**Лекция 4. Оператор цикла for**

**Цель лекции:** Формирование у студентов знаний, связанных с использованием оператора цикла for на языке C#.

В результате изучения данной лекции студенты будут способны:

* Продемонстрировать знание синтаксиса оператора for
* Продемонстрировать понимание циклических алгоритмов

**ОПЕРАТОРЫ ЦИКЛА:** for, while, do-while

Операторы циклов используются для вычислений, повторяющихся многократно.

В C# имеется четыре вида циклов:

* цикл с параметром for;
* цикл с предусловием while;
* цикл с постусловием do-while;
* цикл foreach.

Блок, ради выполнения которого и организуется цикл, называется *телом цикла*. Остальные операторы служат для управления процессом повторения вычислений: это 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. *Проверка условия продолжения цикла* выполняется на каждой итерации либо до тела цикла (цикл с предусловием), либо после тела цикла (цикл с постусловием).

На рисунке 4.1 представлены блок- схемы операторов цикла. Разница между ними состоит в том, что тело цикла с постусловием всегда выполняется хотя бы один раз, после чего проверяется, надо ли его выполнять еще раз. Проверка необходимости выполнения цикла с

предусловием делается до тела цикла, поэтому возможно, что он не выполнится ни разу.

*Параметром цикла* называется переменная, которая используется при проверке условия продолжения цикла и принудительно изменяется на каждой итерации на одну и ту же величину. Если параметр цикла целочисленный, он называется *счетчиком цикла*. Количество повторений такого цикла можно определить заранее.

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, goto. Передавать управление извне внутрь цикла запрещается.

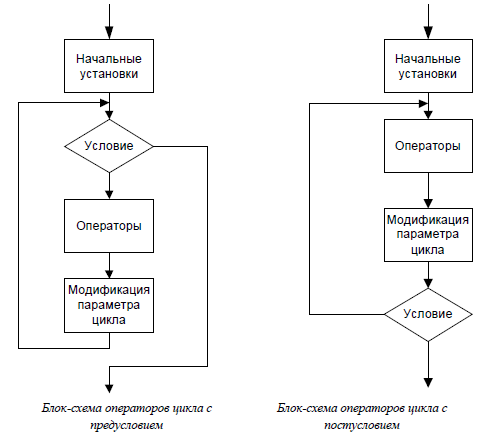


Рисунок 4.1 – Блок-схемы операторов цикла

Ниже приведена общая форма оператора for для повторного выполнения единственного

оператора.

for*(инициализация; условие;* итерация) *оператор;*

А вот как выглядит его форма для повторного выполнения кодового блока:

for(инициализация; условие; итерация)

{

последовательность операторов;

}

где *инициализация,* как правило, представлена оператором присваивания, задающим первоначальное значение переменной, которая выполняет роль счетчика и управляет циклом; *условие* — это логическое выражение, определяющее необходимость повторения цикла; а итерация — выражение, определяющее величину, на которую должно изменяться значение переменной, управляющей циклом, при каждом повторе цикла.

Обратите внимание на то, что эти три основные части оператора цикла for должны быть разделены точкой с запятой. Выполнение цикла for будет продолжаться до тех пор, пока проверка условия дает истинный результат. Как только эта проверка даст ложный результат, цикл завершится, а выполнение программы будет продолжено с оператора, следующего после цикла for.

Цикл for может продолжаться как в положительном, так и в отрицательном направлении, изменяя значение переменной управления циклом на любую величину.

В приведенном ниже примере программы выводятся числа постепенно уменьшающиеся от 100 до -100 на величину 5.

// Выполнение цикла for в отрицательном направлении.

using System;

class DecrFor {

static void Main() {

int x;

for(x = 100; x > -100; x -= 5)

Console.WriteLine(x);

}

}

В отношении циклов for следует особо подчеркнуть, что условное выражение всегда проверяется в самом начале цикла. Это означает, что код в цикле может вообще не выполняться, если проверяемое условие с самого начала оказывается ложным. Рассмотрим

следующий пример.

for(count=10; count < 5; count++)

x += count; // этот оператор не будет выполняться

Данный цикл вообще не будет выполняться, поскольку первоначальное значение переменной count, которая им управляет, сразу же оказывается больше 5. Это означает, что условное выражение count < 5 оказывается ложным с самого начала, т.е. еще до выполнения первого шага цикла.

Оператор цикла for — наиболее полезный для повторного выполнения операций известное число раз. В следующем примере программы используются два цикла for для выявления простых чисел в пределах от 2 до 20. Если число оказывается непростым, то выводится наибольший его множитель.

// Выяснить, является ли число простым. Если оно

// непростое, вывести наибольший его множитель.

using System;

class FindPrimes {

static void Main() {

int num;

int i;

int factor;

bool isprime;

for(num = 2; num < 20; num++) {

isprime = true;

factor = 0;

// Выяснить, делится ли значение переменной num нацело.

for(i=2; i <= num/2; i++) {

if((num % i) == 0) {

// Значение переменной num делится нацело.

// Следовательно, это непростое число.

isprime = false;

factor = i;

}

}

if(isprime)

Console.WriteLine(num + " — простое число.");

else

Console.WriteLine("Наибольший множитель числа " + num +

" равен " + factor);

}

}

}

Ниже приведен результат выполнения этой программы.

2 — простое число

3 — простое число

Наибольший множитель числа 4 равен 2

5 — простое число

Наибольший множитель числа 6 равен 3

7 — простое число

Наибольший множитель числа 8 равен 4

Наибольший множитель числа 9 равен 3

Наибольший множитель числа 10 равен 5

11 — простое число

Наибольший множитель числа 12 равен 6

13 — простое число

Наибольший множитель числа 14 равен 7

Наибольший множитель числа 15 равен 5

Наибольший множитель числа 16 равен 8

17 — простое число

Наибольший множитель числа 18 равен 9

19 - простое число

**Некоторые разновидности оператора цикла for**

Оператор цикла for относится к самым универсальным операторам языка С#, поскольку он допускает самые разные варианты своего применения. Некоторые разновидности оператора цикла for рассматриваются ниже.

**Применение нескольких переменных управления циклом**

В операторе цикла for разрешается использовать две или более переменных для управления циклом. В этом случае операторы инициализации и инкремента каждой переменной разделяются запятой. Рассмотрим следующий пример программы.

// Использовать запятые в операторе цикла for.

using System;

class Comma {

static void Main() {

int i, j;

for(i=0, j=10; i < j; i++, j--)

Console.WriteLine("i и j: " + i + " " + j);

}

}

Выполнение этой программы дает следующий результат.

i и j: 0 10

i и j: 1 9

i и j: 2 8

i и j: 3 7

i и j: 4 6

В данном примере запятыми разделяются два оператора инициализации и еще два итерационных выражения. Когда цикл начинается, инициализируются обе переменные, i и j. Всякий раз, когда цикл повторяется, переменная i инкрементируется, а переменная j декрементируется. Применение нескольких переменных управления циклом нередко оказывается удобным, упрощая некоторые алгоритмы. Теоретически в операторе цикла for может присутствовать любое количество операторов инициализации и итерации, но на практике цикл получается слишком громоздким, если применяется более двух подобных операторов.

Ниже приведен практический пример применения нескольких переменных управления циклом в операторе for. В этом примере программы используются две переменные управления одним циклом for для выявления наибольшего и наименьшего множителя целого числа (в данном случае — 100). Обратите особое внимание на условие окончания цикла. Оно опирается на обе переменные управления циклом.

// Использовать запятые в операторе цикла for для

// выявления наименьшего и наибольшего множителя числа.

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);

Console.WriteLine("Наименьший множитель: " + smallest);

}

}

Ниже приведен результат выполнения этой программы.

Наибольший множитель: 50

Наименьший множитель: 2

Благодаря применению двух переменных управления циклом удается выявить наименьший

и наибольший множители числа в одном цикле for. В частности, управляющая переменная i служит для выявления наименьшего множителя. Первоначально ее значение устанавливается равным 2 и затем инкрементируется до тех пор, пока не превысит половину значения переменной num. А управляющая переменная j служит для выявления наибольшего множителя. Ее значение первоначально устанавливается равным половине значения переменной num и затем декрементируется до тех пор, пока не станет меньше 2. Цикл продолжает выполняться до тех пор, пока обе переменные, i и j, не достигнут своих конечных значений. По завершении цикла оба множителя оказываются выявленными.

**Условное выражение**

Условным выражением, управляющим циклом for, может быть любое действительное выражение, дающее результат типа bool. В него не обязательно должна входить переменная управления циклом. В следующем примере программы управление циклом for осуществляется с помощью значения переменной done.

// Условием выполнения цикла может служить любое выражение типа bool.

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);

}

}

}

Ниже приведен результат выполнения этой программы.

i, j: 0 100

i, j: 1 99

i, j: 2 98

i, j: 3 97

i, j: 4 96

i, j: 5 95

i, j: 6 94

i, j: 7 93

i, j: 8 92

i, j: 9 91

i, j: 10 90

В данном примере цикл for повторяется до тех пор, пока значение переменной done типа не окажется истинным (true). Истинное значение переменной done устанавливается в цикле, когда квадрат значения переменной i оказывается больше или равным значению переменной j.

**Отсутствующие части цикла**

Ряд интересных разновидностей цикла for получается в том случае, если оставить пустыми отдельные части определения цикла. В C# допускается оставлять пустыми любые или же все части инициализации, условия и итерации в операторе цикла for. В качестве примера рассмотрим такую программу.

// Отдельные части цикла for могут оставаться пустыми.

using System;

class Empty {

static void Main() {

int i;

for(i = 0; i < 10; ) {

Console.WriteLine("Проход №" + i);

i++; // инкрементировать переменную управления циклом

}

}

}

В данном примере итерационное выражение в определении цикла for оказывается пустым, т.е. оно вообще отсутствует. Вместо этого переменная i, управляющая циклом, инкрементируется в теле самого цикла. Это означает, что всякий раз, когда цикл повторяется, значение переменной i проверяется на равенство числу 10, но никаких других действий при этом не происходит. А поскольку переменная i инкрементируется в теле цикла, то сам цикл выполняется обычным образом, выводя приведенный ниже результат.

Проход №0

Проход №1

Проход №2

Проход №3

Проход №4

Проход №5

Проход №6

Проход №7

Проход №8

Проход №9

В следующем примере программы из определения цикла for исключена инициализирующая

часть.

// Исключить еще одну часть из определения цикла for.

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

}

}

}

В данном примере переменная i инициализируется перед началом цикла, а не в самом цикле for. Как правило, переменная управления циклом инициализируется в цикле for. Выведение инициализирующей части за пределы цикла обычно делается лишь в том случае, если первоначальное значение данной переменной получается в результате сложного процесса, который нецелесообразно вводить в операторе цикла for.

**Бесконечный цикл**

Если оставить пустым выражение условия в операторе цикла for, то получится *бесконечный цикл,* т.е. такой цикл, который никогда не заканчивается. В качестве примера в следующем фрагменте кода показано, каким образом в С# обычно создается бесконечный цикл.

for(;;) // цикл, намеренно сделанный бесконечным

{

//...

}

Этот цикл будет выполняться бесконечно. Несмотря на то что бесконечные циклы требуются для решения некоторых задач программирования, например при разработке командных процессоров операционных систем, большинство так называемых "бесконечных" циклов на самом деле представляет собой циклы со специальными требованиями к завершению.

**Циклы без тела**

В C# допускается оставлять пустым тело цикла for или любого другого цикла, поскольку

*пустой оператор* с точки зрения синтаксиса этого языка считается действительным. Циклы без тела нередко оказываются полезными. Например, в следующей программе цикл без тела служит для получения суммы чисел от 1 до 5.

// Тело цикла может быть пустым.

using System;

class Empty3 {

static void Main() {

int i;

int sum = 0;

// получить сумму чисел от 1 до 5

for(i = 1; i <= 5; sum += i++);

Console.WriteLine("Сумма равна " + sum);

}

}

Выполнение этой программы дает следующий результат.

Сумма равна 15

Обратите внимание на то, что процесс суммирования выполняется полностью в операторе цикла for, и для этого тело цикла не требуется. В этом цикле особое внимание обращает на себя итерационное выражение.

sum += i++

Подобные операторы не должны вас смущать. Они часто встречаются в программах, профессионально написанных на С#, и становятся вполне понятными, если разобрать их по частям. Дословно приведенный выше оператор означает следующее: сложить со значением переменной sum результат суммирования значений переменных sum и i, а затем инкрементировать значение переменной i. Следовательно, данный оператор равнозначен следующей последовательности операторов.

sum = sum + i;

i++;

**Объявление управляющих переменных в цикле** for

Нередко переменная, управляющая циклом for, требуется только для выполнения самого цикла и нигде больше не используется. В таком случае управляющую переменную можно объявить в инициализирующей части оператора цикла for. Например, в приведенной ниже программе вычисляется сумма и факториал чисел от 1 до 5, а переменная i, управляющая циклом for, объявляется в этом цикле.

// Объявить переменную управления циклом в самом цикле for.

using System;

class ForVar {

static void Main() {

int sum = 0;

int fact = 1;

// вычислить факториал чисел от 1 до 5

for (int i = 1; i <= 5; i++) {

sum += i; // Переменная i действует в цикле.

fact \*= i;

}

// А здесь переменная i недоступна.

Console.WriteLine("Сумма равна " + sum);

Console.WriteLine("Факториал равен " + fact);

}

}

Объявляя переменную в цикле for, не следует забывать о том, что область действия этой переменной ограничивается пределами оператора цикла for. Это означает, что за пределами цикла действие данной переменной прекращается. Так, в приведенном выше примере переменная i оказывается недоступной за пределами цикла for. Для того чтобы использовать переменную управления циклом в каком-нибудь другом месте программы, ее нельзя объявлять в цикле for.

Прежде чем переходить к чтению следующего материала, поэкспериментируйте с собственными разновидностями оператора цикла for. В ходе эксперимента вы непременно

обнаружите замечательные свойства этого оператора цикла.