Лекция 15

Синтаксис операторов циклов: с параметром, предусловием и постусловием

Цель: Освоить правила применения операторов циклов с параметром, предусловием и постусловием

Операторы цикла

Операторы цикла используются для вычислений, повторяющихся многократно. Блок, ради выполнения которого и организуется цикл, называется *телом цикла*. Для управления процессом повторения вычислений служат: начальные установки, проверка *условия продолжения цикла* и модификация параметра *цикла*. Один проход *цикла* называется *итерацией*.

Начальные установки служат для того, чтобы до входа в цикл задать значения переменных, которые в нем используются.

Операторы цикла

Проверка условия продолжения цикла выполняется на каждой итерации либо до тела *цикла* (тогда говорят о цикле *с предусловием*), либо после тела *цикла* (цикл *с постусловием*).

Параметром цикла называется *переменная*, которая используется при проверке *условия продолжения цикла* и принудительно изменяется на каждой итерации, причем, как правило, на одну и ту же величину.

Если *параметр цикла* целочисленный, он называется *счетчиком цикла*.

Операторы цикла

Цикл завершается, если условие его продолжения не выполняется. Возможно принудительное завершение как текущей итерации, так и *цикла* в целом. Для этого служат *операторы* **break**, **continue**, **return** и **goto**. Передавать управление извне внутрь *цикла* запрещается.

Цикл с предусловием while

Формат оператора:

while (выражение) оператор;

Выражение должно быть логического типа. Если результат вычисления выражения равен **true**, выполняется простой или *составной оператор*. Эти действия повторяются до того момента, пока результатом выражения не станет *значение* **false**. После окончания *цикла* управление передается на следующий за ним оператор.

Цикл с предусловием while

Выражение вычисляется перед каждой итерацией *цикла*. Чтобы значение *выражения* изменилось (с **true** на **false**), необходимо, чтобы хотя бы один из операндов этого выражения (*параметр цикла*) изменял свое значение в теле цикла - это должен предусмотреть программист.

Если при первой проверке *выражение* равно **false**, цикл не выполнится ни разу.

Пример 1

Программа выводит для аргумента *x*, изменяющегося в заданных пределах с заданным шагом, таблицу значений следующей функции:

$$y = \begin{pmatrix} x, & x < 0 \\ tx, & 0 \le x < 10 \\ 2t, & x \ge 10 \end{pmatrix}$$

Пример 1

```
class Class1{
    static void Main(){
     // Xn – начальное, Xk – конечное, dX – шаг, t - значение
        double Xn = -2, Xk = 12, dX = 2, t = 2, y;
        Console.WriteLine( "|   x    |    y   |" ); //заголовок таблицы
        double x = Xn;     // x является параметром цикла
        while (x <= Xk) {
              y = t;
              if ( x >= 0 && x < 10 ) y = t * x;
              if ( x >= 10 )  y = 2 * t;
              Console.WriteLine( "| {0,6} | {1,6} |", x, y );
              x += dX;
        }
    }
}
```

Пример 2

```
using System;
class WhileDemo {// Вычислить порядок величины целого числа
    static void Main() {
        int num;   int mag;   num = 435679;   mag = 0;
        Console.WriteLine("Число: " + num);
        while (num > 0) {
            mag++;
            num = num /10;
        };
        Console.WriteLine("Порядок величины: " + mag);
    }
}
```

Цикл с постусловием do-while

Цикл с постусловием имеет вид:

do оператор **while** (выражение);

Сначала выполняется простой или *составной оператор*, образующий тело *цикла*, а затем вычисляется *выражение* (оно должно иметь тип **bool**). Если *выражение* истинно, тело *цикла* выполняется еще раз, и проверка повторяется. Цикл завершается, когда *выражение* станет равным **false** или в теле *цикла* будет выполнен какой-либо *оператор передачи управления*.

Этот вид *цикла* применяется в тех случаях, когда тело *цикла* необходимо обязательно выполнить хотя бы один раз.

Пример 3

```
using System;
    namespace ConsoleApplication1{
      class Class1{
          static void Main(){
              char answer;  // параметр цикла
              do{
                  Console.WriteLine("Купи слоника, а?");
                  answer = (char) Console.Read();
                  Console.ReadLine();
            } while (answer != 'y');
        }
    }
}
```

Пример 4

```
using System;
class DoWhileDemo { // Отобразить цифры целого числа
    static void Main() { // в обратном порядке
        int num;     int nextdigit;      num = 198;
        Console.WriteLine("Число: " + num);
        Console.Write("Число в обратном порядке: ");
        do {
            nextdigit = num % 10;
            Console.Write(nextdigit);
            num = num / 10;
        } while (num > 0) ;
        Console.WriteLine() ;
    }
}
```

Цикл с параметром for

Цикл с параметром используется, когда известно количество повторений (итераций) и имеет следующий формат:

for (инициализация; выражение; модификации) оператор;

Порядок выполнения:

шаг 1 Выполняется *инициализация*, которая служит для объявления величин, используемых в цикле, и присвоения им начальных значений **один раз** в начале исполнения *цикла*.

В этой части можно записать несколько операторов, разделенных запятой, например:

for (int i = 0, j = 20; ...int k, m;**for** (k = 1, m = 0; ...

Областью действия переменных, объявленных в части инициализации *цикла*, является цикл.

Цикл с параметром for

шаг 2 Проверяется *выражение* типа **bool**, которое определяет условие выполнения *цикла*: **если** его результат равен **true**, *оператор* выполняется, **иначе** цикл *завершает работу*. Простой или составной *оператор* представляет собой *тело цикла*.

шаг 3 Выполняются *модификации* (итерации, приращения), которые служат обычно для изменения параметров *цикла* после каждой итерации *цикла*. В части модификаций можно записать несколько операторов через запятую, например:

for (int i = 0, j = 20; i < 5 && j > 10; i++, j--) ...

Выполняются действия **шага 2.**

Цикл с параметром for

Порядок работы оператора модно проследить на примере:

```
int s = 0;
for (int i = 1; i <= 6; i++) s += i;
Console.WriteLine(" s = " +  s + ",   i = " + i);
```

Итерация 1

шаг 1 Инициализация параметра цикла: i = 1

шаг 2 Проверка выражения (условия):

i <= 6 → **true** → выполнение тела цикла: s += i; // s = 1;

шаг 3 Модификация параметра цикла:

i++   // i = 2 → выполняются действия **шага 2.**

Цикл с параметром for

Итерация 2

шаг 2 Проверка выражения (условия):

i <= 6 → **true** → выполнение тела цикла: s += i; // s = 3;

шаг 3 Модификация параметра цикла:

i++ // i = 3 → выполняются действия **шага 2.**

Итерация 3

шаг 2 Проверка выражения (условия):

i <= 6 → **true** → выполнение тела цикла: s += i; // s = 6;

шаг 3 Модификация параметра цикла:

i++ // i = 4 → выполняются действия **шага 2.**

Итерация 4

шаг 2 Проверка выражения (условия):

i <= 6 → **true** → выполнение тела цикла: s += i; // s = 10;

шаг 3 Модификация параметра цикла:

i++ // i = 5 → выполняются действия **шага 2.**

Цикл с параметром for

Итерация 5

шаг 2 Проверка выражения (условия):

i <= 6 → **true** → выполнение тела цикла: s += i; // s = 15;

шаг 3 Модификация параметра цикла:

i++ // i = 6 → выполняются действия **шага 2.**

Итерация 6

шаг 2 Проверка выражения (условия):

i <= 6 → **true** → выполнение тела цикла: s += i; // s = 21;

шаг 3 Модификация параметра цикла:

i++ // i = 7 → выполняются действия **шага 2.**

Итерация 7 (незавершенная)

шаг 2 Проверка выражения (условия):

i <= 6 → **false** → цикл завершает работу: s = 21, i = 7

Пример 5

```
class Class1{
    static void Main(){
        double Xn = -2, Xk = 12, dX = 2, t = 2, y;
        Console.WriteLine( "|   x    |    y   |");
        for (double x = Xn; x <= Xk; x += dX){
           y = t
           if ( x >= 0 && x < 10 ) y = t * x;
           if ( x >= 10 )          y = 2 * t;
           Console.WriteLine( "| {0,6} | {1,6} |", x, y );
    }
}
```

Пример 6

```
using System;
class DecrFor {
    static void Main() {
        int x;
        for (x = 100; x > -100; x -= 5)
        Console.WriteLine(x);
    }
}
```

Пример 7

```
using System;
class Comma {
    static void Main() {
        int i, j;      int smallest, largest;      int num;
        num = 100;      smallest = largest = 1;
        for (i=2, j=num/2; (i <= num/2) & (j >= 2); i++, j--) {
          if ((smallest == 1) & ((num % i) == 0))  smallest = i;
          if ((largest == 1) & ((num % j) == 0) )  largest = j;
        }
      Console.WriteLine("Наибольший множитель: " + largest); // 50
      Console.WriteLine("Наименьший множитель: " + smallest); // 2
    }
}
```

Пример 8

```
using System;
class forDemo {
    static void Main() {
        int i, j;
        bool done = false;
        for (i=0, j=100; !done; i++, j--) {
            if (i*i >= j) done = true;
            Console.WriteLine("i, j: " + i + " " + j);
        }
    }
}
```

Пример 9

```
static void Main() {
      int e;    int result;    int i=0;
      for ( ; i < 10; i++) {
         result = 1;
         e = i;
         while(e > 0) {
             result *= 2;
             e--;
         }
         Console.WriteLine ("2 в степени " + i + " равно "
    + result);
     }
}
```

Пример 10

```
using System;
class Empty2 {
    static void Main() {
        int i;
        i = 0;  // исключить инициализацию из
                // определения цикла
        for ( ; i < 10; ) {
            Console.WriteLine("Проход №" + i);
            i++; // инкрементировать переменную
                 // управления циклом
        }
    }
}
```

Цикл с параметром for

Если оставить пустым выражение условия в операторе цикла **for**, то получится **бесконечный цикл**, т.е. такой цикл, который никогда не заканчивается. Если условие отсутствует, то оно считается истинным.

```
for ( ; ; ) // цикл будет выполняться бесконечно
{
        //...
}
```

Этот цикл будет выполняться бесконечно. Выход из такого цикла программист должен предусмотреть, например, используя оператор **break**.

Пример 11

```
using System;
namespace ConsoleApplication1{
 class Class1{
  static void Main(){
    int ch, a, b;
    for ( ; ; ) {
      do {    Console.Clear();       // очистка экрана
              Console.WriteLine( "1. Ввод данных");
              Console.WriteLine( "2. Расчет суммы");
              Console.WriteLine( "3. Расчет произведения");
              Console.WriteLine( "4. Выход");
              Console.WriteLine( "Выберите пункт меню");
              ch = Convert.ToInt32(Console.ReadLine());
    } while(ch != 1  &&   ch != 2  &&  ch != 3 &&  ch != 4);
```

Пример 11

```
switch(ch) {
   case 1: Console.Write( "Введите целое число - ");
      a = Convert.ToInt32(Console.ReadLine());
      Console.Write( "Введите целое число - ");
      b = Convert.ToInt32(Console.ReadLine());
      break;
  case 2: Console.WriteLine(a+" + "+b+" = "+(a+b));
     Console.ReadKey();  // остановка экрана
     break;
  case 3:  Console.WriteLine(a+" * "+b+" = "+(a*b));
     Console.ReadKey();
     break;
```

Пример 11

```
            case 4:  Environment.Exit(0);  /* Возврат в
   операционную систему */
                break;
          } // end switch(ch)
        }  // end for ( ; ; )
      }  // end Main
    }  // end class Class1
}  // end namespace ConsoleApplication1
```

Пример 12

```
using System;
class Empty3 {
    static void Main() {
        int i;
        int sum = 0;
        for (i = 1; i <= 5; sum += i++ );
        Console.WriteLine("Сумма равна " +
  sum); // 15
    }
}
```

Цикл перебора foreach

Оператор цикла **foreach** служит для циклического обращения к элементам коллекции, которая представляет собой группу объектов. В **C#** определено несколько видов коллекций, к числу которых относится массив, список или другой контейнер.

Общая форма оператора цикла **foreach**:

foreach (тип имя_переменной_цикла **in** коллекция) оператор;

Цикл перебора foreach

тип имя_переменной_цикла обозначает тип и имя переменной управления циклом, которая получает значение следующего элемента коллекции на каждом шаге выполнения цикла **foreach**. А *коллекция* обозначает циклически опрашиваемую коллекцию, которая может представлять собой, например, массив. Следовательно, тип переменной цикла должен соответствовать типу элемента массива.

Кроме того, тип может обозначаться ключевым словом **var**. В этом случае компилятор определяет тип переменной цикла, исходя из типа элемента массива. Это может оказаться полезным для работы с определенного рода запросами.

Но, как правило, тип указывается явным образом.

Цикл перебора foreach

Оператор цикла **foreach** действует следующим образом.

Когда цикл начинается, первый элемент массива выбирается и присваивается переменной цикла.

На каждом последующем шаге итерации выбирается следующий элемент массива, который сохраняется в переменной цикла.

Цикл завершается, когда все элементы массива окажутся выбранными.

Следовательно, оператор **foreach** циклически опрашивает массив по отдельным его элементам от начала и до конца.

Цикл перебора foreach

Следует, однако, иметь в виду, что переменная цикла в операторе **foreach** служит только для чтения.

Это означает, что, присваивая этой переменной новое значение, нельзя изменить содержимое массива.

Оператор **foreach** можно завершить преждевременно, воспользовавшись оператором **break** или **return**.

Пример 13

```
using System;
class ForeachDemo {
    static void Main() {
        int sum = 0;
        int[] nums = new int [10];
        // Задать первоначальные значения элементов массива nums
        for (int i = 0; i < 10; i++)  nums[i] = i;
        foreach (int x in nums) {
          Console.WriteLine("Значение элемента равно: " + x);
          sum += x;
          if (x == 4) break; // прервать цикл, если индекс = 4
        }
      Console.WriteLine("Сумма первых 5 элементов: "+sum);
  }
}
```

Пример 14

```
using System;
class ForeachDemo2 {
    static void Main() {
        int sum = 0;
        int[ , ] nums = new int [3,5];
       // Задать первоначальные значения элементов массива nums
        for (int i = 0; i < 3; i++)
            for (int j=0; j < 5; j++)
                nums[i,j] = (i+1)* (j + 1);
        foreach (int x in nums) {
            Console.WriteLine("Значение элемента равно: " + x) ;
            sum += x;
        }
        Console.WriteLine("Сумма равна: " + sum);
    }
}
```

Пример 15

```
using System;
class Search {
    static void Main() {   // Поиск в массиве
        int[] nums = new int[10];
        int val;
        bool found = false;
        for (int i = 0; i < 10; i++)  nums[i] = i;
        val = 5;
        foreach (int x in nums) {
            if (x == val) {
                found = true;
                break;
            }
        }
        if (found) Console.WriteLine("Значение найдено!");
        }
}
```

Контрольные вопросы

1 Какие варианты цикла **for** вы знаете?

2 В каких случаях применяются циклы с предусловием и постусловием?

3 Какие ограничения имеет оператор **foreach** по сравнению с оператором **for**?

Домашнее задание

Напишите программу, позволяющую пользователю с помощью пунктов меню найти сумму, разность и остаток от целочисленного деления двух целых чисел.