Лекция 5. Операторы цикла с предусловием и постусловием

Цель лекции: Формирование у студентов знаний, связанных с использованием операторов цикла с предусловием и постусловием на языке C#.

В результате изучения данной лекции студенты будут способны:

- Продемонстрировать знание синтаксиса операторов while и do…while
- Выбирать необходимый оператор для использования при программировании циклических алгоритмов

Оператор цикла while

Еще одним оператором цикла в C# является оператор while. Ниже приведена общая форма этого оператора:

while (*условие*) *оператор*;

где *оператор* — это единственный оператор или же блок операторов, а *условие* означает конкретное условие управления циклом и может быть любым логическим выражением. В этом цикле *оператор* выполняется до тех пор, пока *условие* истинно. Как только условие становится ложным, управление программой передается строке кода, следующей непосредственно после цикла.

Ниже приведен простой пример программы, в которой цикл while используется для вычисления порядка величины целого числа.

```
// Вычислить порядок величины целого числа.
using System;
class WhileDemo {
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);
}
}
```

Выполнение этой программы дает следующий результат.

```
Число: 435679
Порядок величины: 6
```

Приведенный выше цикл while действует следующим образом. Сначала проверяется значение переменной num. Если оно больше нуля, то переменная mag, выполняющая роль счетчика порядка величины, инкрементируется, а значение переменной num делится на 10. Цикл повторяется до тех пор, пока значение переменной num остается больше нуля. Как только оно окажется равным нулю, цикл завершается, а в переменной mag остается порядок величины первоначального числового значения.

Как и в цикле for, в цикле while проверяется условное выражение, указываемое в самом начале цикла. Это означает, что код в теле цикла может вообще не выполняться, а также избавляет от необходимости выполнять отдельную проверку перед самим циклом. Данное свойство цикла while демонстрируется в следующем примере программы, где вычисляются целые степени числа 2 от 0 до 9.

```
// Вычислить целые степени числа 2.
using System;
class Power {
static void Main() {
int e;
int result;
for(int i=0; i < 10; i++) {
result = 1;
e = i;
while (e > 0) {
result *= 2;
e--;
```

```
}
Console.WriteLine("2 в степени " + i + " равно " + result);
}
}
}
```

Результат выполнения этой программы приведен ниже.

```
2 в степени 0 равно 1
2 в степени 1 равно 2
2 в степени 2 равно 4
2 в степени 3 равно 8
2 в степени 4 равно 16
2 в степени 5 равно 32
2 в степени 6 равно 64
2 в степени 7 равно 128
2 в степени 8 равно 256
2 в степени 9 равно 512
```

Обратите внимание на то, что цикл while выполняется только в том случае, если значение переменной e больше нуля. А когда оно равно нулю, как это имеет место на первом шаге цикла for, цикл while пропускается.

Оператор цикла do-while

Третьим оператором цикла в C# является оператор do-while. В отличие от операторов цикла for и while, в которых условие проверялось в самом начале цикла, в операторе do-while условие выполнения цикла проверяется в самом его конце. Это означает, что цикл do-while всегда выполняется хотя бы один раз. Ниже приведена общая форма оператора цикла do-while.

do {

операторы;

} while (*условие*);

При наличии лишь одного оператора фигурные скобки в данной форме записи необязательны. Тем не менее они зачастую используются для того, чтобы сделать конструкцию do-while более удобочитаемой и не путать ее с конструкцией цикла while. Цикл do-while выполняется до тех пор, пока условное выражение истинно.

В приведенном ниже примере программы цикл do-while используется для представления отдельных цифр целого числа в обратном порядке.

```
// Отобразить цифры целого числа в обратном порядке.
using System;
class DoWhileDemo {
static void Main() {
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();
} }
```

Выполнение этой программы дает следующий результат.

Число: 198

Число в обратном порядке: 891

Приведенный выше цикл действует следующим образом. На каждом его шаге крайняя слева цифра получается в результате расчета остатка от деления целого числа (значения переменной num) на 10. Полученная в итоге цифра отображается. Далее значение переменной num делится на 10. А поскольку это целочисленное деление, то в его результате крайняя слева цифра отбрасывается. Этот процесс повторяется до тех пор, пока значение переменной num не достигнет нуля.

Применение оператора break для выхода из цикла

С помощью оператора break можно специально организовать немедленный выход из цикла в обход любого кода, оставшегося в теле цикла, а также минуя проверку условия цикла. Когда в теле цикла встречается оператор break, цикл завершается, а выполнение программы возобновляется с оператора, следующего после этого цикла. Рассмотрим простой пример программы.

```
// Применить оператор break для выхода из цикла.
using System;
class BreakDemo {
static void Main() {
// Использовать оператор break для выхода из этого цикла.
for(int i=-10; i <= 10; i++) {
if(i > 0) break; // завершить цикл, как только значение
// переменной i станет положительным
Console.Write(i + " ");
}
Console.WriteLine("Готово!");
}
}
```

Выполнение этой программы дает следующий результат.

-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 Готово!

Как видите, цикл for организован для выполнения в пределах от -10 до 10, но, несмотря на это, оператор break прерывает его раньше, когда значение переменной i становится положительным.

Оператор break можно применять в любом цикле, предусмотренном в C#. В качестве примера ниже приведена версия предыдущей программы, переделанная с целью использовать цикл do-while.

```
// Применить оператор break для выхода из цикла do-while.
using System;
class BreakDemo2 {
static void Main() {
int i;
i = -10;
do {
if(i > 0) break;
Console.Write(i + " ");
i++;
} while(i <= 10);
Console.WriteLine("Готово!");
}
```

А теперь рассмотрим более практический пример применения оператора break.

В приведенной ниже программе выявляется наименьший множитель числа.

```
// Выявить наименьший множитель числа.
using System;
class FindSmallestFactor {
static void Main() {
int factor = 1;
int num = -1000;
for (int i=2; i <= num/i;' i++) {
if((num%i) == 0) {
factor = i;
break; // прервать цикл, как только будет
// выявлен наименьший множитель числа
}
}
Console.WriteLine("Наименьший множитель равен " + factor);
}
}
```

Результат выполнения этой программы выглядит следующим образом.

```
Наименьший множитель равен 2
```

Оператор break прерывает выполнение цикла for, как только будет выявлен наименьший множитель числа. Благодаря такому применению оператора break исключается опробование любых других значений после выявления наименьшего множителя числа, а следовательно, и неэффективное выполнение кода.

Если оператор break применяется в целом ряде вложенных циклов, то он прерывает выполнение только самого внутреннего цикла. В качестве примера рассмотрим следующую программу.

```
// Применить оператор break во вложенных циклах.

using System;

class BreakNested {

static void Main() {

for(int i=0; i<3; i++) {

Console.WriteLine("Подсчет во внешнем цикле: " + i);

Console.Write(" Подсчет во внутреннем цикле: ");

int t = 0;

while(t < 100) {

if(t == 10) break; // прервать цикл, если t равно 10

Console.Write(t + " ");

t++;

}

Console.WriteLine();

}

Console.WriteLine("Циклы завершены.");

}

}
```

Выполнение этой программы дает следующий результат.

Подсчет во внешнем цикле: 0

Подсчет во внутреннем цикле: 0 1 2 3 4 5 6 7 8 9

Подсчет во внешнем цикле: 1

Подсчет во внутреннем цикле: 0 1 2 3 4 5 6 1 8 9

Подсчет во внешнем цикле: 2

Подсчет во внутреннем цикле: 0 1 2 3 4 5 6 1 8 9

Циклы завершены

Как видите, оператор break из внутреннего цикла вызывает прерывание только этого цикла, а на выполнение внешнего цикла он не оказывает никакого влияния. В отношении оператора break необходимо также иметь в виду следующее. Во-первых, в теле цикле может присутствовать несколько операторов break, но применять их следует очень аккуратно, поскольку чрезмерное количество операторов break обычно приводит к нарушению нормальной структуры кода. И во-вторых, оператор break, выполняющий выход из оператора switch, оказывает воздействие только на этот оператор, но не на объемлющие его циклы.

Применение оператора continue

С помощью оператора continue можно организовать преждевременное завершение шага итерации цикла в обход обычной структуры управления циклом. Оператор continue осуществляет принудительный переход к следующему шагу цикла, пропуская любой код, оставшийся невыполненным. Таким образом, оператор continue служит своего рода дополнением оператора break. В приведенном ниже примере программы оператор continue используется в качестве вспомогательного средства для вывода четных чисел в пределах от 0 до 100.

```
// Применить оператор continue.
using System;
class ContDemo {
static void Main() {
// вывести четные числа от 0 до 100.
for (int i = 0; i <= 100; i++) {
if((i%2) != 0) continue; // перейти к следующему шагу итерации
Console.WriteLine(i);
}
}
}
```

Заключение

В рамках данной лекции были изучены операторы цикла с предусловием (`while`) и постусловием (`do-while`), а также правила применения оператора безусловного перехода `break` для принудительного завершения циклических конструкций.

Понимание различий между циклами `while` и `do-while` позволяет программисту осознанно выбирать оптимальный инструмент для реализации алгоритмов с заранее неизвестным числом повторений. Если выполнение тела цикла зависит от предварительной проверки условия (и может не потребоваться ни разу), применяется цикл `while`; если же действия должны выполниться минимум один раз до проверки условия (например, при обработке хотя бы одной цифры числа или вводе данных), используется цикл `do-while`. Грамотное применение оператора `break` позволяет избегать избыточных вычислений, однако требует аккуратности, чтобы не нарушать логическую структуру и читаемость кода, особенно во вложенных циклах.

Вопросы для самопроверки

1. **Принцип работы `while`:** Как работает оператор цикла `while`? Может ли тело цикла `while` не выполниться ни одного раза, и при каком условии это происходит?

2. **Принцип работы `do-while`:** В чем состоит главное отличие цикла `do-while` от циклов `for` и `while`? Сколько раз минимум выполнится тело цикла `do-while`, даже если условие изначально ложно?

3. **Синтаксис `do-while`:** Какой символ обязательно ставится после закрывающей круглой скобки с условием в конце конструкции `do { ... } while (условие);`? Для чего рекомендуется всегда использовать фигурные скобки в этом цикле, даже если в теле всего один оператор?

4. **Выбор типа цикла:** В каких алгоритмических задачах целесообразнее использовать цикл `while` или `do-while` вместо цикла с параметром `for`?

5. **Оператор `break`:** Каково назначение оператора `break` внутри цикла? Что происходит с оставшимися операторами в теле цикла и с проверкой условия продолжения после срабатывания `break`?

6. **Вложенные циклы и `break`:** Если оператор `break` находится внутри вложенного (внутреннего) цикла, прекратит ли он выполнение внешнего цикла? Обоснуйте ответ.

7. **Читаемость кода:** Почему в программах не рекомендуется злоупотреблять большим количеством операторов `break` внутри одного цикла?

Список используемых источников

1. **Шилдт Г.** C# 4.0: полное руководство. — М.: Вильямс, 2019. — 1056 с. *(Глава 5: Управляющие операторы — подробный разбор циклов* `while`*,* `do-while` *и оператора* `break`*).*

2. **Троелсен Э., Джепикс Ф.** Язык программирования C# и платформа .NET. — СПб.: Диалектика, 2022. — 1312 с. *(Раздел: Конструкции итерации и управление потоком выполнения).*

3. **Албахари Д., Албахари Б.**

4. C#. Справочник. Полное описание языка. — СПб.: БХВ-Петербург, 2023. — 1088 с. *(Спецификация операторов* `while`*,* `do` *и инструкций перехода).*

5. **Павловская Т. А.** C#. Программирование на языке высокого уровня: Учебник для вузов. — СПб.: Питер, 2020. — 432 с. *(Глава: Программирование циклических алгоритмов с предусловием и постусловием).*

6. **Официальная документация Microsoft Learn:** Операторы `while` и `do` (справочник по C#) [Электронный ресурс]. — URL: microsoft.com (дата обращения: 13.11.2026).