Операторы цикла

Операторы цикла используются для организации многократно повторяющихся вычислений. К операторам цикла относятся: *цикл с предусловием* while, *цикл с постусловием* do while, цикл с параметром for и цикл перебора foreach.

Цикл с предусловием while

Оператор цикла while организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Формат цикла while:

while (B) S;

где B - выражение, истинность которого проверяется (условие завершения цикла); S - тело цикла - оператор (простой или составной).

Перед каждым выполнением тела цикла анализируется значение выражения B: если оно истинно, то выполняется тело цикла, и управление передается на повторную проверку условия B ; если значение B ложно - цикл завершается и управление передается на оператор, следующий за оператором S.

Если результат выражения B окажется ложным при первой проверке, то тело цикла не выполнится ни разу. Отметим, что если условие B во время работы цикла не будет изменяться, то возможна ситуация зацикливания, то есть невозможность выхода из цикла. Поэтому внутри тела должны находиться операторы, приводящие к изменению значения выражения B так, чтобы цикл мог корректно завершиться.

В качестве иллюстрации выполнения цикла while рассмотрим программу вывода на экран целых чисел из интервала от 1 до n.

```
static void Main()
{
        Console.Write("N= ");
        int n=int.Parse(Console.ReadLine());
        int i = 1;
        while (i <= n)          //пока i меньше или равно n
        Console.Write(" "+ i++ ); //выводим i на экран, затем увеличиваем его на 1
}
```

Результаты работы программы:

n	ответ
10	1 2 3 4 5 6 7 8 9 10

Цикл с постусловием do while

Оператор цикла do while также организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Однако в отличие от цикла while условие завершения цикла проверяется после выполнения тела цикла. Формат цикла do while:

do S while (B);

где B - выражение, истинность которого проверяется (условие завершения цикла); S - тело цикла - оператор (простой или блок).

Сначала выполняется оператор S, а затем анализируется значение выражения B: если оно истинно, то управление передается оператору S, если ложно - цикл завершается, и управление передается на оператор, следующий за условием B. Так как условие B проверяется после выполнения тела цикла, то в любом случае тело цикла выполнится хотя бы один раз.

В операторе do while, так же как и в операторе while, возможна ситуация зацикливания в случае, если условие B всегда будет оставаться истинным.

В качестве иллюстрации выполнения цикла do while рассмотрим программу вывода на экран целых чисел из интервала от 1 до n.

```
static void Main()
{
        Console.Write("N= ");
        int n=int.Parse(Console.ReadLine());
        int i = 1;
        do
        Console.Write(" " + i++);
        //выводим i на экран, затем увеличиваем его на 1
        while (i <= n); //пока i меньше или равно n
}
```

Цикл с параметром for

Цикл с параметром имеет следующую структуру:

for (<инициализация>; <выражение>; <модификация>) <оператор>;

Инициализация используется для объявления и/или присвоения начальных значений величинам, используемым в цикле в качестве параметров (счетчиков). В этой части можно записать несколько операторов, разделенных запятой. Областью действия переменных, объявленных в части инициализации цикла, является цикл и *вложенные блоки*. Инициализация выполняется один раз в начале исполнения цикла.

Выражение определяет условие выполнения цикла: если его результат истинен, цикл выполняется. Истинность выражения проверяется перед каждым выполнением тела цикла, таким образом, цикл с параметром реализован как *цикл с предусловием*. В блоке выражение через запятую можно записать несколько логических выражений, тогда запятая равносильна операции логическое И (&&).

Модификация выполняется после каждой итерации цикла и служит обычно для изменения параметров цикла. В части модификация можно записать несколько операторов через запятую.

Оператор (простой или составной) представляет собой тело цикла.

Любая из частей оператора *for* (инициализация, выражение, модификация, оператор) может отсутствовать, но точку с запятой, определяющую позицию пропускаемой части, надо оставить.

```
static void Main()
{
        Console.Write("N= ");
        int n=int.Parse(Console.ReadLine());
        for (int i=1; i<=n;)     //блок модификации пустой
        Console.Write(" " + i++);
}
```

Вложенные циклы

Циклы могут быть простые или вложенные (кратные, циклы в цикле). Вложенными могут быть циклы любых типов: while, do while, for. Каждый внутренний цикл должен быть полностью вложен во все внешние циклы. "Пересечения" циклов не допускаются.

Рассмотрим пример использования вложенных циклов, который позволит вывести на экран числа следующим образом:

```
static void Main()
{
for (int i = 1; i <= 4; ++i)
        {
                Console.WriteLine())     //1
                for (int j=1; j<=5; ++j)
                        Console.Write(" " + 2);
        }
}
```

Замечание. В строке 1 в блоке модификации содержится два оператора ++i и Console.WriteLine(). В данном случае после каждого увеличения параметра i на 1 курсор будет переводиться на новую строку.

Оператор безусловного перехода goto

Оператор безусловного перехода goto имеет формат:

goto <метка>;

В теле той же функции должна присутствовать ровно одна конструкция вида:

<метка>: <оператор>;

Оператор *goto* передает управление на помеченный меткой оператор. Рассмотрим пример использования оператора goto:

```
static void Main()
{
        float x;
        metka: Console.WriteLine("x="); //оператор, помеченный меткой
        x = float.Parse(Console.ReadLine());
        if (x!=0) Console.WriteLine("y({0})={1}", x, 1 / x );
        else
        {
                Console.WriteLine("функция не определена");
                goto metka; // передача управление метке
        }
}
```

Оператор выхода break

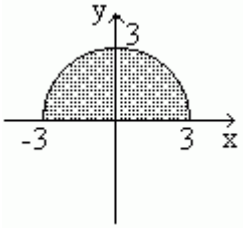
Оператор break используется внутри операторов ветвления и цикла для обеспечения перехода в точку программы, находящуюся непосредственно за оператором, внутри которого находится break.

Оператор перехода к следующей итерации цикла continue

Оператор перехода к следующей итерации цикла continue пропускает все операторы, оставшиеся до конца тела цикла, и передает управление на начало следующей итерации (повторение тела цикла). Рассмотрим оператор continue на примере.

```
static void Main()
{
        Console.WriteLine("n=");
        int n = int.Parse(Console.ReadLine());
        for (int i = 1; i <= n; i++)
        {
                if (i % 2 == 0) continue;
                Console.Write(" " + i);
        }
}
```

Пример. Дана точка на плоскости с координатами (x, y). Составить программу, которая выдает одно из сообщений "Да", "Нет", "На границе" в зависимости от того, лежит ли точка внутри заштрихованной области, вне заштрихованной области или на ее границе.




```
using System;
namespace Hello
{
        class Program
        {
                static void Main()
                {
                        Console.Write("x=");
                        float x = float.Parse(Console.ReadLine());
                        Console.Write("y=");
                        float y = float.Parse(Console.ReadLine());
                        if (x * x + y * y < 9 && y > 0)
                        Console.WriteLine("внутри");
                        else if (x * x + y * y > 9 || y < 0)
                        Console.WriteLine("вне");
                        else Console.WriteLine("на границе");
```

```
                }
            }
        }
```

Пример 2. Составить программу. Дан порядковый номер дня недели, вывести на экран его название.

```
using System;
namespace Hello
{
    class Program
    {
        static void Main()
        {
            Console.Write("n=");
            byte n = byte.Parse(Console.ReadLine());
            switch (n)
            {
                case 1: Console.WriteLine("понедельник"); break;
                case 2: Console.WriteLine("вторник"); break;
                case 3: Console.WriteLine("среда"); break;
                case 4: Console.WriteLine("четверг"); break;
                case 5: Console.WriteLine("пятница"); break;
                case 6: Console.WriteLine("суббота"); break;
                case 7: Console.WriteLine("воскресенье"); break;
                default: Console.WriteLine("ВЫ ОШИБЛИСЬ"); break;
            }
        }
    }
}
```

Пример 3. Вывести на экран: целые числа 1, 3, 5, ..., 21 в строчку через пробел:

```
using System;
namespace Hello
{
	class Program
	{
	static void Main()
		{
			Console.Write("n=");
			byte n = byte.Parse(Console.ReadLine());
			Console.Write("while: ");
			int i = 1;
			while (i <= n)
			{
				Console.Write(" " + i);
				i += 2;
			}

			Console.Write("do while: ");
			i = 1;
			do
			{
				Console.Write(" " + i);
				i += 2;
			}
```

```
                while (i <= n);

                Console.Write("For: ");
                for (i = 1; i<=n; i+=2)
                {
                        Console.Write(" " + i);
                }
        }
    }
}
```