

Основы программирования

Синтаксис языка C#

Операторы
Операторы перехода

Операторы перехода

Осуществляют безусловную передачу управления.

- `break;`
- `continue;`
- `goto;`
- `return;`
- `throw.`

Операторы перехода

- Цель оператора перехода – это точка которой передается управление.
- Цель оператора перехода находится всегда вне блока в котором он находится.
Говорят, что оператор перехода производит *выход* из блока.
- Оператор перехода передает управление за пределы блока, но он никогда не передает управление внутрь блока.

Оператор `break`

Осуществляет выход из ближайшего оператора:

- `switch;`
- `while;`
- `do ... while;`
- `for;`
- `foreach.`

Управление передается оператору, следующему непосредственно за прерванным.

При использовании внутри вложенных циклов оператор `break` прерывает только самый внутренний цикл.

Оператор break. Пример 1

```
for (int i = 0; i < 9; i++)  
{  
    if (i == 5)  
        break;  
    Console.WriteLine(i);  
}
```

0
1
2
3
4

Оператор break. Пример 2

Нахождение наименьшего множителя числа:

```
int factor = 1;
int num = 1000;

for(int i=2; i < num/2; i++) {
    if((num%i) == 0) {
        factor = i;
        break; // Цикл прекращается, когда найден множитель.
    }
}
Console.WriteLine(
    "Наименьший множитель равен " + factor);
```

Наименьший множитель равен 2

Оператор break. Пример 3

Вложенные циклы:

```
for(int i=0; i<3; i++) {  
    Console.WriteLine(  
        "Подсчет итераций внешнего цикла: " + i);  
    Console.Write(  
        "    Подсчет итераций внутреннего цикла: ");  
  
    int t = 0;  
    while(t < 100) {  
        if(t == 10) break; // Останов цикла, когда  
                           // t равно 10.  
        Console.Write(t + " ");  
        t++;  
    }  
    Console.WriteLine();  
}  
Console.WriteLine("Циклы завершены.");
```

Оператор break. Пример 3

Результат работы:

Подсчет итераций внешнего цикла: 0

 Подсчет итераций внутреннего цикла: 0 1 2 3 4 5 6 7 8 9

Подсчет итераций внешнего цикла: 1

 Подсчет итераций внутреннего цикла: 0 1 2 3 4 5 6 7 8 9

Подсчет итераций внешнего цикла: 2

 Подсчет итераций внутреннего цикла: 0 1 2 3 4 5 6 7 8 9

Циклы завершены.

Оператор continue

Оператор `continue` просто пропускает текущую итерацию без завершения цикла.

```
for (int i = 0; i < 9; i++)  
{  
    if (i == 5)  
        continue;  
    Console.WriteLine(i);  
}
```

0
1
2
3
4
6
7
8

Оператор `continue`

Используется только в теле цикла.

В отличие от оператора `break`, завершающего внутренний цикл, `continue` осуществляет переход к следующей итерации этого цикла.

Оператор `continue` осуществляет переход к новой итерации любого оператора цикла.

```
// Выводим четные числа между 0 и 100.  
for(int i = 0; i <= 100; i++) {  
    if((i%2) != 0) continue; // Переход на следующую  
                             // итерацию.  
    Console.WriteLine(i);  
}
```

Оператор `continue`. Особенности

- В циклах `while` и `do-while` оператор `continue` передает управление непосредственно оператору, проверяющему условное выражение, после чего циклический процесс продолжается.
- В цикле `for` после выполнения оператора `continue` сначала вычисляется итерационное выражение, а затем — условное. И только после этого циклический процесс будет продолжен

Оператор возврата из метода return

Прекращает выполнение метода и возвращает управление в точку вызова метода.

Две формы:

- `return;` //без возвращаемого значения
- `return значение;` //с возвращаемым значением

Оператор goto

Оператор безусловного перехода.

Применяются для принудительного изменения порядка выполнения последовательной программы.

Управление передается оператору указанному с помощью метки.

Формы оператора goto:

```
goto метка;
```

```
goto case константа;
```

```
goto default;
```

Использование оператора `goto`

- Использование оператора `goto` НЕЖЕЛАТЕЛЬНО, поскольку нарушает принципы структурного программирования.
- Всегда можно обойтись без оператора `goto`.
- Иногда его использование может быть полезным.

Операторы перехода. Форма 1

`goto метка;`

`метка: оператор;`

- Передает управление внутри текущего блока.
- Метка должна находиться в одном блоке с оператором `goto`, который ссылается на эту метку.
- Внутри блока метка должна быть уникальной.
- Оператор `goto` должен находиться в области видимости метки

Операторы перехода. Форма 2 и 3

```
goto case константа;  
goto default;
```

Используются в переключателе `switch` для передачи управления ветви `case`, или `default`.

Оператор goto. Пример

```
int i=0, j=0, k=0;
for(i=0; i < 10; i++) {
    for(j=0; j < 10; j++ ) {
        for(k=0; k < 10; k++) {
            Console.WriteLine ("i, j, k:
                                " + i + " " + j + " " + k);
            if(k == 3) goto stop;
        }
    }
}

stop: Console.WriteLine("Остановлено! i,j,k:
                        " + i + ", " + j + " " + k) ;
```

Оператор goto. Пример

```
Console.WriteLine("Введите целое число");  
int num = Int32.Parse(Console.ReadLine());  
switch (num % 7) {  
    case 1: Console.WriteLine("1 "); goto case 3;  
    case 2: Console.WriteLine("2 "); goto case 4;  
    case 3: Console.WriteLine("3 "); goto case 5;  
    case 4: Console.WriteLine("4 "); goto case 6;  
    case 5: Console.WriteLine("5 "); break;  
    case 6: Console.WriteLine("6 "); break;  
    default :Console.WriteLine("0"); break;  
}
```

Оператор goto. Пример

Из одного блока case нужно также выполнить код из другого case:

```
string s = Console.ReadLine();
switch (s) {
    case "Q":
        Console.Write("Заглавная буква ");
        goto case "q"; // Переход к коду в case "q"
    case "w": Console.WriteLine("w"); break;
    case "q": Console.WriteLine("q"); break;
    default:
        Console.WriteLine("Ввод не определен");
        break; }
```