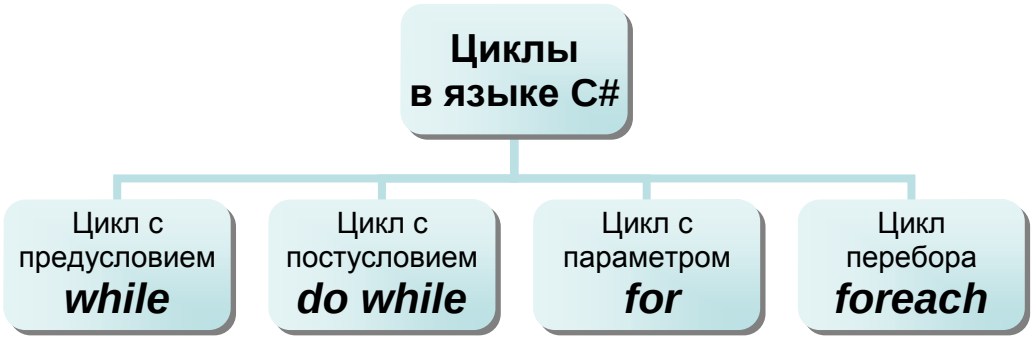
Операторы цикла
(Итерационные операторы в C#)

Итерационные операторы применяются в программах на C# для выполнения каких-либо повторяющихся действий, т.е. для организации циклов. Иногда эти операторы называют *циклическими*. К итерационным операторам в C# относятся операторы `for, while, do, foreach.`



Оператор for

Оператор for предназначен для повторного выполнения оператора или группы операторов заданное количество раз. Вот как выглядит этот оператор в общем виде:

`for ([Инициализация] ; [Условие] ; [Приращение])<Оператор>`

Инициализация используется для объявления и/или присвоения начального значения величине, используемой в цикле в качестве параметра (счетчика). Областью действия переменной, объявленной в части инициализации цикла, является цикл и вложенные блоки.

[**Инициализация**] выполняется один раз перед началом цикла.

Условие или логическое выражение определяет условие выполнения цикла: если его результатом является истина, цикл выполняется. Истинность выражения проверяется перед каждым выполнением тела цикла, таким образом, цикл с параметром реализован как цикл с предусловием.

Перед каждой итерацией (т.е. перед каждым выполнением тела цикла <**Оператор**>) проверяется [**Условие**].

И, наконец, после каждой итерации выполняется оператор [**Приращение**].

Приращение (модификация) выполняется после каждой итерации цикла и служит обычно для изменения параметра цикла.

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.

Как правило, в теле цикла имеется переменная, играющая роль так называемой *переменной цикла*. При каждой итерации переменная цикла изменяет свое значение в заданных пределах.

Начальное значение переменной цикла задается в программе до оператора **for** или в операторе [**Инициализация**]. Предельное значение переменной цикла определяется оператором приращения, а проверка ее текущего значения – в блоке [**Условие**].

Пример. Составим цикл для вывода на экран 15 целых чисел (от 0 до 14):

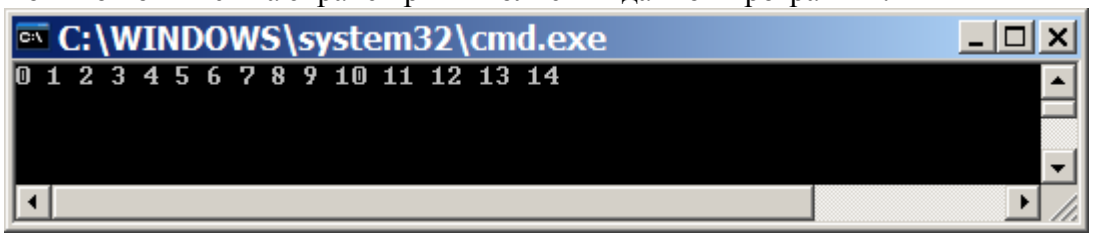
```
int i;
for (i = 0; i < 15; i++)
  {
    Console.Write("{0} ",i);
  }
Console.Read();
```

Здесь переменная `i` используется в качестве переменной (параметра) цикла. Перед началом цикла ей присваивается нулевое значение. Перед каждой итерацией содержимое переменной `i` сравнивается с числом 15. Если число `i` меньше 15, то тело цикла выполняется один раз (т.е. метод `Write` отображает на экране текущее значение переменной `i`).

После выполнения тела цикла значение `i` увеличивается на 1 в блоке приращения (`i++`). Далее переменная цикла вновь сравнивается с числом 15. Когда значение `i` превысит 15, цикл завершится.

Таким образом, параметр цикла анализируется перед выполнением цикла, а изменяется после его выполнения.

Вот что появится на экране при выполнении данной программы:



Пример. Программа выводит числа от 1 до n, которое вводится с клавиатуры:

```
static void Main()
{
  Console.Write("N= ");
  int n=int.Parse(Console.ReadLine());
  for (int i=1; i<=n; i++)
  {
     Console.Write(" {0} ,}",  i);
  }
}
```

Задание. Измените программу так, чтобы:
1. решалась поставленная задача, а блок модификации оказался пустым;
2. числа выводились в обратном порядке;
3. выводились квадраты чисел.

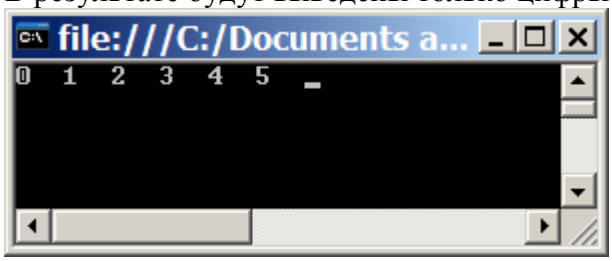
Прерывание цикла

С помощью оператора **break** можно в любой момент прервать выполнение цикла. Например, в следующем примере работа цикла прерывается, когда значение переменной цикла становится больше пяти.

Пример.

```
int i;
for (i = 0; i < 15; i++)
  {
    if (i>5)
      break;
    Console.Write("{0} ",i);
  }
Console.Read();
```

В результате будут выведены только цифры от 0 до 5:



Оператор **break** в паре с условным оператором может заменить блок проверки условия в операторе for.

```
for (i = 0;; i++)
  {
    if (i > 10)
      break;
    System.Console.Write("{0} ",i);
  }
```

Как видите, здесь пропущена проверка условия – она выполняется оператором `if` внутри цикла.

Далее можно опустить все блоки оператора `for`, выполняя инициализацию, проверку условия, а также итерацию самостоятельно:

```
int i = 0;
for (;;)
  {
    if (i > 10)
      break;
    System.Console.Write("{0} ",i);
    i++;
  }
```

Это позволяет реализовывать любую необходимую логику циклической обработки. Например, можно изменять значение переменной цикла перед итерацией, а не после нее:

```
for (i = 0; i > 10;)
  {
    i++;
    System.Console.Write("{0} ",i);
  }
```

При создании цикла обязательно нужно предусмотреть условие его завершения. Если этого не сделать, цикл будет выполняться бесконечно. Программа при этом будет работать вхолостую на одном месте – «зациклится».

Вот пример цикла, у которого нет выхода:

```
for (i = 0;;)   //зацикливание!
  {
    i++;
    System.Console.Write("{0} ",i);
  }
```

Здесь не предусмотрена проверка значений переменной цикла, поэтому программа будет постоянно выводить на консоль возрастающие значения, пока вы не прервете ее работу нажатием клавиш Control+C или закрыв консольное окно.

Возобновление цикла

В отличие от оператора **break**, прерывающего цикл, оператор **continue** позволяет возобновить выполнение цикла с самого начала.

Пример.

```
int i;
for (i = 0; ; i++)
  {
    Console.Write("{0} ",i);
    if (i<9)
      continue;
    else
      break;
   }
Console.Read();
```

Если в ходе выполнения цикла значение переменной `i` не достигла девяти, цикл возобновляет свою работу с самого начала (т.е. с вывода значения переменной цикла на консоль). Когда указанное значение будет достигнуто, то выполнение цикла прервется оператором `break`.

Оператор 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 (простой или составной оператор).

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 внутри цикла не будет оператора (или операторов), в результате выполнения которых условие B на какой-то итерации станет ложным, то произойдет зацикливание, 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
    {
       //выводим i на экран, затем увеличиваем его на 1
       Console.Write(" {0}", i++ );
    }
}
```

Задание. Измените программу так, чтобы:
1. числа выводились в обратном порядке;
2. выводились только нечетные числа.

Оператор `while` проверяет условие завершения цикла перед выполнением тела цикла:

```
i = 0;
while(i < 10)
{
   System.Console.Write("{0} ",i );
   i++;
}
```

В отличие от оператора `for` оператор `while` никак не изменяет значения переменной цикла, поэтому мы должны позаботиться об этом сами.

Перед тем как приступить к выполнению цикла, мы устанавливаем начальное значение параметра цикла `i`, равное нулю. После выполнения тела цикла мы сами изменяем значение параметра цикла, увеличивая его на единицу.

Цикл будет прерван, как только значение переменной `i` превысит 10.

В цикле `while` можно использовать описанные ранее операторы прерывания цикла `break` и возобновления цикла `continue`.

Следующий цикл будет выполняться бесконечно:

```
while(true)
{
   System.Console.Write("{0} ",i );
   i++;
}
```

Еще раз напомннаем вам, что нужно всегда предусматривать возможность выхода из цикла.

Оператор do

Оператор do используется вместе с ключевым словом while. При этом условие завершения цикла проверяется после выполнения его тела:

```
i = 0;
do
{
   System.Console.Write("{0} ",i );
   i++;
} while(i < 10);
```

В этом цикле мы сами устанавливаем начальное значение параметра цикла i и сами его изменяем, увеличивая на единицу. Как только это значение достигнет 10, цикл будет прерван.

Аналогично циклу while цикл do допускает прерывание оператором break и возобновление оператором continue.

Оператор foreach

Для обработки таких типов данных, как *массивы* и *контейнеры*, язык C# предлагает очень удобный оператор foreach, для которого нет аналога в языках программирования C и C++.

Примеры решения практических задач

Задача 1. Написать программу, которая выводит на экран квадраты всех четных чисел из диапазона от A до B (A и B целые числа, при этом A≤ B).

Указания по решению задачи. *Из диапазона целых чисел от A до B необходимо выбрать только четные числа. Напомним, что четными называются числа, которые делятся на два без остатка. Кроме того, четные числа представляют собой упорядоченную последовательность, в которой каждое число отличается от предыдущего на 2. Решить эту задачу можно с помощью каждого оператора цикла.*

```
using System;

namespace Example
{
   class Program
   {
      static void Main()
      {
         Console.Write("a= ");
         int a=int.Parse(Console.ReadLine());
         Console.Write("b= ");
         int b=int.Parse(Console.ReadLine());
         int i;
```

```
            Console.Write("FOR: ");
            a=(a%2==0)? a: a+1;
            for (i=a; i<=b; i+=2)
            {
                Console.Write(" {0}", i*i);
            }
            Console.Write("\nWHILE: ");
            i=a;
            while (i<=b)
            {
                Console.Write(" {0}", i*i);
                i+=2;
            }

            Console.Write("\nDO: ");
            i=a;
            do
            {
                Console.Write(" {0}", i*i);
                i+=2;
            }
            while (i<=b);
        }
    }
}
```

Результаты работы программы:

a	b	сообщение на экране
-11	11	FOR: 100 64 36 16 4 0 4 16 36 64 100
		WHILE: 100 64 36 16 4 0 4 16 36 64 100
		DO: 100 64 36 16 4 0 4 16 36 64 100

Задание. Измените программу так, чтобы она выводила на экран только нечетные числа из диапазона от A до B (A и B целые числа, при этом A≤B).

Задача 2. Постройте таблицу значений функции с шагом h.

$$y(x)=\begin{cases}(x^3+1)^2, & \text{при } x<0;\\ 0, & \text{при } 0\le x<1;\\ |x^2-5x+1|, & \text{при } x\ge 1.\end{cases}$$

для x∈[a, b] с

```
using System;

namespace Example
{
  class Program
  {
      static void Main()
      {
          Console.Write("a= ");
          double a=double.Parse(Console.ReadLine());
          Console.Write("b= ");
          double b=double.Parse(Console.ReadLine());
          Console.Write("h= ");
          double h=double.Parse(Console.ReadLine());
          double y;
          int i=1;
          Console.WriteLine("{0,3} {1,5} {1,5}","#","x", "f(x)");
          for (double x=a; x<=b; x+=h, ++i)
          {
              if (x<0)
              {
                  y=Math.Pow(Math.Pow(x,3)+1,2);
              }
              else
              {
                  if (x<1)
                  {
                    y=0;
                  }

                   else
                   {
                     y=Math.Abs(x*x-5*x+1);
                   }
              }
              Console.WriteLine("{0,3} {1,5:f2} {2,5:f2}",i,x,y);
          }
      }
  }
}
```

Результат работы программы:

#	x	f(x)
1	-1,00	0,00
2	-0,50	0,77
3	0,00	0,00
4	0,50	0,00
5	1,00	3,00
6	1,50	4,25
7	2,00	5,00

Задание. Изменить программу так, чтобы она строила таблицу значений для функции, заданной следующим образом: $y = \begin{cases} 0, & \text{если } x < 0; \\ x^2 + 1, & \text{если } x \geq 0 \text{ и } x \neq 1; \\ 1, & \text{если } x = 1. \end{cases}$