В. М. Гуровиц (gurovicmccme.ru), *Записки уроков по программированию для восьмых классов в ЦО 218 и в ФМШ 2007 г. Москвы, версия 26 сентября 2008 г.*

В программах, которые мы писали до сих пор, каждый оператор выполнялся один раз. Для того, чтобы написать большую программу, совершающую миллионы операций, пришлось бы писать миллионы строк кода, если бы не операторы, которые позволяют повторять те или иные действия нужное количество раз. Их называют циклическими операторами или просто циклами.

Цикл for

Покажем, как можно вывести на экран числа от 1 до 20:

```
for i:=1 to 20 do
  writeln(i);
```

Дословный перевод звучит так:

```
для i от 1 до 20 выполняй
  печатай i
```

Таким образом, данный фрагмент программы будет выводить на экран (в столбик, поскольку использован оператор `writeln`) последовательно натуральные числа от 1 до 20. Здесь мы воспользовались новым оператором, который называется оператором цикла — `for`. Работает он следующим образом. Переменной `i` (можно использовать переменную с любым именем) присваивается начальное значение 1 (опять же, можно указать и любое другое начальное значение). Затем выполняется оператор, который указан после слова `do` (это может быть абсолютно любой оператор, как и в случае с условным оператором). Затем значение переменной `i` увеличивается на 1 (это наш «счетчик цикла»). Затем опять выполняется оператор после `do`. Так повторяется до тех пор, пока значение переменной `i` не превысит конечного значения (в данном случае это число 20). Таким образом, оператор цикла `for` повторяет вложенный в него оператор, изменяя каждый раз значение счетчика в заданных пределах: от начального до конечного.

Общий вид этого оператора такой:

```
for имя_переменной := начальное_значение to конечное_значение do
  оператор;
```

Приведем еще один пример:

```
for i:=1 to 100 do
  write(1);
```

Эта программа сто раз напечатает... число 1. Вы спросите: а зачем тогда нужна переменная `i`. Видимо на этот вопрос нужно ответить так: чтобы программа «не сбилась со счету». В данном случае вы никак не используете значение этой переменной, оно нужно лишь для того, чтобы программа, постоянно сравнивая это значение с конечным, вовремя остановилась.

Приведем более сложный пример:

```
for i:=1 to 100 do
  if i mod 2 = 0 then
    writeln(i);
```

Эта программа перебирает все числа от 1 до 100, и печатает только четные. Здесь внутри оператора `for` выполняется условный оператор, внутри которого выполняется оператор `write`. Эту же программу можно написать по-другому:

```
for i:=1 to 50 do
  write(2*i);
```

В этом случае мы перебираем все числа от 1 до 50, каждое умножаем на 2 и выводим получившееся произведение. Есть и третий вариант:

```
m:=0;
for i:=1 to 50 do begin
  m:=m+2;
  write(m);
end;
```

Здесь мы заводим новую переменную `m`, которая изначально равна нулю, а на каждом шаге увеличивается на 2. Поскольку внутри цикла выполняются два оператора, то они окружены ключевыми словами `begin` — `end`: эта конструкция используется с оператором цикла точно так же, как и с условным оператором.

Заметим, что программа вида

```
for i:=1 to 3 do
  write(3);
```

полностью эквивалентна такой программе:

```
i:=1;
write(i);
i:=i+1;  {i=2}
write(i);
i:=i+1;  {i=3}
write(i);
```

Поэтому может показаться, что цикл не привносит ничего нового в программу, а лишь сокращает ее объем. Это не так, и вот тому пример:

```
read(n);
for i:=1 to n do
  writeln(i);
```

Эта программа выводит числа от `1` до `n`, где `n` задается пользователем, то есть не известно до начала выполнения программы. Написать аналогичную программу без использования цикла вам не удастся.