Структурные (сложные) операторы

Операторы IF и CASE

Оператор IF

Оператор if позволяет выбрать один из двух возможных вариантов развития программы. Выбор осуществляется в зависимости от *выполнения* условия.

В общем виде инструкция if записывается так:

if условие **then begin**
// здесь *инструкции, которые надо выполнить,*
// если условие истинно.
end
else begin

// здесь *инструкции, которые надо выполнить,*
// если условие ложно.
end;

Обратите внимание, что перед else (после end) точка с запятой не ставится.

```
program Project1;
//вводятся 2 числа, вывести наибольшее
 {$APPTYPE CONSOLE}
 uses  SysUtils;
var a,b:real;
begin
write('a='); readln(a);  //ввод числа a
write('b='); readln(b);   //ввод числа b
if a>b then writeln('max a=',a:6:2) // вывод a
    else writeln('max b=',b:6:2); // вывод b
readln;
end.
```

```
program Project2;
//вводятся 2 числа, вывести наибольшее
 {$APPTYPE CONSOLE}
 uses  SysUtils;
var a,b:real;
begin
write('a='); readln(a);  //ввод числа a
write('b='); readln(b);   //ввод числа b
if a>b then writeln('max a=',a:6:2); //вывод числа a

If b>a then writeln('max b=',b:6:2); //вывод числа b

readln;
end.
```

```
program Project3;
//вводятся 2 числа, вывести наибольшее, если
//числа равны вывести сообщение
 {$APPTYPE CONSOLE}
 uses  SysUtils;
var a,b:real;
begin
write('a='); readln(a);  //ввод числа a
write('b='); readln(b);   //ввод числа b
if a>b then writeln('max a=',a:6:2); //вывод числа a
If b>a then writeln('max b=',b:6:2); //вывод числа b
If b=a then writeln('a=b  ‘,a:6:2); //вывод сообщения a=b
readln;
end.
```

```
program Project4;
//вводятся 2 числа, вывести наибольшее,
// если числа равны вывести сообщение
 {$APPTYPE CONSOLE}
 uses  SysUtils;
var a,b:real;
begin
write('a='); readln(a);  //ввод числа a
write('b='); readln(b);   //ввод числа b
if a>b then writeln('max a=',a:6:2)  //вывод числа a
          //вывод числа b
     else   If b>a then writeln('max b=',b:6:2)
//вывод сообщения a=b
                        else writeln('a=b  ‘,a:6:2);
readln;
end.
```

Логические операции

Над логическими аргументами определены следующие операции:
NOT - логическое отрицание ("НЕ")
AND - логическое умножение ("И")
OR - логическое сложение ("ИЛИ")
XOR - логическое "Исключающее ИЛИ"

Результаты выполнения этих операций над переменными A и B логического типа приведены в таблице истинности.

A	B	not A	A and B	A or B	A xor B
true	true	false	true	true	false
true	false		false	true	true
false	true	true	false	true	true
false	false		false	false	false

```
program Project5;
//вводятся 2 числа, вывести наибольшее,
// если числа равны вывести сообщение
 {$APPTYPE CONSOLE}
 uses  SysUtils;
var a,b:real;
begin
write('a='); readln(a);  //ввод числа a
write('b='); readln(b);   //ввод числа b
//вывод числа a
if (a>b) and (a<>b) then writeln('max a=',a:6:2);
//вывод числа b
If (b>a) and (b<>a) then writeln('max b=',b:6:2);
//вывод сообщения a=b
 If a=b writeln('a=b  ‘,a:6:2);
readln;
end.
```

```
program Project6;
//вводятся 2 числа, вывести наибольшее,
// если числа равны вывести сообщение
 {$APPTYPE CONSOLE}
 uses  SysUtils;
var a,b:real;
begin
write('a='); readln(a);  //ввод числа a
write('b='); readln(b);   //ввод числа b
if  (a>b) and (a<>b)
     then writeln('max a=',a:6:2)
     else    If (b>a) and (b<>a)
                then   writeln('max b=',b:6:2)
                else    writeln('a=b  ‘,a:6:2);
readln;
end.
```

Оператор case

Оператор case позволяет эффективно реализовать множественный выбор. В общем виде она записывается следующим образом:

case *Селектор of*
список1: **begin** *{ инструкции 1 }* end;
список2: **begin** *{ инструкции 2 }* end;
списокM: **begin** *{ инструкции N }* **end;**
else
Begin *{ инструкции }* **end;**
end;

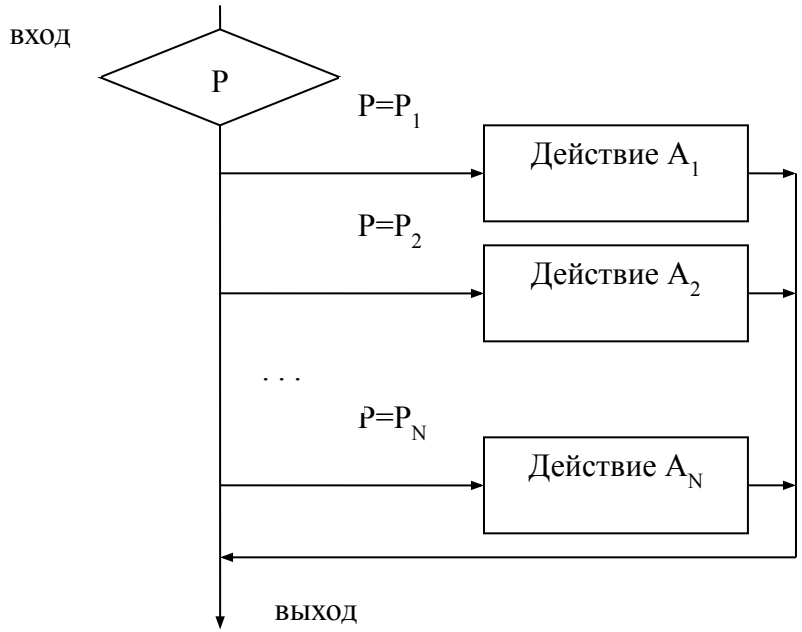
Оператор case

Селектор — выражение, значение которого определяет дальнейший ход выполнения программы (т. е. последовательность инструкций, которая будет выполнена);

Список N — список констант. Если константы представляют собой диапазон чисел, то вместо списка можно указать первую и последнюю константу диапазона, разделив их двумя точками. Например, список 1, 2, 3, 4, 5, 6 может быть заменен диапазоном 1..6.

- Селектор может быть любого стандартного скалярного типа, кроме вещественного.
- Селектор может относиться к пользовательским типам: перечисляемому или интервальному.
- Список констант выбора состоит из произвольного количества значений или диапазонов, отделенных друг от друга запятыми.
- Границы диапазона записываются двумя константами через разграничитель «..». Тип констант должен совпадать с типом селектора.

Блок- схема использования CASE



Использование Case: в программу вводится номер дня и программа выводит соответствующее название дня:

«Рабочий день», «Суббота!», «Воскресенье!».

Пример 1

```
...
   case n of
   1,2,3,4,5: Writeln('Рабочий день‘) ;
   6: Writeln('Суббота!‘);
   7: Writeln ('Воскресенье!‘);
end;
...
```

Использование Case: в программу вводится номер дня и программа выводит соответствующее название дня:

«Рабочий день», «Суббота!», «Воскресенье!».

Пример 2

```
…
case n of
   1..5: Writeln('Рабочий день‘) ;
   6: Writeln('Суббота!‘);
   7: Writeln ('Воскресенье!‘);
end;

…
```

Использование Case: в программу вводится номер дня и программа выводит соответствующее название дня:

«Рабочий день», «Суббота!», «Воскресенье!».

Пример 3

```
…
case n of
   6: Writeln('Суббота!‘);
   7: Writeln ('Воскресенье!‘);
else Writeln('Рабочий день‘) ;
   end;
…
```

Использование Case: в программу вводится номер дня и программа выводит соответствующее название дня: «Рабочий день», «Суббота!», «Воскресенье!», если введено недопустимое значение, то программа выводит соответствующее сообщение .

Пример 4

```
...
case n of
   1..5: Writeln('Рабочий день‘) ;
   6: Writeln('Суббота!‘);
   7: Writeln ('Воскресенье!‘);
else Writeln(‘Ошибка ввода‘) ;
   end;

...
```

Циклический вычислительный процесс (ЦВП).

Операторы циклов (повторов)

Циклические вычисления реализуются с помощью 3-х операторов циклов

FOR ... TO(DOWNTO) ... DO

(оператор цикла со счетчиком);

WHILE ... DO

(оператор цикла с предусловием);

REPEAT ... UNTIL

(оператор цикла с постусловием

Оператор **FOR.**
Форматы записи оператора.

FOR *<счетчик цикла>* **:=** *<нач. значение>* **TO** *<конечное зн.>* **DO**
begin
{операторы тела цикла};
end;

или

FOR *<счетчик цикла>* **:=** *<нач. зн.>* **DOWNTO** *<кон. зн.>* **DO**
begin
{операторы тела цикла};
end;

Использование оператора For

```
…
FOR n := 1 TO 5 DO
WRITE(n,'    ');
…
```

Результат работы фрагмента программы

1 2 3 4 5

_

Использование оператора For …

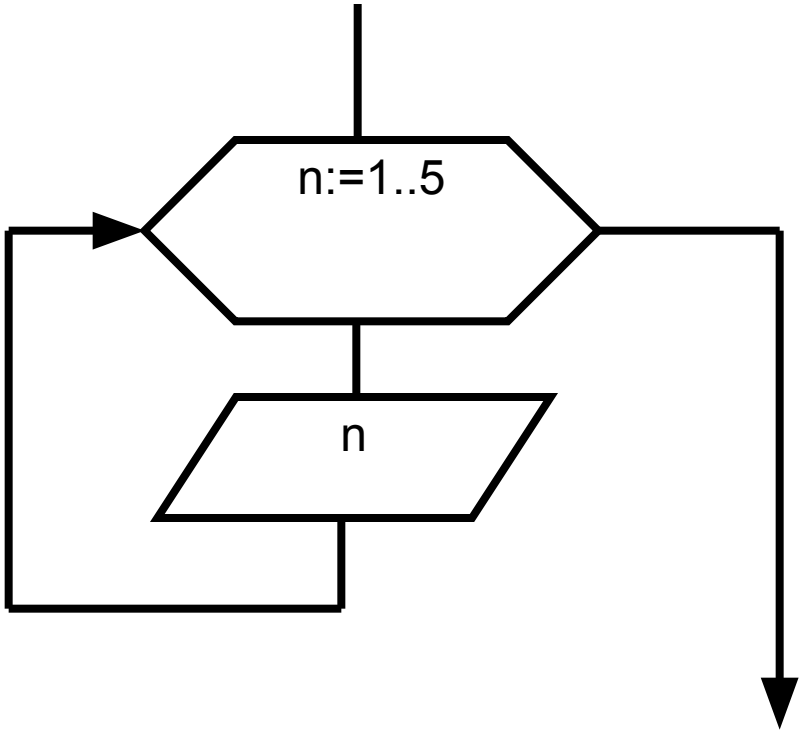
```
…
FOR n := 5 DOWNTO 1 DO
WRITE(n,'    ');
…
```

Результат работы фрагмента программы

5 4 3 2 1

_

Блок-схема



Найти сумму чисел S=1+2+3+…+n для заданного n.

```
…
write(‘n=‘); readln(n);
s:=0;
for i:=1 to n do
    s:=s+i;
writeln(‘s=‘,s); //вывод результата
…
```

Результат работы фрагмента программы при n=5

n=5

S=15

Найти сумму чисел S=1+2+3+…+n для заданного n.

```
…
write(‘n=‘); readln(n);
s:=0;
for i:=1 to n do
    begin
    s:=s+i;
    write(‘ s=‘,s); //вывод результата после каждого цикла
    end;
…
```

Результат работы фрагмента программы при n=5

n=5

S=1 S=3 S=6 S=10 S=15

Найти произведение чисел F=1*2*3*…*n для заданного n, факториал числа n (n!).

```
…
write(‘n=‘); readln(n);
F:=1;
for i:=1 to n do
   begin
   F:=F*i;
   write(‘ F=‘,F);      //вывод результата
   end;
…
```

Результат работы фрагмента программы при n=5

```
n=5
F=1 F=2 F=6 F=24 F=120
```

Рассчитать факториал числа (n+1)! для заданного n: F=1*2*3*4*…*n*(n+1).

…

```
write(‘n=‘); readln(n);
F:=1;
for i:=1 to n do
   F:=F*(i+1);
writeln(‘F=‘,F);
```

…

Результат работы фрагмента программы при n=5

```
n=5
F= 720
```

Рассчитать x^n для заданных x и n.

```
...
write('x='); readln(x);
write('n='); readln(n);
A:=1;
for i:=1 to n do
    A:=A*x;
writeln('A=',A);
...
```

Результат работы фрагмента программы при x=2 и n=5

x=2
n=5
A=32

Рассчитать x^n для заданных x и n.

```
…
write(‘x=‘); readln(x);
write(‘n=‘); readln(n);
A:=1;
for i:=1 to n do
    begin
    A:=A*x;
    write(‘A=‘,A);
    end;
…
```

Результат работы фрагмента программы при x=2 и n=5

x=2

n=5

A=2 A=4 A=8 A=16 A=32

Оператор **WHILE...DO**

Формат оператора

WHILE <*логическое условие*> DO
begin
{операторы тела цикла};
end;

логическое условие - оператор отношения (логическое выражение), определяющий условие завершения цикла;
операторы тела цикла — любой исполнимый оператор или блок операторов, заключенных в операторные скобки.

Операторы, входящие в тело цикла, выполняются до тех пор, пока при проверке логического условия получаем результат **TRUE** («истина»).

Оператор цикла While … do …

При использовании оператора **WHILE:**

- может не выполниться ни одного цикла, если логическое условие ложно;
- произойдет зацикливание программы, если логическое условие будет всегда истинно.

Пример использования оператора While … do …

```
…
i:=4;
While i<=7 do
    Begin
      Write(i:2);
      i:=i+1
    End;
…
```

Результат работы программы

4 5 6 7

Оператор цикла REPEAT...UNTIL

Формат оператора

REPEAT

<1 *оператор тела цикла;*>

<2 *оператор тела цикла*;>

...

<N *оператор тела цикла* >

UNTIL <*логическое условие***>*;**

Логическое условие — логическое выражение, определяющее условие завершения цикла;

операторы тела цикла – любой оператор или группа операторов.

При использовании оператора REPEAT… UNTIL … :

1. Операторы тела цикла выполнятся обязательно один раз при любом логическом условии.
2. Если логическое условие будет постоянно ложно, то программа зациклится.

Пример использования оператора REPEAT … UNTIL

```
….
i:=4;
Repeat
Write(i:2);
    i:=i+1
Until i>7;
…
```

Результат: 4 5 6 7

Пример работы операторов циклов

Пример. Найти сумму конечного ряда. Значение x вводится с клавиатуры.

$$S = \sum_{i=1}^{10} x * i$$

```
program Project61;
var  x,s:real;
i:integer;
begin
Write('x='); Readln(x);
s:=0;
for i:=1 to 10 do
     s:=s+x*i;
Writeln('s=',s:3:5);
Readln;
end.
```

```
program Project62;
var  x,s:real;
i:integer;
begin
Write('x='); Readln(x);
s:=0;
i:=1;
repeat
   s:=s+x*i;
    i:=i+1;
until i>10;
Writeln('s=',s:3:5);
Readln;
end.
```

```
program Project63;
var  x,s:real;
i:integer;
begin
Write('x='); Readln(x);
s:=0;
i:=1;
while i<=10 do
begin
  s:=s+x*i;
   i:=i+1;
end;
Writeln('s=',s:3:5);
Readln;
end.
```

Отличия и особенности работы с циклическими операторами

Цикл с предусловием While
(пока условие истинно)

1. До начала цикла должны быть сделаны начальные установки переменных, управляющих условием цикла, для корректного входа в цикл

2. В теле цикла должны присутствовать операторы, изменяющие переменные условия так, чтобы цикл через некоторое число итераций (повторов) завершился

3. Цикл работает пока условие истинно (пока True)

4. Цикл завершается, когда условие становится ложным (до False)

5. Цикл может не выполниться ни разу, если исходное значение условия при входе в цикл False

6. Если в теле цикла требуется выполнить более одного оператора, то необходимо использовать составной оператор (begin … end)

Цикл с постусловием Repeat
(до истинности условия)

1. До начала цикла должны быть сделаны начальные установки переменных, управляющих условием цикла, для корректного входа в цикл

2. В теле цикла должны присутствовать операторы, изменяющие переменные условия так, чтобы цикл через некоторое число итераций (повторов) завершился

3. Цикл работает пока условие ложно (пока False)

4. Цикл завершается, когда условие становится истинным (до True)

5. Цикл обязательно выполнится, как минимум, один раз

6. Независимо от количества операторов в теле цикла, использование составного оператора не требуется

Цикл со счетчиком (с параметром)
For … to (downto) … do

1. Начальная установка переменной счетчика цикла до заголовка не требуется
2. Изменение в теле цикла значений переменных, стоящих в заголовке не допускается
3. Количество итераций цикла неизменно и точно определяется значениями нижней и верхней границ и шага приращения
4. Цикл может не выполниться ни разу, если шаг цикла будет изменять значение счетчика от нижней границы в направлении, противоположном верхней границе