Информационные технологии

Лекция №6

Управляющие структуры в Pascal

Условный оператор

```
if <условие> then <оператор 1> [else <оператор 2>]
```

Условие – значение типа boolean или логическая операция.
Если условие верно, выполняется оператор, или блок операторов, следующий за then, в противном случае выполняется блок операторов после else, если он есть.
Условия могут быть вложенными и в таком случае, любая встретившаяся часть else соответствует ближайшей к ней "сверху" части then.

```
Var
  a : integer;
Begin
  readln (a);
  if a > 10 then
    writeln('a more than 10')
  else
    writeln('a less than 10');
end.
```

Условный оператор

```
if <условие> then <оператор>;
```

```
var
  i: integer;
Begin
  write('Введите Ваш возраст: ');
  readln(i)
  If i > 19 then
    writeln('Вы уже не можете выступать в соревнованиях RoboCupJunior');
End.
```

Возраст участников соревнований по робототехнике RoboCupJunior ограничен 19 годами

Оператор выбора одного из вариантов

```
case Выражение of
  Вариант1: Оператор1;
  Вариант2: Оператор2;
  ВариантN: ОператорN;
  [else ОператорN1;]
end;
```

Выражение в простейших случаях может быть целочисленным или символьным. В качестве вариантов можно применять:

1. Константное выражение такого же типа, как и выражение после case. Константное выражение отличается от обычного тем, что не содержит переменных и вызовов функций, тем самым оно может быть вычислено на этапе компиляции программы, а не во время выполнения.
2. Интервал, например: 1..5, 'a'..'z'.
3. Список значений или интервалов, например: 1,3,5..8,10,12.

Выполняется оператор case следующим образом: вычисляется выражение после слова case и по порядку проверяется, подходит полученное значение под какой-либо вариант, или нет. Если подходит, то выполняется соответствующий этому варианту оператор, иначе - есть два варианта. Если в операторе case записана часть else, то выполняется оператор после else, если же этой части нет, то не происходит вообще ничего.

Рассмотрим пример. Пусть пользователь вводит целое число от 1 до 10, программа должна приписать к нему слово "ученик" с необходимым окончанием (нулевое, "а" или "ов").

```
program SchoolChildren;
var
  n: integer;
begin
  write('Число учеников --> ');
  readln(n);
  write(n,' ученик');
  case n of
    2..4: write('а');
    5..10: write('ов');
  end;
  readln;
end.
```

```
program Example_CASE_1;
Var
  A : integer;
begin
    Write('Введите оценку: ');
    Readln(A);
    case A of
        2 : Writeln('неудовлетворительно');
        3 : Writeln('удовлетворительно');
        4 : Writeln('хорошо');
        5 : Writeln('отлично')
        else Writeln('Ошибка!')
    end;
end.
```

```
PROGRAM Example_CASE_2;
VAR
  Hour : integer;
BEGIN
    Read( Hour );
    CASE Hour OF
        0, 24 : Write('Полночь');
        1..4 : Write('Ночь');
        5..7 : Write('Раннее утро');
        8..11 : Write('Утро');
        12 : Write('Полдень');
        13..17 : Write('День');
        18..23 : Write('Вечер')
        ELSE Write('Введено число вне диапазона 0..24!')
    END;
END.
```

Операторы циклов

Цикл с параметром (со счетчиком)

```
for <переменная>:=<нач_значение> to <кон_значение> do <оператор>.
```

Вместо to возможно слово downto. Рассмотрим такой пример: требуется вывести на экран таблицу квадратов натуральных чисел от 2 до 20.

```
var
  i: integer;
begin
  for i:=2 to 20 do
    writeln(i,' ',sqr(i));
end.
```

```
VAR i : integer;
BEGIN
  FOR i := 1 TO 10 DO Write(i:5);
  FOR i := 5 DOWNTO 0 DO Write(i:5)
END.
```

```
CONST n = 10;
      m = 3;
VAR i,
    k : integer;
BEGIN
  k := 1;
  FOR i := k TO n DO Write(i:5);
  FOR i := (m+2) DOWNTO 0 DO Write(i:5)
END.
```

Операторы FOR могут быть вложенными друг в друга, например

```
VAR
  i,j : integer;
BEGIN
  WriteLn('Таблица умножения');
  FOR i := 1 TO 10 DO begin
     FOR j := 1 TO 10 DO Write((i*j):5);
    WriteLn;
  end;
END.
```

Цикл с предусловием

```
while <условие> do <оператор>.
```

пока условие истинно, выполняется оператор (в этом случае оператор может не выполниться ни разу, т.к. условие проверяется до выполнения). Под оператором здесь понимается либо простой, либо составной оператор (т.е. несколько операторов, заключённых в begin ... end).

```
var
  i: integer;
begin
  i := 1;
  while i < 10 do
    begin
      writeln('текущее значение i: ', i);
      i := i + 1;
    end;
end.
```

```
VAR
  Factorial, N : Integer;
BEGIN
  Factorial := 1; {стартовое значение факториала =0! }
  N := 1;         {стартовое значение для условия цикла }
  WHILE N<=10 DO                  {заголовок цикла, условие }
   begin                          {начало тела цикла }
     Factorial := Factorial*N; {вычисление факториала N! }
     N := N + 1                   {N должно меняться в цикле}
   end;                           {конец тела цикла }
  WriteLn('10!= ',Factorial);  {вывод результата расчета }
END.
```

Цикл с постусловием

```
repeat <оператор> until <условие>
```

Цикл работает следующим образом: выполняется оператор, затем проверяется условие, если оно пока еще не выполнилось, то оператор выполняется вновь, затем проверяется условие, и т. д. Когда условие, наконец, станет истинным выполнение оператора, расположенного внутри цикла, прекратится, и далее будет выполняться следующий за циклом оператор.

```
var
 i : integer;
begin
  I := 1;
  repeat
    writeln('текущее значение i: ', i);
    i := i + 1;
  until i = 10;
end.
```

```
VAR
  N : Integer;
BEGIN
  . . .
  REPEAT
    Write('Введите целое число от 0 до 10: ');
    ReadLn(N);
  UNTIL (N>=0) and (N<=10);
  . . .
END.
```

“Вечный цикл”

```
Program super;
begin
  repeat until False;
end.
```

Процедуры BREAK, CONTINUE, EXIT и HALT

Процедура BREAK применяется для досрочного прекращения циклов WHILE, REPEAT…UNTIL и FOR.

Пример: Определение номера первого нулевого элемента в массиве A

```
VAR
  i, num : Integer;
  A : array[1..10] of Integer;
BEGIN
  . . . { Ввод элементов массива A }
  num := 0;
  FOR i:=1 TO 10 DO { Цикл для i от 1 до 10 }
   IF A[i]=0 THEN begin { если найден нулевой элемент }
                    num := i; { запоминаем его номер }
                    BREAK      { прекращаем цикл }
                  end;
  IF num <> 0 THEN WriteLn(num)
              ELSE WriteLn('Нулевых элементов нет');
END.
```

Процедура CONTINUE вызывает переход к следующей циклической итерации игнорируя расположенные ниже операторы, составляющие тело цикла.

Определение наибольшего общего делителя (НОД) двух чисел

```
VAR
  CommDiv,         { кандидат в НОД }
  num1,            { первое число }
  num2 : Integer; { второе число }
BEGIN
  Write('Введите первое число: '); ReadLn(num1);
  Write('Введите второе число: '); ReadLn(num2);
  FOR CommDiv := num1 DOWNTO 1 DO begin
    IF (num2 mod CommDiv)<> 0 THEN CONTINUE;
    IF (num1 mod CommDiv) = 0 THEN begin
       WriteLn('Наибольший общий делитель: ', CommDiv);
       BREAK
    end {if}
end {for}
END.
```

В примере применен цикл FOR, в котором по убывающий перебираются возможные кандидаты в наибольший общий делитель. Критерий делимости – равенство нулю остатка от деления (операция mod). Если кандидат (значение переменной CoomDiv) не является делителем числа num2, по команде CONTINUE начинается следующий проход цикла, а операторы, оставшиеся в теле цикла, не выполняются. Если число (CommDiv) оказывается делителем числа num2 и num1, что проверяется вторым оператором IF тела цикла, значит, оно – общий делитель. А поскольку перебор идет от больших к меньшим значениям, первый найденный общий делитель – наибольший. Выводим его на печать и прерываем цикл командой BREAK.

EXIT

Процедура EXIT завершает работу своего программного блока. Если EXIT вызывается внутри процедуры или функции, то их работа завершается. Если EXIT вызывается в основном блоке программы, то это приводит к ее завершению.

HALT

Процедура HALT, или более полно HALT(n), завершает работу программы с кодом завершения n. Этот код впоследствии может быть проанализирован, в частности командой IF ERRORLEVEL в среде MS-DOS. Значение ERRORLEVEL после остановки программы будет равно значению n. Вызов процедуры HALT без параметра эквивалентен вызову HALT(0).