Тема 5

ВЕТВЛЕНИЯ И ЦИКЛЫ

Условный оператор

Решим следующую задачу. Даны два целых числа a и b. Написать программу для нахождения их частного.

```
var
 a, b: integer;
 c: real;
begin
 read(a, b);
 c := a/b;
 write(c);
end.
```

Приведенная программа работает для всевозможных a и b, кроме тех случаев, когда b равно нулю. При b равном нулю возникает ошибка. На нуль делить нельзя, но хотелось бы, чтобы программа сама обработала подобный случай и попросила пользователя ввести другие данные. Подобное решение может выглядеть следующим образом:

```
var
 a, b: integer;
begin
 read(a, b);
 if b = 0 then          // проверяем b на равенство нулю
  write('Введите не 0.')// выводим сообщение, если b
                        // равно 0
 else
  write(a/b);           // делим, если b не равно 0
end.
```

В приведенной программе условный оператор `if` позволил предусмотреть два варианта работы: при $b = 0$ и $b \neq 0$, т. е. позволил организовать ***ветвление***.

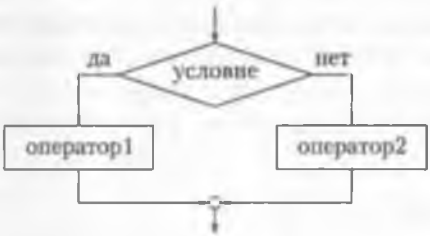
Ветвление — это такая форма организации действий, при которой в зависимости от выполнения (истинности) или невыполнения (ложности) условия выполняется одна либо другая последовательность действий.

Схема условного оператора такова:

```
if условие then
 оператор1
else
 оператор2;
```

Если условие истинно, то выполняется оператор1, иначе — оператор2. То есть из двух команд выполнится только одна в зависимости от истинности условия.

На блок-схеме условный оператор может быть представлен следующим образом:



Пример 1. Даны два числа. Составить программу, которая находит меньшее из них.

```
var a, b, min: real; // min — для меньшего числа
begin
 read(a, b);
 if a > b then       // если a > b, то меньшее число — b
  min := b           // записываем его в min
 else
  min := a;          // иначе меньшее число — a
 write(min);
end.
```

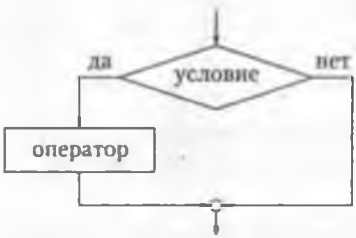
Кроме полной формы условного оператора, приведенной выше, существует и сокращенная.

Схема сокращенной формы условного оператора:

```
if условие then
 оператор;
```

Если условие истинно, то выполнится оператор. Если условие ложно, то оператор выполняться не будет. То есть сокращенная форма условного оператора отличается от полной отсутствием ветви **else**.

На блок-схеме сокращенная форма условного оператора может быть представлена следующим образом:



Пример 2. Дано целое число. Написать программу, которая заменяет его на 0, если число отрицательное, и выводит его без изменений в противном случае.

```
var a: integer;
begin
 read(a);
 if a < 0 then
  a := 0;
 write(a);
end.
```

Пример 3. Даны три целых числа. Написать программу, которая подсчитывает количество чисел, равных нулю в данном наборе.

```
var a, b, c, k: integer;
            // k – количество чисел, равных нулю
begin
 read(a, b, c);
 k := 0;    // ни одно число, равное 0, не найдено
 // если a = 0, увеличиваем k на 1
 if a = 0 then k := k + 1;
 // если b = 0, увеличиваем k на 1
 if b = 0 then k := k + 1;
 // если c = 0, увеличиваем k на 1
 if c = 0 then k := k + 1;
 write(k);
end.
```

Перед проверкой чисел на равенство нулю в переменную k заносится число 0. Это необходимо, поскольку в следующих строках k увеличивается на единицу: $k := k + 1$. То есть в k заносится то, что было в k плюс один. А что было в k? Чтобы не возникало такой неопределенности, и происходит задание начального значения переменной k.

Обратите внимание, что все условные операторы выполняются независимо друг от друга, т. е. переменная b сравнивается с нулем вне зависимости от значения переменной a, а переменная c сравнивается с нулем вне зависимости от значений переменных a и b. Каждый из условных операторов либо увеличит значение k на единицу, либо не увеличит.

Часто необходимо, чтобы в случае истинности или ложности условия выполнялась не одна команда, а несколько. В таком случае требуется группу команд написать внутри составного оператора:

```
begin операторы end;
```

Пример 4. Даны три целых числа. Написать программу, выводящую те из них, которые больше нуля, и подсчитывающую их произведение.

```
var a, b, c, p: integer;
begin
 read(a, b, c);
 p := 1;
 if a > 0 then
 begin
  p := p * a;
  writeln(a);
 end;
 if b > 0 then
 begin
  p := p * b;
  writeln(b);
 end;
 if c > 0 then
 begin
  p := p * c;
  writeln(c);
 end;
 write(p);
end.
```

Коротко о главном

Ветвление — это такая форма организации действий, при которой в зависимости от выполнения (истинности) или невыполнения (ложности) условия выполняется одна либо другая последовательность действий.

Существует две формы оператора **if**: полная и сокращенная.

Схема полной формы оператора **if**:

```
if условие then
 оператор1
else
 оператор2;
```

Схема сокращенной формы оператора **if**:

```
if условие then
 оператор;
```

Для объединения нескольких команд используется составной оператор. Его схема:

```
begin операторы end;
```