Урок 4. Программирование циклов

Цель работы: изучить определение, назначение и виды циклов, приобрести навыки в решении задач с помощью операторов цикла, усвоить правила и особенности их применения.

В C/C++ существуют все три типа операторов цикла: цикл с предусловием, цикл с постусловием и цикл с параметром.

Цикл с предусловием. Формат оператора цикла с предусловием:

```
while(выражение) оператор;
```

Цикл повторяет свое выполнение, пока значение выражения отлично от нуля, т. е. заключенное в нем условие цикла ***истинно***.

В качестве примера использования оператора цикла рассмотрим программу вычисления факториала целого положительного числа N!.

Пример 1.

```
//  Программа  вычисления факториала
#include  <iostream>
using namespace std;
void  main( )
{
long  int  F;
int  i, N;
cout <<"N =";  cin >> N;
F = i = 1;
while(i <= N) F = F * i++;
cout << "\n" << N <<"!= " << F;
}
```

Обратите внимание на операторы в теле цикла. В программе можно было написать два оператора присваивания, объединив их фигурными скобками. Однако использованный способ записи более лаконичен и более характерен для C/C++.

Этот же самый оператор можно было записать еще короче: F *= i++. При практическом использовании этой программы не следует забывать, что факториал — очень быстро растущая функция, и поэтому при определенных значениях N выйдет из диапазона, соответствующего типу long int. Задав для переменной F тип unsigned long, можно сдвинуть эту границу, но этого может оказаться недостаточно. Предлагаем в качестве самостоятельного задания исследовать предельные значения N для двух указанных типов переменной F. Интересно свойство следующего оператора:

```
while(1);
```

Это бесконечный пустой цикл. Использование в качестве выражения константы 1 приводит к тому, что условие повторения цикла все время остается истинным и работа цикла никогда не заканчивается. Тело в этом цикле представляет собой ***пустой оператор***. При исполнении такого оператора программа будет «топтаться на месте».

Рассмотрим еще один пример использования оператора цикла while. Вернемся к задаче итерационного вычисления суммы гармонического ряда: 1 + 1/2 + 1/3 +... с заданной точностью e.

Пример 2.

```
// Сумма гармонического  ряда
#include  <iostream>
#include  <limits.h>
using namespace std;
void  main( )
{
int  n = 1;
double  S = 0,  eps;
cout << "Точность:";
cin >> eps;
while(1.0/n > eps  && n < INT_MAX)
S += l./n++;
cout << "\nСумма =" << S;
}
```

Файл limits.h, подключаемый препроцессором, содержит определения предельных констант для целых типов данных. В частности, константа с именем INT_MAX равна максимальному значению типа int в данной реализации компилятора. Если для типа int используется двухбайтовое представление, то INT_MAX = 32767.
В этом же заголовочном файле определены и другие константы:

```
INT_MIN = -32768;       LONG_MAX = 2147483647  и т.д.
```

Цикл с постусловием. Формат оператора цикла с постусловием:

```
do оператор while (выражение);
```

Цикл выполняется до тех пор, пока `выражение` отлично от нуля, т.е. заключенное в нем условие цикла истинно. Выход из цикла происходит после того, как значение выражения станет ложным, иными словами равным нулю. Таким образом, в операторе do ... while в C/C++ в конце пишется условие повторения цикла. В качестве примера рассмотрим программу вычисления N!, в которой используется цикл с постусловием.

Пример 3.

```
// Программа вычисления факториала
#include <iostream>
using namespace std;
void main( )
{
long int F;
int i, N;
cout << "N ="; cin >> N;
F = i = 1;
do  F *= i++;
while (i <= N);
cout << "\n" << N << "!=" << F;
}
```

Цикл с параметром. Формат оператора цикла с параметром:

```
for(выражение_1; выражение_2; выражение_3) оператор;
```

Выражение 1 выполняется только один раз в начале цикла. Обычно оно определяет начальное значение параметра цикла (инициализирует параметр цикла). Выражение 2

— это условие выполнения цикла. Выражение 3 обычно определяет изменение параметра цикла, оператор — тело цикла, которое может быть простым или составным. В последнем случае используются фигурные скобки.

Алгоритм выполнения цикла for представлен на блок-схеме

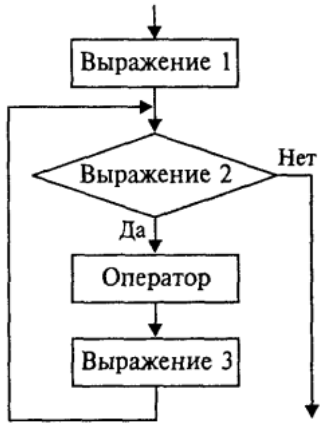

Рис. 1. Алгоритм выполнения цикла for

Обратите внимание на то, что после вычисления выражения 3 происходит возврат к вычислению выражения 2 — проверке условия повторения цикла.
С помощью цикла for нахождение N! можно организовать следующим образом:

```
F = 1 ;
for(i = 1; i <= N; i++)  F = F * i;
```

Используя операцию «запятая», можно в выражение 1 внести инициализацию значений сразу нескольких переменных:

```
for(F = 1, i = 1; i <= N; i++)  F = F * i;
```

Некоторых элементов в операторе может не быть, однако разделяющие их точки с запятой обязательно должны присутствовать. В следующем примере инициализирующая часть вынесена из оператора for:

```
F = 1;
i = 1 ;
for(; i <= N; i++)  F = F * i;
```

Ниже показан еще один вариант вычисления N!. В нем на месте тела цикла находится пустой оператор, а вычислительная часть внесена в выражение 3.

```
for(F = 1, i = 1; i <= N; F = F * i, i++);
```

Этот же оператор можно записать в следующей форме:

```
for(F = 1, i = 1; i <= N; F *= i++);
```

В языке C/C++ оператор for является достаточно универсальным средством для организации циклов. С его помощью можно программировать даже итерационные циклы, что невозможно в Паскале. Вот пример вычисления суммы элементов гармонического ряда, превышающих заданную величину e:

```
for(n = 1, S = 0; 1.0/n > eps && n < INT_MAX; n++)  S+=1.0/n;
```

И наконец, эта же самая задача с пустым телом цикла:

```
for(n = 1, S = 0; 1.0/n > eps && n < INT_MAX; S += 1.0/n++);
```

Следующий фрагмент программы на C/C++ содержит два вложенных цикла for. В нем запрограммировано получение на экране таблицы умножения.

```
for(x = 2; x <= 9; x++)
      for(y = 2; y <= 9; y++)
            cout << "\n" << x << " * " << y << " = "<< x * y;
```

На экране будет получен следующий результат:

```
2 * 2 = 4
2 * 3 = 6
……
9 * 8 = 72
9 * 9 = 81
```

Оператор continue. Если выполнение очередного шага цикла требуется завершить до того, как будет достигнут конец тела цикла, используется оператор continue. Следующий фрагмент программы обеспечивает вывод на экран всех четных чисел в диапазоне от 1 до 100.

```
for( i = 1; i <= 100; i++)
{
if(i%2) continue; cout << "\t" << i;
}
```

Для нечетных значений переменной i остаток от деления на 2 будет равен единице, этот результат воспринимается как значение «истина» в условии ветвления, и выполняется оператор continue. Он завершит очередной шаг цикла, выполнение цикла перейдет к следующему шагу.

Оператор goto. Оператор безусловного перехода goto существует в языке C/C++, как и во всех других языках программирования высокого уровня. Однако с точки зрения структурного подхода к программированию его использование рекомендуется ограничить. Формат оператора:

```
goto  метка;
```

Метка представляет собой идентификатор с последующим двоеточием, ставится перед помечаемым оператором. Одна из ситуаций, в которых использование goto является оправданным — это необходимость «досрочного» выхода из вложенного цикла. Вот пример такой ситуации:

```
for(...)
      {  while  (...)
            {  for(...)
                  {...  goto  exit  ...}
            }
      }
exit: cout << "Выход  из  цикла";
```

При использовании оператора безусловного перехода необходимо учитывать следующие ограничения:

- нельзя входить внутрь блока извне;
- нельзя входить внутрь условного оператора (if ...else...);

• нельзя входить внутрь переключателя;
• нельзя входить внутрь цикла.

Упражнения

1. Используя циклы while, do - while и for, написать три варианта программы получения на экране таблицы синусов для значений аргумента в диапазоне от 0 до $\pi/2$ с заданным числом шагов.
2. Вычислить и вывести все члены числового ряда

   1, 1/2!, 1/3!, …, 1/N!,

   значение которых превышает 10^{-5}.
3. Напечатать в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр.
4. Дано целое n > 2. Напечатать все простые числа из диапазона [2, n].
5. Составить программу перевода целого десятичного числа в двоичную систему счисления.
6. Составить программу перевода целого десятичного числа в шестнадцатеричную систему счисления.

Литература

1. **Семакин И.Г., Шестаков А.П.** Основы программирования: Учебник.- М.: Мастерство, 2002.- 432 с.