Операторы языка Паскаль. Простые и составные операторы.

Простые операторы.

1) Оператор присваивания.

Синтаксис:

<имя переменной>:=<выражение>

Выполнение: вычисляется значение выражения, стоящего в правой части оператора, и полученное значение присваивается переменной, имя которой указано в левой части оператора.

При вычислении значения выражения необходимо учитывать тип данных и операции, которые над ними выполняются. Если хотя бы один элемент данных, входящих в выражение, относится к вещественному типу или же в выражении встречается операция деления, то результат будет принадлежать вещественному типу данных.

2) Вызов процедуры.

Синтаксис:

<имя процедуры>[(<список значений параметров>)];

Выполнение: происходит переход к стандартной или определенной пользователем процедуре с неявным присваиванием фактических значений формальным параметрам процедуры.

Организовать ввод и вывод данных в языке Паскаль можно с помощью стандартных процедур read, readln, write, writeln.

Ввод/вывод данных всегда связан с обменом информацией между оперативной памятью и внешними носителями информации, в качестве которых могут выступать как файлы так и консольные устройства ввода/вывода (клавиатура, дисплей, принтер и т.д.).

Синтаксис:

read([<имя устройства ввода>,]<список имен переменных>);

readln([<имя устройства ввода>,]<список имен переменных>);

write([<имя устройства вывода>,]<список значений>);

writeln([<имя устройства вывода>,]<список значений>);

Выполнение:

ввод данных: с устройства ввода последовательно считываются значения и присваиваются переменным, имена которых указаны в списке. При этом, необходимо следить за тем, чтобы совпадали типы у переменных и присваиваемых им значений (нельзя присвоить символьное значение вещественной переменной и т.п.);

вывод данных: значения, указанные в списке, последовательно выводятся на устройство вывода данных.

Отличие процедур read и readln, write и writeln состоит в том, что при выполнении процедур writeln и readln к последнему значению автоматически дописывается управляющий символ конца строки, что означает переход на новую строку экрана или файла.

Составные операторы.

Составной оператор представляет собой последовательность операторов, заключенных в операторные скобки begin и end.

Выполнение составного оператора заключается в том, что один за другим выполняются операторы, указанные внутри операторных скобок, в той последовательности, как они записаны, до тех пор, пока не будет полностью исчерпана вся последовательность.

1. ***Структурные операторы.***

1) Условный оператор

Синтаксис:

if <условие>
then <оператор>
[else <оператор>];

где <условие> - выражение булевского типа;
<оператор> - любой оператор языка Паскаль (в том числе и составной).

Выполнение: вычисляется значение выражения, стоящего в условии. Если это значение истинно, то выполняется оператор, следующий за служебным словом then. Если условие ложно, то выполняется оператор, стоящий за служебным словом else.

Условный оператор может использоваться и в сокращенной форме, когда отсутствует часть оператора, начиная со служебного слова else. В этом случае оператор выполняется при истинном условии, а если условие ложно, то происходит переход к оператору, следующему за условным. Например:

```
if x>y                  |   if x>y
then                    |   then
    begin               |   x:=0;
        x:=0;           |   y:=0;
        y:=0;
    end;
```

В условном операторе за служебными словами then и else могут следовать любые операторы языка Паскаль, в том числе и условные. Поэтому возможны «вложенные» условные операторы. Например,

```
if x>y
then x:=x-y
else if x=y
        then x:=0
        else y:=y-x;
```

Условие может быть составным. В этом случае каждая его часть заключается в скобки. Например:

```
if (x>5) and (x<15)
if (a=b) or (not(t))
```

2) Оператор варианта

С помощью условного оператора осуществляется выбор одного из двух возможных действий в зависимости от значения булевского выражения. Однако, часто бывает необходимо выбрать одно из нескольких (больше двух) действий. Для реализации такого рода ситуаций в языке Паскаль предусмотрен специальный оператор варианта.

Синтаксис:

```
case <выражение> of
        <значение 1>: <оператор>;
        …
        <значение n>: <оператор>;
        [else <оператор>]
end;
```

где <выражение> - выражение любого скалярного типа (кроме real);
<оператор> - любой оператор языка Паскаль.

Выполнение: вычисляется значение выражения, стоящего после case. Это значение используется для выбора одного из возможных действий. Если полученное значение выражения совпадает с одним из перечисленных после of, то выполняется соответствующий оператор. Если один и тот же оператор должен выполняться при различных значениях выражения, то эти значения могут быть указаны через запятые или в виде интервала. Если в операторе присутствует часть else, то оператор, указанный за данным служебным словом, выполняется тогда, когда значение выражения не совпадает ни с одним из значений, перечисленных после of. Например:

Пусть m – номер месяца, y – порядковый номер года. Определить d – количество дней в соответствующем месяце.

```
case m of
        1,3,5,7,8,10,12: d:=31;
        4,6,9,11: d:=30;
        2: if (y mod 4=0) and (y mod 100 <>0)
              then d:=29
              else d:=28
end;
```

<u>Структурные операторы.</u>

3) Оператор цикла

Оператор цикла используется для организации многократного повторения выполнения одних и тех же операторов. В языке Паскаль существует три типа оператора цикла.

а) оператор цикла с предусловием

Синтаксис:

```
while <условие> do
        <оператор>;
```

где <условие> - булевское выражение;
<оператор> - любой оператор языка Паскаль, называемый «телом цикла».

Выполнение: вычисляется значение выражения, стоящего в условии. Если это значение истинно, то выполняется оператор и снова происходит возврат к вычислению значения булевского выражения. Как только условие станет ложным, выполнение цикла завершается и выполняется оператор, следующий за телом цикла.

Оператор цикла с предусловием может быть не выполнен ни разу в том случае, если при первом же вычислении значения условия оно будет ложным.

Замечание.

Оператор, стоящий в теле цикла, обязательно должен изменять значение условия, иначе цикл будет выполняться бесконечное число раз.

б) оператор цикла с постусловием

Синтаксис:

repeat

<оператор>

until <условие>;

где <условие> - булевское выражение;

<оператор> - любой оператор языка Паскаль.

Выполнение: выполняется оператор, стоящий в теле цикла, затем вычисляется значение выражения, стоящего в условии. Если это значение ложно, то происходит переход к выполнению оператора, стоящего в теле цикла. Как только условие станет истинным, выполнение цикла завершается.

Для оператора цикла с постусловием справедливо замечание, сделанное выше.

Отличие циклов с предусловием и с постусловием:

- ✓ при выполнении цикла с предусловием тело цикла выполняется, если условие истинно, а в цикле с постусловием повторение осуществляется, если значение условия ложно;
- ✓ тело цикла с постусловием обязательно будет выполнено хотя бы один раз независимо от значения условия, тогда как тело цикла с предусловием может быть не выполнено ни разу;
- ✓ если в теле цикла с предусловием используется составной оператор, то он заключается в операторные скобки. При использовании составного оператора в теле цикла с постусловием роль операторных скобок выполняют служебные слова repeat и until.

в) оператор цикла с параметром

Оператор цикла с параметром целесообразно применять тогда, когда заранее известно количество повторений цикла. Оператор цикла с параметром используется в двух видах:

1. *Синтаксис*:

for <имя перем.>:= <выражение 1> to <выражение 2> do

<оператор>;

где <имя переменной> - идентификатор переменной целого типа, называемой параметром цикла;

<выражение 1> и <выражение 2> - выражения, задающие начальное и конечное значения параметра цикла;

<оператор> - любой оператор языка Паскаль.

Для объяснения выполнения обозначим:

i – имя переменной, z1 и z2 – выражения 1 и 2 соответственно, s – оператор, образующий тело цикла.

Тогда цикл имеет вид:

for i:= z1 to z2 do s;

Выполнение: вычисляются значения выражений z1 и z2. Если z1>z2, то выполнение оператора цикла завершается. Если z1<=z2, то переменной i присваивается значение z1 и выполняется оператор s. Затем значение

переменной I увеличивается на 1 и вычисляется значение булевского выражения i<=z2. Если это значение истинно, то вновь выполняется оператор s, берется следующее значение параметра цикла и т.д. Выполнение оператора цикла с параметром завершается тогда, когда станет ложным значение булевского выражения i<=z2.

2. *Синтаксис*:

for <имя перем.>:= <выражение 1> downto <выражение 2> do
<оператор>;

В обозначениях из предыдущего раздела оператор цикла принимает следующий вид:

for i:= z1 downto z2 do s;

В этом случае z1>z2 и значение параметра цикла при каждом повторении цикла не увеличивается, а уменьшается на 1.

Выполнение: вычисляются значения выражений z1 и z2. Если z1<z2, то выполнение оператора цикла завершается. Если z1>=z2, то переменной i присваивается значение z1 и выполняется оператор s. Затем значение переменной i уменьшается на 1 и вычисляется значение булевского выражения i>=z2. Если это значение истинно, то вновь выполняется оператор s, берется следующее значение параметра цикла и т.д. Выполнение оператора цикла с параметром завершается тогда, когда станет ложным значение булевского выражения i>=z2.

Замечание.

После завершения выполнения оператора цикла значение параметра цикла не определено.

4) Примеры использования структурных операторов.

1. Вычислить значение факториала числа.

```
Program factorial;
   uses crt;
   var n, i:integer; p:longint;
Begin
   clrscr;
   readln(n);
   p:=1;
   for i:=1 to n do
      p:=p*i;
   write(n,'! = ', p);
End.
```

2. Вычислить наибольший общий делитель двух целых чисел a и b.

```
Program NOD;
   uses crt;
   var a,b: integer;
Begin
   clrscr;
   readln(a,b);
   a1:=a;
   b1:=b;
   while a<>b do
```

```
        if a>b then a:=a-b
        else b:=b-a;
    writeln('НОД(',a1,',',b1,') = ',a);
    readln;
End.
```

3. Вычислить наименьшее из 20 целых чисел.

```
Program min_of_20;
    uses crt;
    var n, min, i: integer;
Begin
    clrscr;
    writeln('Введите первое число');
    readln(n);
    min:=n;
    for i:=2 to 20 do
        begin
            writeln('Введите ',i , '-е число');
            readln(n);
            if n<min then min:=n
        end;
    writeln('min = ', min);
    readln;
End.
```