная строкового типа, результат будет целое число. Пример:

Dlina:= length (Stroka3);

В этом случае значение переменной с именем " Dlina " будет равно 6.

Функция **Concat**(s1,s2,..,sN: String): String;

Соединяет последовательно строки s1,s2,..,sN.Пример:

Stroka3:= Concat(Stroka1, Stroka2);

Результат будет такой, если бы мы сложили эти две строки.

Функция **Copy**(s:String,Index:Integer,Count:Integer): String;

Выделяет из строки s подстроку длиной Count символов, начиная с позиции Index. Пример:

Stroka4:= Copy(Stroka3,3,2);

В результате строковая переменная Stroka4 примет значение 'fa'.

Функция pos (sub, s:String): byte;

Параметры Sub и S являются выражениями строкового типа. Данная функция ищет подстроку, заданную параметром Sub, в строке S и возвращает значение типа byte, являющееся позицией первого символа подстроки в строке. Если подстрока не найдена, то функция возвращает значение 0. Пример:

Pos1:=pos('asd','hgfasdrtyasd');

Pos2:=pos('asd1','hgfasdrtyasd');

Обратите внимание, что подстрока 'asd' встречается дважды в строке 'hgfasdrtyasd', но в результате переменная с именем Pos1 примет значение 4, а не 10. Таким образом, эта функция определяет номер позиции первого символа найденной в строке 'hgfasdrtyasd' подстроки 'asd'. Значение Pos2 будет равно 0, так как данной подстроки нет в искомой строке.

Процедура **Delete**(var s:String; Index: Integer; Count: Integer);

Данная процедура удаляет из строки s число символов, соответствующее параметру Count, начиная с символа, номер которого задан параметром Index. После действия данной процедуры измененная строка будет храниться под старым именем. Пример:

Stroka:='asdfghjkl';

Delete(Stroka; 4; 3);

Значение переменной с именем Stroka примет значение - ' asdkl'.

Задания для самостоятельной работы

1. Напишите программу, которая находит количество вхождений подстроки Sub в строку Stroka. Исходная строка и подстрока вводится с клавиатуры. Программу сохраните в файле "Primer10.pas".

2. Напишите программу, которая считывает текст исходной строки с клавиатуры и выводит ее на экран в обратном порядке. Программу сохраните в файле "Primer11.pas".

3. Используя свойство строковой переменной как массива символов, напишите программу, которая находит количество вхождений подстроки Sub в строку Stroka. Исходная строка и подстрока вводится с клавиатуры. Программу сохраните в файле "Primer14.pas".

Лабораторная работа №9

Тема: Записи и множества

Сложные типы данных

Операции над множествами, использование множеств. Комбинированные типы данных – записи. Оператор присоединения

Структурный тип, характеризуемый методом структурирования и типами своих компонентов, имеет более одного значения. Если тип компонента является структурным, то получаемый в результате структурный тип имеет более одного уровня структурирования. Структурный тип может иметь неограниченные уровни структурирования.

```
Стуктурный тип [packed]  Тип массив | Тип запись | Тип объект | Тип множество | Тип файл
```

Слово packed (упакованный) в описании структурного типа требует от компилятора уплотнить хранимые данные, даже за счет уменьшения скорости доступа к компоненту в переменной этого типа. Слово packed не имеет никакого действия в Borland Pascal, поскольку упаковка выполняется здесь автоматически всюду, где это возможно.

Тип запись

Тип запись содержит установленное число элементов или полей, которые могут быть различных типов. Описание типа запись указывает тип каждого поля и идентификатор, который именует поле.

```
Тип запись = Record
  [Список полей]
End
Список полей=
 Фиксированная часть;
 [Вариантная часть;]
```

Фиксированная часть типа запись содержит список фиксированных полей вместе с идентификатором и типом для каждого поля. Каждое поле содержит информацию, которая всегда отыскивается одним и тем же способом.

```
Фиксированная часть = Список идентификаторов: Тип;
```

Приведем пример типа запись:

```
record
  year:  integer;          { год }
  month: 1..12;            { месяц }
  day:   1..31;            { число }
end
```

В вариантной части, изображенной на синтаксической диаграмме описания типа запись, память распределяется более чем для одного списка полей, поэтому доступ к информации может быть осуществлен более чем одним способом. Каждый список полей является вариантом. Варианты налагаются друг на друга в памяти, поэтому в любое время возможен доступ ко всем полям во всех вариантах.

```
Вариантная часть
Case Тип определителя поля Of Вариант
Идентификатор : [Тип определителя поля: идентификатор регулярного типа]
Вариант = Константа : (Список полей);
End;
```

Из диаграммы следует, что каждый вариант идентифицирован по крайней мере одной константой. Все константы должны быть отличными друг от друга и иметь порядковый тип, совместимый с типом поля признака. Доступ к вариантным и фиксированным полям один и тот же.

В вариантной части можно указать необязательный идентификатор - идентификатор признака поля. При наличии идентификатора признака поля он становится идентификатором дополнительного фиксированного поля записи - поля признака. Программа может использовать значение поля признака для указания, какой вариант является активным в настоящий момент. Без указания поля признака программа выбирает вариант по другому критерию.

Ниже приводятся несколько примеров типов запись:

```
    record
      firstName,lastName : string[40];
      birthDate : Date;
      case citizen : boolean of
        True  : (birthPlace: string[40]);
        False : (country   : string[20];
                 entryPort : string[20];
                 entryDate : Date;
                 exitDate  : Date);
    end;
record
  x,y : real;
  case kind: Figure of
   rectangle: (height,wigth: real);   {прямоугольник}
   triangle : (size1,side2,angle: real);{треугольник}
   circle   : (radius: real);         { круг }
end
```

Задания для практической работы

1. Спишите запись АНКЕТА и поместите в неё следующую информацию:

Ф.И.О. (фамилия, имя, отчество), адрес (улица, номер дома, номер квартиры), пол, возраст. Определите, сколько лиц женского и сколько лиц мужского пола проживают по одной улице.

2. Воспользовавшись записью из задачи 1, определить сколько лиц женского и сколько мужского пола проживает в одном доме.

3. Воспользовавшись записью АНКЕТА из задачи 1, определить сколько лиц мужского пола в возрасте старше 18 лет и младше 60 проживают по одной улице.

Лабораторная работа №10

Тема: Работа с различными видами файлов

Работа с текстовыми файлами

Процедуры и функции для обработки текстовых файлов. Ввод-вывод данных

Информация в текстовых (последовательных) файлах хранится в виде последовательности символов. Символы образуют строки произвольной длины. В конце каждой строки записываются 2 особых символа, имеющие коды #13 (перевод строки) и #10 (возврат каретки), которые отделяют данную строку от следующей.

В программе на ВР текстовый файл представлен файловой переменной типа **Text**:

Var имя файловой переменной : Text;

Связь файловой переменной с дисковым именем файла осуществляется с помощью оператора:

Assign(файловая переменная, имя файла)

Открытие файла для чтения выполняется оператором:

Reset(файловая переменная)

Открытие файла для записи выполняется оператором:

Rewrite(файловая переменная)

Открытие файла для дополнения (дозаписи) выполняется оператором:

Append(файловая переменная)

Закрытие файла выполняется оператором:

Close(файловая переменная)

Упражнение 1:

Выясните у преподавателя местонахождение текстового файла Ewords.txt, содержащего около 2000 английских слов (каждое слово в отдельной строке). Просмотрите этот файл при помощи программы «Блокнот» (или программмы FAR) и затем составьте следующую программу:

```
program Pr_1;
{Подсчитать к-во слов в файле Ewords.txt}
Var F:Text;S:string;NW:Integer;
Begin
     Assign(F,'C:\BP\Temp\Ewords.txt');
     Reset(F); NW:=0;
     Repeat
         Readln(f,S);
         if S<>'' then Inc(NW);
     until EOF(F);
     if S<>'' then Inc(NW);
     Close(F);
     writeln('К-во слов в файле Ewords.txt=',NW);
     Readln;
END.
```

Примечание:

Функция EOF возвращает значение True, если достигнут конец файла.

Составим программу Pr9_5, которая запрашивает у пользователя букву латинского алфавита (A..Z), а затем создает текстовый файл с именем [буква].txt, в который помещает слова из файла Ewords.txt, начинающиеся на эту букву. (Например, пользователь ввел букву ‘F’. Надо создать файл с именем F.txt.)

Разберем текст этой программы:

```
program Pr_2;
uses Crt;
{Запросить букву A..Z и затем создать}
{текстовый файл с именем [буква].txt}
{который должен содержать все слова из Ewords.txt}
{начинающиеся на эту букву. Вывести к-во записанных слов}
Var FromF,ToF:Text;S:string;NW:Integer;CH:Char;NameTo:string;
Begin ClrScr;
Write('Введите букву A..Z: ');Readln(CH);CH:=UpCase(CH);
if (CH<’A’) or (CH>’Z’) then begin
```

```
   writeln('Необходимо ввести букву латинского алфавита!');
   readln;             Halt(0);
end;
NameTo:='C:\BP\Temp\'+CH+'.txt';
     Assign(FromF,'C:\BP\Temp\Ewords.txt');
     Reset(FromF); NW:=0;          Assign(ToF,NameTo);
     Rewrite(ToF);
     Repeat
         Readln(FromF,S);
         if UpCase(S[1])=CH then begin
           Inc(NW);Writeln(ToF,S);
         end;
     Until EOF(FromF);
     Close(FromF);Close(ToF);
     writeln('В файл ',NameTo,' записано ',NW,' слов.');
     Readln;
END.
```

В начале работы запрашивается символ латинского алфавита и затем он переводится в верхний регистр при помощи функции UpCase. После этого производится проверка, содержится ли в действительности в переменной CH символ из диапазона A..Z. Если это не так, то выводится сообщение и программа завершает работу.

В противном случае в переменной NameTo формируется имя текстового файла для записи.

После этого файл Ewords.txt открывается для чтения, а файл с именем, содержащимся в переменной NameTo – для записи.

Затем начинается цикл чтения файла Ewords.txt. Очередная строка из этого файла попадает в переменную S. Если первый символ в строке S совпадает по верхнему регистру с символом, содержащимся в переменной CH, то при помощи оператора Writeln(ToF,S) происходит запись значения переменной S в очередную строку файла для записи.

По достижении конца файла Ewords.txt цикл завершается и оба файла закрываются.

Задания для самостоятельной работы

1. Составить программу, которая подсчитывает в файле Ewords.txt количество слов, начинающихся и заканчивающихся на одну и ту же букву. Программа должна выводить на экран найденные слова.

2. Составить программу, которая выводит на экран из файла Ewords.txt те слова, которые читаются одинаково справа налево и слева направо.

3. Составить программу, которая запрашивает у пользователя букву латинского алфавита (A..Z), а затем подсчитывает и выводит на экран все слова, начинающиеся на эту букву.

4. Составить программу, которая заносит в одномерный массив количество слов в файле Ewords.txt, начинающихся с определенной буквы латинского алфавита и затем выводит статистику на экран.

5. Составить программу, которая находит строку наибольшей длины в файле Ewords.txt, выводит ее на экран и печатает также количество символов в ней.

6. Составить программу, которая создает файл WordsE.txt, в котором содержатся все слова из файла Ewords.txt в обратном порядке (первым будет слово ZERO, а последним – слово ABROAD). При этом все слова в файле WordsE.txt должны быть записаны прописными буквами.

Типизированные файлы

Процедуры и функции для обработки типизированных файлов. Ввод-вывод данных

Файловый тип состоит из линейной последовательности компонентов, которые могут иметь любой тип за исключением файлового типа или структурного типа, содержащего компонент с файловым типом. Число компонентов описанием файлового типа не устанавливается.

```
файловый тип:
file of тип;
```

Если слово of и тип компонента опущены, то тип обозначает нетипизированный файл. Нетипизированные файлы представляют собой каналы ввода-вывода нижнего уровня, в основном используемые для прямого доступа к любому файлу на диске, независимо от его внутреннего формата.

Функция	Описание
Assign	Присваивает имя внешнего файла файловой переменной.
BlockRead	Считывает из нетипизированного файла одну или более записей.
BlockWrite	Записывает в нетипизированный файл одну или более записей.
ChDir	Выполняет смену текущего каталога
Close	Закрывает открытый файл.
Erase	Стирает внешний файл.
Eof	Возвращает для файла состояние end-of-file (конец файла).
FilePos	Возвращает текущую позицию в файле. Для текстовых файлов не используется.
FileSize	Возвращает текущий размер файла. Для текстовых файлов не используется
Flush	Сбрасывает буфер текстового файла вывода.
Getdir	Возвращает текущий каталог на заданном диске.
IOResult	Возвращает целое значение, являющееся состоянием последней выполненной операции ввода-вывода
MkDir	Создает подкаталог.
Read	Считывает одно или более значений из файла в одну или более переменных.
Rename	Переименовывает внешний файл
Reset	Открывает существующий файл.
Rewrite	Создает и открывает новый файл.
RмDir	Удаляет пустой подкаталог.
Seek	Перемещает текущую позицию в файле на заданный элемент. Для текстовых файлов не используется.
SeekEof	Возвращает для текстового файла состояние "конец файла"
SeekEoln	Возвращает для текстового файла состояние "конец строки".
SetTextBuf	Назначает для текстового файла буфер ввода- вывода.
Truncate	Усекает размер файла до текущей позиции. Для текстовых файлов не используется
Write	Записывает в файл одно или более значений.

Задания для самостоятельной работы: Создать соответственно своему варианту базу

данных. Сформировать требуемые запросы. Программа должна иметь простейшее меню из следующих пунктов:

1 Ввод информации
1 Получение ответов на запросы (несколько пунктов)
1 Выход из программы".

Программа должна быть красочно оформлена.

Варианты заданий

1.

№	Фио абонента	дата	автор	Издательство	название	Год изд.	цена

Запросы:

- сколько абонентов пользовалось данной книгой,
- сколько абонентов поставлено на учет данного числа,
- получить список абонентов, на руках у которых книга стоимости не ниже заданной.

2.

№	Название театра	дата	время	Название спектакля	Автор	режиссер	цена

Запросы:

- получить список театров, в которых идут спектакли с указанным названием,
- получить список режиссеров, поставивших заданное количество спектаклей,
- получить название театров и спектаклей, которые идут данного числа.

3.

№	Магазин	отдел	Наименование продукции	Количество единиц	Цена за единицу

Запросы:

- получить список магазинов, продающих продукцию данного наименования,
- получить список отделов данного магазина,
- получить название магазинов, имеющих минимальную цену заданного продукта.

Лабораторная работа №11

Тема: Реализация графики на языке Паскаль

Константы модуля Graph.

Константы цвета

Black = 0; {Черный} Blue = 1; {Синий} Green = 2; {Зеленый} Cyan = 3; {Голубой}
Red = 4; {Красный} Magenta = 5; {Фиолетовый} Brown = 6; {Коричневый}
LightGray = 7; {Светлосерый} DarkGray = 8; {Темносерый} LightBlue = 9; {Яркосиний}
LightGreen = 10; {Яркозеленый} LightCyan = 11; {Яркоголубой} LightRed = 12; {Розовый}
LightMagenta = 13; {Малиновый} Yellow = 14; {Желтый} White = 15; {Белый}

Константы типов и толщины линий

SolidLn = 0; {Сплошная} DottedLn = 1; {Точечная} CenterLn = 2; {Штрихпунктирная}
DashedLn = 3; {Пунктирная} NormWidth=1; {Нормальная толщина}
ThickWidth = 3; {Тройная толщина}

Константы шаблона штриховки

EmptyFill = 0; SolidFill = 1; LineFill = 2;

LtSlashFill = 3; SlashFill = 4; BkSlashFill = 5;
LtBkSlashFill = 6; HatchFill = 7; XHatchFill = 8; InterleaveFill = 9; WideDotFill = 10;
CloseDotFill = 11; UserFill = 12.
{Заполнение цветом фона}
{Сплошная штриховка}
{Горизонтальная штриховка}
{/// штриховка}
{/// штриховка толстыми линиями}
{\\\ штриховка толстыми линиями}
{\\\ штриховка}
{Заполнение прямой клеткой}
{Заполнение косой клеткой}
{Заполнение частой сеткой}
{Заполнение редкими точками}
{Заполнение частыми точками}
{Тип задается пользователем}

Процедуры модуля Graph

Arc(X, Y: Integer; U1, U2, R: Word)

Строит дугу окружности текущим цветом с текущими параметрами линии. X, Y - координаты центра дуги, U1 - угол до начальной точки дуги, отсчитываемый против часовой стрелки от горизонтальной оси, направленной слева направо, U2 - угол до конечной точки дуги, отсчитываемый так же, как U1, R - радиус дуги.

Bar(X1, Y1, X2, Y2: Integer)

Строит прямоугольник, закрашенный текущим цветом с использованием текущего стиля (орнамента, штриховки). X1, Y1, X2, Y2 - координаты левого верхнего и правого нижнего углов прямоугольника.

Bar3D(X1, Y1, X2, Y2: Integer; Glubina: Word; Top: Boolean)

Строит параллелепипед, используя текущий стиль и цвет. X1, Y1, X2, Y2 - координаты левого верхнего и правого нижнего углов передней грани; Glubina - ширина боковой грани (отсчитывается по горизонтали), Top - признак включения верхней грани (если True - верхняя грань вычерчивается, False - не вычерчивается).

Circle(X, Y: Integer; R: Word)

Рисует текущим цветом окружность радиуса R с центром в точке (X,Y).

ClearDevice

Очищает графический экран, закрашивает его в цвет фона.

ClearViewPort

Очищает выделенное графическое окно, закрашивает его в цвет фона.

CloseGraph

Закрывает графический режим, т.е. освобождает память, распределенную под драйверы графики и файлы шрифтов, и восстанавливает текстовый режим работы экрана.

Ellipse(X, Y: Integer; U1, U2, XR, YR: Word)

Рисует дугу эллипса текущим цветом; X, Y - координаты центра эллипса; U1, U2 - углы до начальной и конечной точек дуги эллипса (см. процедуру Arc); XR, YR - горизонтальная и вертикальная полуоси эллипса.

FillEllipse(X, Y: Integer; XR, YR: Word)

Рисует заштрихованный эллипс, используя X,Y как центр и XR,YR как горизонтальную и вертикальную полуоси эллипса.

FillPoly(N: Word; Var PolyPoints)

Рисует и штрихует многоугольник, содержащий N вершин с координатами в PolyPoints.

InitGraph(Var Driver, Mode: Integer; Path: String)
Организует переход в графический режим. Переменные Driver и Mode содержат тип графического драйвера и его режим работы. Третий параметр определяет маршрут поиска графического драйвера. Если строка пустая (т.е. равна "), считается, что драйвер находится в текущем каталоге.
Line(X1, Y1, X2, Y2: Integer)
Рисует линию от точки X1, Y1 до точки X2,Y2.
LineTo(X, Y: Integer)
Рисует линию от текущего указателя к точке X,Y.
MoveTo(X, Y: Integer)
Смещает текущий указатель к точке X,Y.
OutTextXY(X, Y: Integer; TextString: String)
Выводит текст в заданное место экрана.
PieSlice(X, Y: Integer; U1, U2, Radius: Word)
Строит сектор круга, закрашенный текущей штриховкой и цветом заполнения. X, Y - координаты центра сектора круга; U1 и U2 - начальный и конечный углы сектора, отсчитываемые против часовой стрелки от горизонтальной оси, направленной вправо; Radius - радиус сектора.
PutPixel(X, Y: Integer; Color: Word)
Выводит точку цветом Color с координатами X, Y.
Rectangle(X1, Y1, X2, Y2)
Рисует контур прямоугольника, используя текущий цвет и тип линии. X1, Y1 - координаты левого верхнего угла прямоугольника,
X2, Y2 - координаты правого нижнего угла прямоугольника.
Sector(X, Y: Integer; U1, U2, XR, YR: Word)
Рисует и штрихует сектор эллипса радиусами XR, YR с центром в X, Y от начального угла U1 к конечному углу U2.
SetBkColor(Color: Word)
Устанавливает цвет фона. SetColor(Color: Word)
Устанавливает основной цвет, которым будет осуществляться рисование.
SetFillStyle(Pattern, Color: Word)
Устанавливает образец штриховки и цвет. SetLineStyle(LineStile, Pattern, Thickness: Word)
Устанавливает толщину и стиль линии.
SetTextStyle(Font, Direction, CharSize: Word)
Устанавливает текущий шрифт, направление (горизонтальное или вертикальное) и размер текста.
SetViewPort(X1, Y1, X2, Y2: Integer; ClipOn: Boolean)
Устанавливает прямоугольное окно на графическом экране. Параметр ClipOn определяет "отсечку" элементов изображения, не умещающихся в окне.
Функции
GetMaxX и GetMaxY
Возвращает значения максимальных координат экрана в текущем режиме работы, соответственно, по горизонтали и вертикали.
GraphResult
Возвращает значение GrOk, соответствующее коду 0, если все графические операции программы выполнились без ошибок, или возвращает числовой код ошибки (от -1 до -14).
Составим следующую небольшую программу и запустим ее на выполнение:
program Pr_graph;

```
uses Crt,Graph;
var
  grDriver: Integer;    grMode: Integer;   ErrCode: Integer;    MaxX,MaxY:integer;
begin
  grDriver := Detect;    InitGraph(grDriver, grMode,'C:\Bp\Bin');
  ErrCode := GraphResult;   if ErrCode = grOk then
  begin  { Do graphics }
    MaxX:=GetMaxX;MaxY:=GetMaxY;
    Line(0, 0, GetMaxX, GetMaxY);
    Readln;      CloseGraph;
    writeln('MaxX=',MaxX,'MaxY=',MaxY);      readln;
  end
  else
    Writeln('Graphics error:', GraphErrorMsg(ErrCode));
end.
```

Нетрудно заметить, что после запуска этой программы компьютер переходит в графический режим и в этом режиме рисуется линия белого цвета от верхнего левого угла к правому нижнему.

Задания для самостоятельной работы

Напишите программу, которая рисует на экране изображение электрической схемы, включающей различные электрические приборы.

Задание.