ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР
ПРОГРАММИРОВАНИЯ И ВЫСОКИХ ТЕХНОЛОГИЙ

2020г.

Операторы языка C#

AGENDA

- ✓ Управляющие операторы
- ✓ Операторы переходов
- ✓ Операторы проверки условий
- ✓ Операторы циклов

int.Parse преобразует строку (string) в int (если может).

Convert.ToInt32 преобразует в int значение любого типа, из перечисленных здесь. boolean, double, float и т.д. А самое главное, может преобразовать экземпляр любого класса, реализующего IConvertible.

Если говорить о строках, то Convert.ToInt32 внутри всё равно использует int.Parse. Но есть нюанс:

```
public static int ToInt32(String value) {
    if (value == null)
        return 0;
    return Int32.Parse(value, CultureInfo.CurrentCulture);
}
```

Для неинициализированной строки int.Parse выбросит исключение, Convert.ToInt32 вернет 0.

Управляющие операторы

В языке программирования **C#** существуют специальные операторы, которые в зависимости от вычисляемых значений выражений позволяют управлять ходом выполнения программы.

Управляющие операторы разделяются на три категории:

- **Операторы выбора**. Вводятся ключевыми словами **if**, **if** ... **else** ..., **switch**.
- **Итеративные операторы**. Вводятся ключевыми словами **while**, **do** ... **while**, **for**, **foreach**.
- **Операторы перехода** (в рамках методов). Вводятся ключевыми словами **break**, **continue**, **goto**, **return** и **throw**.

Отдельно стоит упомянуть про **блок** (составной оператор) – это последовательность операторов, заключенная в операторные скобки **{ }**.

Блок воспринимается компилятором как один оператор и может использоваться всюду, где синтаксис требует одного оператора, а алгоритм – нескольких.

Операторы переходов

Оператор **break** – используется для выхода из блоков операторов циклов и оператора **switch**.

```
for(int i = 1; i <= 10; i++)
{
    if (i == 3)
        break;

    Console.WriteLine("i = {0}", i);
}
```

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.

Оператор **break**, выполняющий выход из оператора **switch**, оказывает воздействие только на этот оператор, но не на объемлющие его циклы.

Операторы переходов

Оператор **continue** – используется для запуска новой итерации цикла.

```
for(int i = 1; i = 10; i++)
{
    if (i % 2 == 1)
        continue;

    Console.WriteLine("i = {0}", i);
}
```

В циклах **while** и **do-while** оператор **continue** вызывает передачу управления непосредственно условному выражению, после чего продолжается процесс выполнения цикла. А в цикле **for** сначала вычисляется итерационное выражение, затем условное выражение, после чего цикл продолжается.

Операторы переходов

Оператор **goto** – используется для передачи управления на помеченный оператор.

```
<блок кода>
goto marker;
<блок кода> - будет пропущен
marker:
<блок кода>
```

Главный недостаток оператора **goto** с точки зрения программирования заключается в том, что он вносит в программу беспорядок и делает ее практически неудобочитаемой.

Однако иногда применение оператора **goto** может, скорее, прояснить, чем запутать ход выполнения программы.

Операторы переходов

Оператор **return** – используется для завершения методов и возврата результата выполнения метода.

```
   void MyVoidMethod()
   {
      <блок кода>
      return; - конец метода
      <блок кода> - не выполнится
   }
   // будут выполнены инструкции до оператора return, после
чего метод завершится.

   int MySumMethod(int a, int b)
   {
      return a + b;
   }
   //  оператор  return  вернет  вызывающему  методу  сумму
пременных a и b.
```

Операторы проверки условий

Операторы **if**, **if-else**, **if-else if-else** – используются для проверки условия.

```
if (условие){
   <блок кода>
}
else if (условие){
   <блок кода>
}
else{
   <блок кода>
}
```

```
if (условие){
   if (условие){
      <блок кода>
   }
}
else{
   <блок кода>
}
```

Условие – это некоторое условное выражение, а блок кода – набор вырожений выполняемых если условие верно.

Операторы **else** и **else-if** не является обязательными.

Тернарная форма записи:

```
<переменная> = <условие> ? <выражение> : <выражение>;
```

Операторы проверки условий

Операторы **if**, **if-else**, **if-else if-else** – используются для проверки условия.

```
switch (<переменная>)
{
case <константа>:
    <блок кода>
    <оператор перехода>
default:
    <блок кода>
    <оператор перехода>
}
```

```
//Новая форма запись в C# 8:
<переменная> switch
{
<константа> => <выражение>,
<константа> => <выражение>,
<константа> => <выражение>,
<константа> => <выражение>,
        _ => <выражение>
 };
```

Выражения из ветви **default** выполняется в том случае, если ни одна из констант выбора не совпадает с заданным выражением.

Ветвь **default** не является обязательной.

Если происходит совпадение с одним из условий выбора, то выполняются операторы, связанные с этим условием, вплоть до оператора **break**.

Операторы циклов

Оператор **for** – используется для повторения группы операторов определенное количество итераций.

```
for (<инициализация>; <условие>; <итерация>)
{
    <блок кода>
}
```

Инициализация – установление начального значения переменной-счетчика.

Условие – логическое выражение, определяющее необходимость повторения. Цикл выполняется когда результат проверки условия - true.

Итерация – выражение, котором изменяется значение счетчика при каждом повторении цикла.

Выполнение – пока условие дает положительный результат. Условие выполнения проверяется в начале цикла.

Операторы циклов

Разновидности оператора **for:**

Применение нескольких переменных управления циклом

```
for(int i = 0, j = 10; i < j; i++, j--)
{
   Console.WriteLine("Элементы {0},{1}", i, j);
}
```

Использование различных условий

```
flag = true;
for(int i = 0; flag; i++)
{
if (i == 5)
   flag = false;
}
```

Операторы циклов

Разновидности оператора **for:**

Отсутствие итерации

```
for(int i = 0; i < 10; )
{
    i++;
    Console.WriteLine(“Значение i = {0}”, i);
}
```

Бесконечный цикл

```
for(int i = 0; ; )
{
    i++;
    Console.WriteLine(“Значение i = {0}”, i);
}
```

Операторы циклов

Оператор **do-while** – используется для повторения группы операторов с постусловием.

```
do
{
   <группа операторов>
} while (<условие>)
```

Условие – логическое выражение.
Оператор (группа операторов) выполняется пока условие истинное (**true**).

В операторе **do-while** условие выполнения цикла проверяется в самом его конце. Это означает, что цикл **do-while** всегда выполняется хотя бы один раз.

Операторы циклов

Оператор **while** – используется для повторения группы операторов с предусловием.

```
while (<условие>)
{
    <группа операторов>
}
```

Условие – логическое выражение.
Оператор (группа операторов) выполняется пока условие истинное (**true**).

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.

Операторы циклов

Оператор **while** – используется для перебора элементов объектов перечисляемых типов (список, массив).

```
foreach (<тип> <идентификатор> in <коллекция>)
{
    <блок кода>
}
```

Чтобы класс работал с **foreach** необходимо реализовать интерфейс **IEnumerable**.

Цикл **foreach** последовательно извлекает элементы контейнера и только для чтения.

Практические задания

Задание 1

Создать массив элементов произвольного типа. С помощью циклов переберать все элементы этого массива и вывести их на консоль.

С помощью цикла со счетчиком вывести на экран в одну строку все двузначные числа, кратные 5.

С помощью цикла с постусловием вывести на экран в столбик последовательность чисел : -20, -40, ...,-100.

Задание 2

Ввести с клавиатуры символ. Определить, необходимо ли нам переместить фигуру вверх, вниз, вправо, влево в зависимости от введенного символа (W, S, A, D). Вывести результат решения на экран. В случае отсутствия необходимости перемещения вывести соответствующее сообщение.