Основы языка программирования Turbo Pascal

Среда Turbo Pascal
1. Структура программы.
2. Данные.
3. Типы данных.
4. Оператор присваивания.
5. Операторы вывода данных.
6. Операторы ввода данных.
7. Условный оператор. (Программы разветвляющейся структуры)
8. Оператор выбора (варианта).
9. Цикл с предусловием.
10. Цикл с постусловием Repeat.
11. Цикл с параметром.
12. Процедуры и функции
13. Массивы
14. Работа с величинами строкового типа в паскале
15. Текстовые Файлы (очень краткая справка)
16. Модуль CRT
17. Модуль Graph

Среда Turbo Pascal

Среда Турбо-Паскаля – первое, с чем сталкивается любой программист, приступающий к работе с системой. Термином "среда Турбо-Паскаля" будем обозначать разнообразные сервисные средства, которые не относятся непосредственно к языку программирования, но служат для облегчения процесса разработки, отладки и испытания программ.

Для вызова Турбо-Паскаля необходимо запустить файл: ***TP\BIN\turbo.exe.***

После успешного вызова системы верхняя строка экрана содержит меню возможных режимов работы Турбо-Паскаля, нижняя строка – краткую справку о назначении основных функциональных клавишах. Вся остальная часть экрана принадлежит так называемому окну редактора. Это окно используется для ввода и редактирования текста программы. Верхняя строка редактора (вторая сверху) содержит служебную информацию текстового редактора: номер строки и позиция в ней, на которой в данный момент находится курсор, имя редактируемого файла и др.

Кроме окна редактора в Турбо-Паскале используются также окно отладочного режима и окно вывода результатов выполнения программы.

Все управление средой осуществляется в основном с помощью системы последовательно разворачивающихся меню (верхняя строка редактора). Меню фиксирует некоторое текущее состояние диалоговой среды и предлагает несколько альтернативных путей перехода из этого состояния. Каждое конкретное меню реализуется в виде небольшого окна с текстом, которое как бы накладывается на существующее на экране отображение. Содержащиеся в меню альтернативы условимся называть опциями. Таким образом, меню образует древовидную структуру.

Рассмотрим, в общем, некоторые основные опции среды:

File – содержит опции работы с дисками и файлами (загрузить программу, записать программу, сменить каталог, выход из программы и т.д.)

Run – содержит опции по выполнению и трассировки программ.

Compile – содержит опции компилирования программ и др.

и т.д.

Значение некоторых клавиш:

F10	- выход в верхнее меню;
Alt + выделенная буква	- выбор конкретного пункта меню;
Ctrl + F9	- выполнение программы;
Alt + F9	- компиляция программы;
Alt + F5	- посмотреть результаты выполнения программы.
Delete	- удалить символ под курсором;
Backspase	- удалить символ слева от курсора;
Ctrl + K , B	- пометить начало блока;
Ctrl + K , K	- пометить конец блока;
Ctrl + K , C	- копировать выделенный блок;
Ctrl + K , V	- переместить выделенный блок;
Ctrl + K , Y	- удалить выделенный блок.

1. Структура программы.

Текст программы представляет собой последовательность строк и состоит из заголовка, раздела описаний, раздела операторов:

Program Имя ;

Раздел описаний:

const *констант*

var *переменных*

procedure *процедур*

function *функций*

и др.

Begin

Раздел операторов:
оператор 1;
оператор 2;
...
оператор n

End.

I. *Заголовок программы*

II. *Раздел описаний* предназначен для объявления всех встречающихся в программе данных (имена переменных с указанием их типов, постоянные со значением и т.д.)

III. *Раздел операторов* состоит из последовательности команд - операторов.
Друг от друга операторы отделяются точкой с запятой.

2. Данные.

Все величины, используемые в программе можно разделить на две группы:

1) ***постоянные*** - значения которых не меняются в процессе ее выполнения;
   описываются в разделе описаний «const»:
   const *Имя = значение*;
2) ***переменные*** - величины, значения которых могут изменяться в процессе выполнения программы.
   описываются в разделе описаний «var»:
   var *Имя : Тип*;

Все используемые в программе переменные должны быть определены с указанием их типов.
Тип переменных определяют в зависимости от значений, которые принимает эта переменная (см. таблицу «Типы данных»).

3. Типы данных.

Любая константа, переменная, значение функции или выражения в Турбо-Паскале характеризуется своим типом.

Тип определяет множество допустимых значений, которые может иметь объект (константа, переменная и т.д.), а также множество допустимых операций, которые применимы к объекту.

Таблица 1. **Типы данных.**

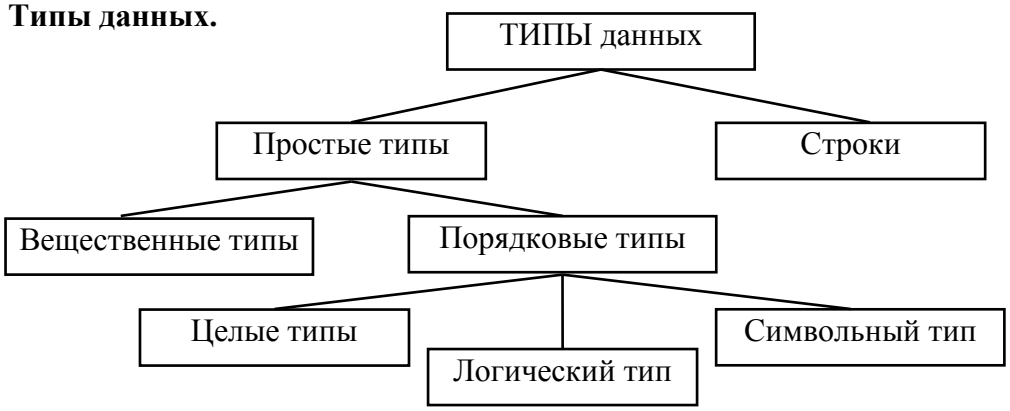

Таблица 2. **Характеристики типов данных.**

Тип	Допустимые значения	Размер
Ц е л ы е		
Shortint	- 128 . . 127	1 байт
Integer	- 32768..32767	2 байта
Longint	- 2147483648..2147483647	4 байта
Byte	0 . . 255	1 байт
Word	0 . . 65535	2 байта
В е щ е с т в е н н ы е		
Real	2.9e-39..1.7e38	6 байт
Single	1.5e-45..3.4e38	4 байта
Double	5.0e-324..1.7e308	8 байт
С и м в о л ь н ы й		
Char	один символ	1 байт
С т р о к и		
String	строка длиной до 255 символов	1-255 байт
Л о г и ч е с к и й		
Boolean	False , True	1бит

Наиболее распространенные типы данных:

1) Целый тип (Integer) – данные, которые принимают только целые значения.

Например, 5; 0; -5 и т.д.

2) Вещественный тип (Real) – все числа (целые + дробные).

Например, 5,2; -5,2; 5; -5; 0; 1/2; -1/2 и т.д.

Следует помнить, что в программировании десятичная запятая заменяется десятичной точкой, то есть не 2,5 - а 2.5

3) Символьный тип (Char) - данные, значения которых какой-то один символов, заключенный в апострофы.

Например, 'А' , '1', 'ф' , и т.д.

4) Строковый тип (String) - данные, значения которых представляют из себя набор символов, заключенный в апострофы.

Например, 'Ответ' , 'Привет' и т.д.

5) Логический тип (Boolean) - данные, которые могут принимать одно из значений True (истина) или False (ложь).

4. Оператор присваивания.

Для задания значений переменной можно использовать оператор присваивания:

имя переменной := *выражение*;

Например,

a:= 5; b:= 7; c:= (a + b) * 2;

Правила записи арифметических выражений:

1) выражение записывается в одну строку;

например, $\frac{1}{12}+\frac{4}{a+c}-c^2$ записывается: 1/12 + 4/(a+c) – c*c

2) в выражении можно использовать только круглые скобки;
3) нельзя записывать подряд два знака операций:

неправильно: 3*a*-2 правильно: 3*a*(-2)

4) нельзя пропускать знак умножения:

неправильно: 2a+4c правильно: 2*a+4*c

5) операции выполняются слева направо в соответствии со старшинством операций, изменить порядок выполнения можно используя скобки;
6) если в выражении используются переменные, то они должны получить значения до первого своего использования.

Таблица 3. **Операции.**

Операция	Приоритет	Действие	Тип операндов	Тип значения
not	1	отрицание	логический	логический
*	2	умножение	числовой	тип операндов
/	2	деление	числовой	вещественный
div	2	целочисленное деление	целый	целый
mod	2	остаток от деления	целый	целый
and	2	логическое и	логический	логический
+	3	сложение	числовой,строковый	тип операндов
-	3	вычитание	числовой	тип операндов
or	3	логическое или	логический	логический
xor	3	исключительное или	логический	логический
=	4	равно	любой *	логический
<>	4	не равно	любой	логический
<=	4	меньше, равно	любой	логический
<	4	меньше	любой	логический
>	4	больше	любой	логический
>=	4	больше, равно	любой	логический

* любой из известных: числовой, логический, символьный, строковый.

Таблица 4. **Некоторые стандартные Функции.**

Обращение	Тип параметра	Тип результата	Реализуемое действие
abs(x)	числовой	тип аргумента	модуль аргумента
sqrt(x)	числовой	вещественный	корень квадратный
sqr(x)	числовой	тип аргумента	квадрат аргумента
round(x)	числовой	целый	округляет аргумент до целых
int(x)	числовой	вещественный	целая часть числа
trunc(x)	числовой	целый	целая часть аргумента
frac(x)	числовой	вещественный	дробная часть числа
odd(x)	целый	логический	true, если число нечетное и false, если число четное
Random		вещественный	псевдослучайное число $0 \le i < 1$
Random(x)	целый	целый	псевдослучайное целое число $0 \le i < x$
Randomize			инициация датчика псевдослучайных чисел
pi		вещественный	$\pi = 3{,}141592653...$
sin(x)	числовой	вещественный	синус аргумента (угол в радианах)
cos(x)	числовой	вещественный	косинус аргумента (угол в радианах)
ArcTan(x)	числовой	вещественный	арктангенс

5. Операторы вывода данных.

1) write (*список вывода*);

где список вывода - это набор переменных или строк символов(заключенных в апострофы), разделенных запятыми.

Например,

write ('Ответ: S=', S);

2) writeln (*список вывода*);

осуществляет сначала вывод на экран дисплея списка вывода, а затем курсор переходит на новую строку.

3) writeln;

переход курсора на новую строку.

4) write(*имя переменной :n :m*);

или

write(*имя переменной :n*);

вывод чисел по формату:

n - количество символов в изображении всего значения переменной;

m - количество символов после десятичной точки (записывается только для вывода вещественных - real - значений).

Например,
по операторам

```
x:=5.987;
write('x=',x:5:1);
```

будет напечатано

```
x=  5.9
```

6. Операторы ввода данных.

1) **read (*список ввода*);**

где список ввода - это набор переменных, разделенных запятыми.
Например,

```
read( A, B );
```

2) **readln (*список ввода*);**

осуществляет сначала ввод с клавиатуры, а затем курсор переходит на новую строку.
Значения переменных, указанных в списке ввода задаются с клавиатуры после запуска программы на выполнение. Компьютер, встречая данный оператор, приостанавливает работу и ждет как будут введены все значения. Значения вводятся через пробел и заканчиваются нажатием клавиши Enter.
Например,

```
writeln ('Введите значения A, B, C');
readln(A,B,C);
```

7. Условный оператор. (Программы разветвляющейся структуры)

Существует два варианта записи оператора **If**:

1. Полное ветвление:

If *Условие* **then** *Оператор1* **else** *Оператор2*;

2. Неполное ветвление:

If *Условие* **then** *Оператор1*;

Где, **If** (если), **then** (тогда), **else** (иначе) - служебные слова;
Оператор1, Оператор2 - любой из операторов:

1. Простой оператор.
   Например, if a>=0 then a:=sqrt(a) else a:=sqrt(abs(a));
2. Пустой оператор - оператор, который не выполняет никаких действий.
   Например, if a>=0 then a:=sqrt(a) else ;
   После else стоит пустой оператор. В этом случае служебное слово else можно не писать.
   Например, if a>=0 then a:=sqrt(a);
3. Составной оператор - набор операторов, заключенных в операторные скобки begin end.
   Например, if a>=0 then a:=sqrt(a) else begin a:=abs(a); a:=sqrt(a) end;
4. Еще один условный оператор.
   Например, if a>0 then if a>4 then b:=sqrt(a) else b:=a*a else b:= a div 2;
   В этом случае действует правило: первое else относится к ближайшему предшествующему ему if.

Условие:

1. Простое - логическая переменная, логическая функция или логическое выражение.
   В логических выражениях применяют знаки отношений: < (меньше), <=(меньше или равно), = (равно), > (больше), >= (больше или равно), <> (не равно).
   Например, x > 5, x + y <= z, odd(a), a * 2 = 6.
2. Составное - состоит из простых условий, связанных между собой логическими операциями *and* (и), *or* (или), *not* (не), *xor* (исключительное или). Каждое простое условие в составном заключается в скобки.
   Например,

```
if (a>9) and (a<100) then write( 'a - двузначное число' )
                     else  write( 'a - не двузначное число' );
```

   Составные условия рассматриваются как единое целое.
   Правила определения истинности даны в табл. 5.

8. Оператор выбора (варианта).

Общий вид:

Case *Выражение* **of**
Константа1: Оператор1;
Константа2: Оператор2;
...
КонстантаN: ОператорN
else *Оператор*
end;

В этом операторе:
Case (случай), **of** (из), **else** (иначе), **end** (конец) - служебные слова;
Выражение - выражение или переменная порядкового типа;
Константа1, Константа2,..., КонстантаN - константы, с которыми сравнивается значение выражения, они должны быть того же типа, что и *Выражение* ;
Оператор1, Оператор2, ...,ОператорN - операторы (простые или составные), из которых выполняется тот, с константой которого совпадает значение выражения;
Оператор - оператор, который выполняется, если среди констант нет вычисленного значения выражения.

Недопустимо, чтобы одна и та же константа встречалась в списке выбора более одного раза.

В случае, когда для разных констант необходимо выполнить один и тот же оператор, их можно объединить в группу, перечислив через запятую или указав диапазон значений.

Else в операторе *case* может отсутствовать.

9. Цикл с предусловием.

Общий вид:

While *Условие* **do** *Оператор*;

В этом операторе:
While (пока), **do** (делай) - служебные слова.
Условие - условие входа в цикл: цикл **While** выполняется до тех пор, пока условие истинно и прекращается после того, как условие станет ложным.
Оператор - простой или составной оператор.
Например,

```
x:=0;
P:=1;
while x<N do
begin
   x:=x+1;
  P:=P*x;
end;
```

Данная часть программы находит N! (факториал числа N, т.е. 1*2*3*…*N).

При выполнении цикла **While** условие цикла проверяется **перед** каждым исполнением тела цикла. Если условие ложно с самого начала, то тело цикла не выполнится ни разу.
Например,

```
x:=10;
while x<10 do
begin
   x:=x+1; write(x:5)
end;
```

10. Цикл с постусловием Repeat.

Общий вид:

Repeat
Оператор1;
Оператор2;
...
until *Условие;*

В этом операторе:
Repeat (повторяй), **until** (до) - служебные слова;
Условие - условие выхода из цикла: цикл **Repeat** выполняется до тех пор, пока условие ложно, как только оно становится истинным, действие цикла прекращается;
Оператор1, Оператор2, ... - простые, составные или структурированные операторы.

Применяя цикл **Repeat** необходимо помнить, что условие проверяется **после** каждого выполнения тела цикла. Поэтому тело цикла всегда выполнится хотя бы один раз.
Например,

```
S:=0;
x:=0;
repeat
  x:=x+2;
  S:=S+x;
until x=10;
  write('Сумма=', S)
```

Данная часть программы находит сумму целых чисел от 2 до 10, а затем выводит ее на экран.

11. Цикл с параметром.

Существует два варианта цикла с параметром:
1. **For** *Параметр*:=*Начальное значение* **to** *Конечное значение* **do**
Оператор;
2. **For** *Параметр*:=*Начальное значение* **downto** *Конечное значение* **do**
Оператор;
В этих операторах:
For (для), **to** (до), **downto** (обратно к), **do** (выполнить) - служебные слова;
Параметр цикла - переменная порядкового типа;
Начальное и *Конечное значения* параметра цикла - выражение того же типа, что и *Параметр*. Они вычисляются один раз перед первой попыткой выполнить тело цикла и не меняются в цикле.
Оператор - простой или составной оператор.

Для цикла **For...to...do...** последовательность значений параметра цикла следующая: начальное значение, непосредственно следующее за ним и так далее до конечного значения включительно.
Например,

```
readln(N)
P:=1;
for x:=1 to N do  P:=P*x;
```

Данная часть программы находит N! (факториал числа N, т.е. 1*2*3*…*N, значение N вводится с клавиатуры).

Для цикла **For...downto...do...** последовательность значений параметра цикла следующая: начальное значение, непосредственно идущее перед ним, и так далее, до конечного значения включительно.
Например,

```
for x:=10 downto 1 do  write(x:5);
```

Данная часть программы напечатает последовательность целых чисел от 10 до 1 включительно.

12. Процедуры и функции

Процедура – это относительно независимая часть программы, оформленная особым образом и снабженная оригинальным именем, которое позволяет вызывать ее из программы при необходимости.

Процедуры полезно использовать в двух случаях:

- тогда, когда одни и те же действия необходимо выполнить несколько раз в разных местах программы, возможно при различных исходных данных;
- в процедуру полезно выделять и однократно выполняемые действия, когда они представляют собой логически независимую часть программы.

Описание процедуры:

Procedure *Имя* (*список формальных параметров***);**
Раздел описаний
Begin
Раздел операторов
End;

Примечание. Процедуры описываются в разделе описаний основной программы или в другой процедуре.

Список формальных параметров может отсутствовать.

В разделе описаний процедуры/функции может находиться все, что и в основной программе: описание переменных, констант, типов, других процедур и функций. Он может и отсутствовать.

Вызов процедуры:

Имя* (фактические параметры***);**

Примечание. Процедуры вызываются из основной программы или из других процедур.

Список ***формальных параметров*** – аргументов процедуры/функции - это список, используемых в процедуре переменных, описанных в заголовке процедуры/функции с указанием типа, значения которых передаются из места вызова процедуры.

Например, Procedure Paint (x,y,cp,cl: integer);

При вызове процедуры в качестве аргументов задаются конкретные значения формальных параметров – это ***фактические параметры***.

Т.е. ***фактические параметры*** – это те исходные данные, которые передаются в формальные параметры при каждом обращении к ней.

Например, Paint (100,100,1,15);

Очевидно, что между формальными и фактическими параметрами должно быть соответствие:

- по количеству параметров,
- по типу параметров,
- по порядку следования.

В том случае, если одно и то же вычисление приходится выполнять несколько раз, имеет смысл создать функцию, которую производила бы эти вычисления и выдавала результат.

Функции пользователя представляют собой последовательность операторов, в результате выполнения которых вычисляется одно значение, которое затем присваивается имени функции.

Описание функции:

Function *Имя_функции* (*список формальных параметров***): *тип результата*;**
Раздел описаний
Begin
Раздел операторов;
Имя_функции*:=Результат;***
End;

Вызов функции:

Переменная:= ***Имя_функции*** (*фактические параметры*);

Все переменные/именованные константы программы относительно данной процедуры/функции делятся на два вида: ***локальные*** – описанные внутри процедуры или в заголовке и ***глобальные*** – описанные вне ее.

Примечание.

Важное отличие локальных переменных от глобальных состоит в том, что они видны (известны их значения) только внутри своей процедуры/функции и не видны в других процедурах и основной программе.

Однако из процедуры видны все глобальные объекты. Однако, если локальные переменные имеют те же названия, что и глобальные, то глобальные переменные не будут видны внутри процедуры/функции.

Формальные параметры делятся на два вида:

Параметры-Значения	***Параметры-Переменные***
Если параметры определяют исходные данные, которые нежелательно изменять в ходе выполнения программы, то это параметры-значения. Они играют роль входных параметров – их значения передаются только в одном в направлении - из места вызова процедуры/ф-ции в процедуру/ ф-цию.	Если необходимо передать значения переменных в процедуру/функцию, а затем возвратит их изменившиеся значения в место вызова процедуры/функции – то это параметры-переменные. Они играют роль как входных так и выходных параметров.
Действуют только внутри процедуры/ф-ции и не могут передавать свои значения фактическим параметрам.	Могут получать значения фактических параметров, изменять их в процедуре/ф-ции и возвращать новые значения фактическим параметрам.
Описываются в заголовке:	
Имя_переменной: тип;	***Var*** *Имя_переменной: тип;*
Соответствующие им фактические параметры могут быть	
константами, переменными, выражениями	только переменными

Задача. Найти S=2!+3!+5!+8!

Решение. Для вычисления N! определим функцию fact.

```
PROGRAM Ex1;
    var S:Longint;
    Function fact(n:integer):longint;
            var P:longint; x:integer;
    Begin
            P:=1;
            for x:=1 to n do P:=P*x;
            fact:=P
    End;
BEGIN
    S:=fact(2)+fact(3)+fact(5)+fact(8);
    writeln('S=',S)
END.
```

Решение. Для вычисления N! определим процедуру fact.

```
PROGRAM Ex1;
    var F1,F2,F3,F4,S:Longint;
    Procedure  fact(n:integer; var f:longint);
            var x:integer;
    Begin  f:=1; for x:=1 to n do f:=f*x;
    End;
BEGIN
    fact(2,F1);  fact(3,F2);  fact(5,F3);  fact(8,F4);
    S:= F1+F2+F3+F4;
    writeln('S=',S)
END.
```

13. Массивы

Массив – это упорядоченная совокупность однотипных элементов, объединенных общим именем.

Одномерные массивы - последовательности

Например,
Массив A: 2, 5, 6, 10, 12.
Массив X: -1.2, 0, 1.2
Массив M: ‘a’,’b’,’c’,’d’,’e’,’f’.
Массив Slovo: ’Зима’, ’Весна’, ’Лето’, ’Осень’.

Любые данные, используемые в программе должны быть описаны в разделе описаний.

Описание одномерного массива:
1 способ: в разделе описания переменных.
Var *Имя_массива* **: array**[*Тип_индек*са] **of** *Тип_элементов_массива***;**
2 способ: через предварительное описание типа-массива.
Type *Имя_типа* **= array**[*Тип_индек*са] **of** *Тип_элементов_массива***;**
Var *Имя_массива* **:** *Имя_типа_массива***;**
Где,
Тип_индекса – любой порядковый тип (целый, символьный, логический, перечисляемый, тип-диапазон). Чаще всего используют тип-диапазон (1..10 или 0..5 и т.п.)

Например, опишем массивы A и M 1-м способом:.

```
Program Ex;
    Var A : array[1..5] of Integer;
        M : array[1..6] of Char;
```

а массивы X и Slovo – 2-м способом:

```
Program Ex;
    Type Massiv1 = array[1..3] of Real;
         Massiv2 = array[1..4] of String;
    Var X : Massiv1;
        Slovo: Massiv2;
```

Элементы массива идентифицируются его именем и индексом. И если в математике принято писать индекс внизу, то в Турбо-Паскале индекс заключают в квадратные скобки. В качестве индекса может быть переменная соответствующего типа или выражение.
Например, A[1], A[n], A[n+1].
Элементы массива используются как обычные переменные.
Например,

```
A[1]:=10;
K:=(A[1]+A[n])/2;
A[4]:= A[1]+A[2]+A[3];
```

Ввод и вывод элементов одномерного массива рассмотрим на примерах:
1 способ: ввод с клавиатуры
Массив A:

```
Program Ex;
    Var A : array[1..5] of Integer;
        i: integer;
Begin
    Writeln (‘Введите элементы массива:’);
    For i:=1 to 5 do
    Begin
      Writeln (‘A[’, i, ‘]=’);
      Readln (A[i]);
    End;
End.
```

Массив M:

```
Program Ex;
    Const n=6;
    Var M : array[1..n] of Char;
        i: integer;
Begin
    Writeln (‘Введите элементы массива:’);
    For i:=1 to n do
    Begin
      Writeln (‘M[’, i, ‘]=’);  Readln (M [i]);
    End;
End.
```

2 способ: в разделе описания констант

Массив Slovo:

```
Program Ex;
    Type Massiv = array[1..4] of String;
    Const Slovo: Massiv = (’Зима’, ’Весна’, ’Лето’, ’Осень’);
    Var  i: integer;
Begin
    Writeln (‘Вывод элементов массива на экран:’);
    For i:=1 to 4 do Write (M [i]);
End.
```

Двумерные массивы - матрицы

Например,

Массив К –матрица размера 3 на 4:

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

Описание двумерного массива:

1 способ: в разделе описания переменных.

Var *Имя_массива* **: array**[*Тип_индекса1, Тип_индекса2*] **of** *Тип_элементов_массива***;**

2 способ: через предварительное описание типа-массива.

Type *Имя_типа* **= array**[*Тип_индекса1, Тип_индекса2*] **of** *Тип_элементов_массива***;**

Var *Имя_массива* **:** *Имя_типа_массива***;**

Например, опишем массив К:

1 вариант:

```
Program Ex;
    Var K : array[1..3, 1..4] of Integer;
```

2 вариант:

```
Program Ex;
    Const n=3; m=4;
    Var K : array[1..n, 1..m] of Integer;
```

3 вариант:

```
Program Ex;
    Const n=3; m=4;
    Type Massiv =  array[1..n, 1..m] of Integer;
    Var K : Massiv;
```

Элементы массива идентифицируются его именем и индексами. Например, К[1,1], К[n, m], К[n-1, 3].

Ввод и вывод элементов двумерного массива рассмотрим на примерах:

1 способ: ввод с клавиатуры, вывод в виде таблицы:

```
Program Ex;
    Const n=3; m=4;
    Var K : array[1..n, 1..m] of Integer;
        i, j: integer;
Begin
    Writeln (‘Введите элементы массива:’);
    For i:=1 to n do
        For j:=1 to m do
        Begin
            Writeln (‘K[’, i, ‘,’  , j,  ’]=’);
            Readln (K[i,j]);
        End;

    Writeln (‘Вывод массива K:’);
    For i:=1 to n do
    Begin
        For j:=1 to m do Write (K[i, j]: 6);
        Writeln
    End;
End.
```

2 способ: в разделе констант, вывод в виде таблицы:

```
Program Ex;
    Const n=3; m=4;
          K : array[1..n, 1..m] of Integer= ((1,2,3,4),
                                            (5,6,7,8),
                                            (9,10,11,12));
    Var     i, j: integer;
Begin
    Writeln (‘Вывод массива K:’);
    For i:=1 to n do
    Begin
        For j:=1 to m do Write (K[i, j]: 6);
        Writeln
    End;
End.
```

14. Работа с величинами строкового типа в паскале

Тип ***string*** (строка) используется для обработки текста и похож на одномерный массив символов: ***array*** *[0..n]* ***of*** *char*. Строка трактуется как цепочка символов.

К любому символу в строке можно обратиться точно также как к элементу одномерного массива. Например, если S='Pascal', то S[4]='c'.

Пустая строка (нулевое значение строки) обозначается следующим образом: S:=''

Описать строки можно:

- ограничивая размер строки
  Например, строка *st* будет содержать не более 5 символов: **var** st: **string**[5];
- не задавая размер, в этом случае строка может содержать не более 255 символов.

Встроенные операции, функции для работы со строками

Название	Описание	Тип результата	Пример
операция сцепления (+)	соединяет значения строковых констант и переменных в одну	строковый	st:='м'+'и'; {Результат:st='ми'} st:=st+'р'; {Результат:st='мир'}
Length(*st*)	возвращает длину строки *st*	целый	A:=length ('слон'); Результат: A=4
Concat(*s1, s2,..., sn*)	сцепляет подряд строки-параметры *s1, s2, ..., sn*.	строковый	st:=concat ('пар', 'о', 'воз') Результат: st='паровоз'
Copy(*st, i, k*)	копирует из строки *st k* символов, начиная с *i.*	строковый	st:=copy ('паровоз', 5, 3) Результат: st='воз'
Pos(*sub, st*)	находит первое вхождение подстроки *sub* в строку *st* (результат равен номеру позиции, с которой она начинается или 0, если подстрока не найдена)	целый	A:=pos ('слон', 'заслон') Результат: A=3 B:=pos ('клон', 'клоун') Результат: B=0
Upcase(*ch*)	преобразует маленькую латинскую букву в большую	символь-ный	S:=upcase ('f') Результат: S='F'

Встроенные процедуры для работы со строками

Название	Описание	Пример
Delete(*st, i, k*)	удаляет из строки *st k* символов, начиная с *i*	st:= 'паровоз'; delete (st,1,4); Результат: st='воз'
Insert(*sub, st, i*)	вставляет подстроку *sub* в строку *st,* начиная с символа с номером *i*	st:= 'сон'; insert ('л', st, 2); Результат: st ='слон'
Str(*x, st*)	преобразует число x в переменную *st* (типа ***string***) Можно задавать формат преобразования – ***str (x:8:3, st)***, где *8* – длина строки *st*, а *3* – количество позиций в дробной части	str (pi:4:2, st); Результат: st='3.14'
val(*st, x, code*)	преобразует строку символов st в число. Параметр code=0, если преобразование возможно, или номеру позиции в строке st, где обнаружен ошибочный символ	st:='45.1'; val (st, x, code); Результат: x=45.1; code=0

15. Текстовые Файлы (очень краткая справка)

Описание файлового типа

Любой файл имеет три характерные особенности:

1. У него есть имя, что дает возможность программе одновременно работать с несколькими файлами;
2. Он содержит компоненты одного типа (любой тип Паскаля, кроме файлового);
3. Длина вновь создаваемого файла никак не оговаривается при его объявлении и ограничивается только емкостью устройств внешней памяти.

Описание переменных файлового типа:

Var *Имя_переменной*: file of *Тип_компонентов*;

В большинстве случае файлы состоят из текстовых строк или записей.
Описание переменных файлового типа, значения которого будет текст:

Var *Имя_переменной*: text;

Процедуры и функции работы с переменными файлового типа:

Каждому файлу ставится в соответствие файлового переменная определенного типа, поэтому перед началом работы с файлом необходимо установить данное соответствие:

Assign(F,Name);

где *F* – переменная файлового типа;
Name – название файла – переменная или константа строкового типа.
Например, файловой переменной F поставим в соответствие файл с полным именем ‘D:\text.txt’

Assign(F,‘D:\text.txt’);

При работе с файлом, прежде всего, необходимо его открыть, для этого используют одну из трех процедур:

Reset(F); открывает существующий файл для дополнения;
Rewrite(F); открывает новый файл, стирая предыдущий с таким же названием.
Append(F); открывает существующий файл для дополнения, указатель ставится на конец файла.

Операция закрытия файла является логическим окончанием работы с файлом:

Close(F);

Для обработки текстовых файлов используются процедуры **Read** и **Write**, **Readln** и **Writeln**, обеспечивающие соответственно чтение и запись одной строки и более в текстовый файл.

Задача. Найти сумму 10 вводимых с клавиатуры чисел. Числа и результат вычислений занести в текстовой файл Work.txt.

	Комментарий
Program Ex;	*Файловая переменная*
Var F:Text;	*Переменные для работы*
S,x,k:integer;	
Begin	*ставим в соответствие файлу переменную*
Assign(F, ‘C:\Work.txt’);	*открываем файл для записи*
Rewrite(F);	
S:=0;	
For K:=1 to 10 do	
Begin	*вводим число с клавиатуры*
Write(‘Введите число’); Readln(x);	*заносим значение переменной X в файл*
Writeln(F,x);	
S:=S+x;	
End;	*выводим ответ на экран*
Writeln(‘Ответ:’,S);	*записываем в файл слово “Ответ” и значение S*
Writeln(F, ‘Ответ:’,S)	*закрываем файл*
Close(F);	
End.	

16. Модуль CRT

Процедуры и функции модуля Crt управляют вводом с клавиатуры, выводом на экран дисплея, звуком. Чтобы воспользоваться командами модуля Crt сразу после заголовка программы необходимо вызвать модуль делается это указанием имени модуля после служебного слова USES:

```
program Primer;
  uses Crt;
```

Некоторые процедуры и функции модуля Crt:

Функция **ReadKey** принимает значение считываемого символа, при этом символ не выдается на экран.

Записывается: A:=readkey;

где переменная A символьного типа Char.

Данную функцию можно использовать для задержки результатов выполнения программы на экране до нажатия какой-нибудь клавиши.

Например,

```
....
writeln('x=',x);
readkey;
end.
```

KeyPressed анализирует нажатие клавиши на клавиатуре (принимает значение True, если клавиша нажата и False - в противном случае).

GotoXY располагает курсор в заданной позиции. Размеры экрана: 40x25 или 80x25

Записывается: gotoxy(A,B);

где A и B переменные целого типа Byte.

Например, ...

```
gotoxy(10,2); writeln('Привет');
```

ClrScr очищает экран и располагает курсор в верхнем левом углу.

Записывается: clrscr;

TextColor - установка цвета символов.

Записывается: textcolor(C);

где C - выражение типа Byte в интервале 0..15, соответствующее коду одного из цветов.

TextBackground - установка цвета фона.

Записывается: TextBackground (C);

где C - выражение типа Byte в интервале 0 .. 7.

Delay - создает задержку на заданное число милисекунд (0,001 сек.)

Записывается: Delay(MS);

где MS - выражение типа Word (Word - целый тип, диапазон значений: 0..65535)

Пример программы, в которой используются некоторые возможности модуля Crt.

```
program PRIMER;
  uses CRT;        {вызов модуля Crt}
  var A,B,C:real;
begin
  write('введите два числа');
  readln(A,B);
  C:= A + B;
  ClrScr;          {очистка экрана}
  writeln('Ответ C= ',C);
  ReadKey          {показывает результаты до тех пор, пока не будет нажата какая-нибудь клавиша}
end.
```

17. Модуль Graph

Константы, переменные, процедуры и функции графики становятся доступными только после подключения к программе модуля Graph.

Подключение производится сразу после заголовка программыпо команде: **uses Graph;**

Некоторые процедуры и функции модуля Graph

1. Управление экраном в графическом режиме

1.1. Установить графический режим работы:

InitGraph(dr, reg, p);

где dr - переменная типа Integer, определяющая тип графического драйвера;
reg - переменная типа Integer, определяющая тип графического адаптера;
p - переменная типа String, определяющая путь к файлу драйвера.

Например :

```
{ 1 вариант }
 Program Ex;
  uses Graph;
  var dr, reg : integer;
 Begin
  dr:=detect;
  {detect - машина сама
определит тип}
  InitGraph(dr, reg,'d:\tp\bgi');
  ...
```

```
{ 2 вариант }
 Program Ex;
  uses Graph;
  var dr, reg : integer;
 Begin
  dr:=VGA; reg:= VGAHi;
  InitGraph(dr,
reg,'d:\tp\bgi')
  ...
```

! Параметры dr и reg в процедуре InitGraph - параметры-переменные. Это означает, что на их месте не могут находиться константы, а только переменные.

1.2. Очистка графического экрана:

ClearDevice;

1.3. Возврат в текстовый режим:

CloseGraph ;

2. Координаты

Система координат: начало координат находится в верхнем левом углу экрана, ось абсцисс (Ox) направлена слева направо, ось ординат (Oy) - сверху вниз. Координаты - натуральные числа.

2.1. Получить максимальные (в данном режиме) значения координат X и Y соответственно:

GetMaxX; GetMaxY;

Например:

```
x := GetMaxX;  y := GetMaxY;  z:=x - y;
```

2.2. Получить текущие координаты относительно установленного окна :

GetX; GetY;

2.3. Установить прямоугольное окно на графическом экране:

SetViewPort(x1,y1,x2,y2, ot) ;

где x1, y1 - координаты левого верхнего угла;
x2, y2 - координаты правого нижнего угла
ot - выражение типа boolean, если true то элементы которые не входят в окно отсекаются.

2.4. Очистка графического окна:

ClearViewPort

3. Точки и линии

! Все линии вычерчиваются текущим цветом и стилем (см. 7.1, 4.1).

3.1. Поставить точку с координатами (x,y) цветом C:

PutPixel(x, y, C);

3.2. Вычертить отрезок прямой с началом в т. (x1,y1) и концом в т. (x2,y2):

Line(x1, y1, x2, y2);

3.3. Вычертить отрезок прямой от текущего положения курсора до положения, заданного смещениями его координат dx и dy, (т.е., если курсор находился в точке (a,b), то после выполнения этой процедуры эта точка соединиться с точкой (a + dx, b + dy):

LineRel (dx, dy);

4. Установка стиля линий

4.1. Установка вида,образца,толщины линии:

SetLineStyle(v,ex,t);

где V - вид линии, определяется константами:

SolidLn = 0; { сплошная линия }
DottedLn = 1; { точечная }
CenterLn = 2; { штрихпунктирная _ . _ . }
DashedLn = 3; { пунктирная _ _ _ _ }
UserBitLn= 4; { установленный пользователем }

Ex - данный параметр заполняется только для V=4, в противном случае берется равным нулю.

t - толщина линии - может принимать только одно из двух значений:

NormWidth = 1; {толщина в 1 пиксель }
ThickWidth = 3; {толщина в 3 пикселя }

Например:

SetLineStyle(0,0,3); {устанавливает сплошную линию толщиной в 3 пикселя.}

5. Многоугольники

5.1. Прямоугольник со сторонами параллельными сторонам экрана, у которого левый верхний угол находится в точке (x1,y1), а нижний правый в (x2,y2):

RecTangle(x1,y1,x2,y2);

5.2. Параллелепипед:

Bar3D(x1,y1,x2,y2,g,b);

где x1,y1,x2,y2 - координаты диагонали передней грани;
g - глубина в пикселях;
b - выражение типа Boolean ; если b = True, то верхняя грань изображается, если b=False, то - нет.

6. Дуги, окружности, эллипсы.

6.1. Окружность с центром в точке (x,y) радиуса R:

Circle(x, y, R) ;

6.2. Дуга окружности с центром в т. (x,y) от угла U1 до угла U2 радиуса R:

ARC (x, y, u1, u2, R);

где u1, u2 - начальный и конечный углы - задаются в градусах.
Если u1 = 0 и u2 = 359, то вычерчивается окружность.

6.3. Дуга эллипса с центром в точке (x,y) от угла u1 до угла u2 с горизонтальным радиусом Rx и вертикальным RY:

Ellipse (x, y, u1, u2, Rx, Ry);

Если u1 = 0 и u2 = 359, то эллипс вычерчивается целиком.

7. Цвет

7.1. Установить текущий цвет C (переменная или константа типа word) - для линий и букв:

SetColor(C);

7.2. Установить цвет фона CF (переменная или константа типа word):

SetBkColor(CF

7.3. Функция определения максимально доступного кода цвета:

GetMaxColor;

Например:

C:=GetMaxColor; SetColor(C);

7.4. Получить цвет точки с заданными координатами:

GetPixel(x, y)

8. Закраска, штриховка.

8.1. Установка стиля (st) и цвета (c) штриховки:

SetFillStyle (st, c);

где st - определяет тип штриховки:

0 - штриховка фоном; 1 - сплошная;

c - цвет штриховки.

8.2. Закраска произвольной замкнутой фигуры цветом, определенным в процедуре SetFillStyle:

FloodFill(x, y, cg);

где x,y - координаты точки внутри фигуры; cg - цвет границы фигуры.

8.3. Закраска прямоугольной области, задаваемой координатами диагонали AB A(x1,y1)B(x2,y2):

Bar(x1,y1,x2,y2);

8.4. Вычерчивание и закраска сектора окружности с центром в т. (x,y) от угла u1 до угла u2 радиуса r:

Pieslice(x,y,u1,u2,r);

9. Вывод текста.

9.1. Вывод текста, начиная с текущего положения курсора:

OutText(S);

где S - выражение типа String или Char.

9.2. Вывод текста, начиная с точки (x,y):

OutTextXY(x, y, S);

9.3. Установка вида шрифта, направления и размера:

SetTextStyle(h, n, r);

где h - номер шрифта - от 0 до 4;

n - направление вывода текста (0 - слева направо, 1 - снизу вверх),

r - размер букв от 1 до 10.