Переменные, операции и выражения

Лекция №4

Переменная

- **Переменная** – это именованная область памяти. В переменную можно записывать данные и считывать. Данные, записанные в переменной, называются **значением**переменной.

Переменная

- Для того, чтобы использовать переменную, ее сначала нужно объявить:

```
static void Main(string[] args)
{
   int a; // объявляем переменную a типа int
   a = 5; // записываем в переменную a число 5
   int b, c; // объявить можно сразу несколько переменных через запятую
   long e = 10; // при объявлении переменной можно сразу же задавать ей значение, это называется инициализацией
   float f = 5.5f; // чтобы записать число с плавающей точкой типа float, нужно после значения добавлять суффикс f.
   char g = 'g'; // объявление символьной переменой g с ее инициализацией значением символа 'g'
}
```

Константы

- **Константа** – это переменная, значение которой нельзя изменить. Константы используются для гарантирования того, что данные в этой переменной не изменятся. Для того, чтобы объявить константу, перед обычным объявлением переменной нужно добавить ключевое слово const:

```
static void Main(string[] args)
{
   const int months = 12; // объявление константы
   months = 13; // ошибка компиляции
}
```

Ключевое слово var

Начиная с версии C# 3.0 в язык было добавлено ключевое слово **var**, которое позволяет создавать переменные без явного указания типа данных. Тип данных такой переменной определяет компилятор по контексту инициализации.

```
static void Main(string[] args)
{
   var number = 5; // number будет типа int
   var text ="some text"; // text будет типа string
   var number2 = 0.5; // number2 будет типа double
}
```

Ключевое слово var

- var сохраняет принцип строгой типизации в Си-шарп. Это означает, что после того, как для переменной уже был определен тип, в нее нельзя записать данные другого типа:

```
static void Main(string[] args)
{
   var number = 5;
   number = "some text"; // ошибка, number
определен как int
}
```

Выражения

- **Выражения** строятся из операндов – констант, переменных, функций, – объединенных знаками операций и скобками. При вычислении выражения определяется его значение и тип.

Выражения

Правила вычисления выражений задают:

- **приоритет** операций,
- для операций одного приоритета **порядок применения** – слева направо или справа налево;
- преобразование типов операндов и выбор реализации для перегруженных операций;
- тип и значение результата выполнения операции над заданными значениями операндов определенного типа.

	Категория	Операции	Порядок
0	Первичные	(expr), x.y, x->y, f(x), a[x], x++, x—, new,typeof(t), checked(expr),unchecked(expr)	Слева направо
1	Унарные	+, -, !, ~, ++x, —x, (T)x, sizeof(t)	Слева направо
2	Мультипликативные (Умножение)	*, /, %	Слева направо
3	Аддитивные(Сложение)	+, -	Слева направо
4	Сдвиг	<< ,>>	Слева направо
5	Отношения, проверка типов	<, >, <=, >=, is, as	Слева направо
6	Эквивалентность	==, !=	Слева направо
7	Логическое И (AND)	&	Слева направо
8	Логическое исключающее ИЛИ (XOR)	^	Слева направо
9	Логическое ИЛИ (OR)	|	Слева направо
10	Условное логическое И	&&	Слева направо
11	Условное логическое ИЛИ	||	Слева направо
12	Условное выражение	? :	Справа налево
13	Присваивание	=, *=, /=, %=, +=, -=, <<=, >>=,	Справа налево
	Склеивание c null	&=, ^=, |=??	

Инкремент и дикремент

- Операции инкремента (++) и декремента (--) увеличивают и уменьшают операнд на единицу. Они имеют две формы записи — *префиксную*, когда знак операции записывается перед операндом, и *постфиксную*.

Инкремент и дикремент

- {
- int a = 10;
- ++a;// a = 11
- Console.WriteIn(a++);//11, a = 12
- }

Операция new

- Операция new служит для создания нового объекта. Формат операции:

  new тип ([аргументы])
- С помощью этой операции можно создавать объекты как ссылочных, так и значимых типов, например:

```
object z = new object();
int i = new int();          // то же самое, что int
   i = 0;
```

Операции отрицания

- *Арифметическое отрицание* (унарный минус –) меняет знак операнда на противоположный.
- *Логическое отрицание* (!) определено для типа bool. Результат операции — значение false, если операнд равен true, и значение true, если операнд равен false.
- *Поразрядное отрицание* (~), часто называемое побитовым, инвертирует каждый разряд в двоичном представлении операнда типа int, uint, long или ulong.

Явное преобразование типа

- Операция используется для явного преобразования величины из одного типа в другой. (тип) выражение. Здесь тип — это имя того типа, в который осуществляется преобразование, а выражение чаще всего представляет собой имя переменной, например:
- long b = 300;
- int a = (int) b; // данные не теряются
- int d = (byte) a; // данные теряются

Умножение

- *Операция умножения* (*) возвращает результат перемножения двух операндов.

Таблица 3.2. Результаты вещественного умножения

*	+y	-y	+0	-0	+	-	NaN
+x	+z	-z	+0	-0	+	-	NaN
-x	-z	+z	-0	+0	-	+	NaN
+0	+0	-0	+0	-0	NaN	NaN	NaN
-0	-0	+0	-0	+0	NaN	NaN	NaN
+	+	-	NaN	NaN	+	-	NaN
-	-	+	NaN	NaN	-	+	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Деление

- *Операция деления* (/) вычисляет частное от деления первого операнда на второй.
- Если оба операнда целочисленные, результат операции округляется вниз до ближайшего целого числа. Если делитель равен нулю, генерируется исключение System.DivideByZeroException.
- Для финансовых величин (тип decimal) при делении на 0 и переполнении генерируются соответствующие исключения, при*исчезновении порядка* результат равен 0.

Остаток от деления

- *Операция остатка от деления* (%) если оба операнда целочисленные, результат операции вычисляется по формуле x - (x / y) * y. Если делитель равен нулю, генерируется исключение System.DivideByZeroException.
- Если хотя бы один из операндов вещественный, результат операции вычисляется по формуле x – n * y, где n — наибольшее целое, меньшее или равное результату деления x на y.

Операции сдвига

- *Операции сдвига* (<< и >>) применяются к целочисленным операндам. Они сдвигают двоичное представление первого операнда влево или вправо на количество двоичных разрядов, заданное вторым операндом.
- При *сдвиге влево* (<<) освободившиеся разряды обнуляются. При *сдвиге вправо* (>>) освободившиеся биты заполняются нулями, если первый операнд беззнакового типа, и знаковым разрядом в противном случае.

Операции отношения

- *Операции отношения* (<, <=, >, >=, ==, !=) сравнивают первый операнд со вторым. Операнды должны быть арифметического типа. Результат операции — логического типа, равен true или false.

Условные логические операции

- *Условные логические операции* И (&&) и ИЛИ (||) чаще всего используются с операндами логического типа.
- Результат операции *логическое И* имеет значение true, только если оба операнда имеют значение true.
- Результат операции *логическое ИЛИ* имеет значение true, если хотя бы один из операндов имеет значение true. Если значения первого операнда достаточно, чтобы определить результат операции, второй операнд не вычисляется.

Условная операция(тернарная операция)

- Условная операция (? :) имеет три операнда. Ее формат:
- операнд_1 ? операнд_2 : операнд_3
- Первый операнд — выражение, для которого существует неявное преобразование к логическому типу. Если результат вычисления первого операнда равен true, то результатом условной операции будет значение второго операнда, иначе — третьего операнда. Вычисляется всегда либо второй операнд, либо третий. Их тип может различаться.

Операции присваивания

- *Операции присваивания* (=, +=, -=, *= и т. д.) задают новое значение переменной.
- Формат операции *простого присваивания* (=):
- переменная = выражение
- Механизм выполнения операции присваивания такой: вычисляется выражение и его результат заносится в память по адресу, который определяется именем переменной, находящейся слева от знака операции. То, что ранее хранилось в этой области памяти, теряется.

Операции присваивания

- Примеры операторов присваивания:
- a = b + c / 2; x = 1;
- x = x + 0.5; ⇔ x+=0.5;
- a = b = c = 15;

Преобразование типов данных с помощью класса System.Convert

- Класс System.Convert предоставляет полный набор методов для поддерживаемых преобразований. Он обеспечивает не зависящий от языка способ выполнения преобразований и доступен во всех языках, предназначенных для среды CLR.

Класс System.Convert

- string myString = "true";
- try{ bool myBool = Convert.ToBoolean(myString); Console.WriteLine(myBool);}
- catch (FormatException){ Console.WriteLine("{0} is not a Boolean value.", myString);}

Класс System.Convert

- string newString = "123456789";
- try{ int myInt = Convert.ToInt32(newString); Console.WriteLine(myInt);}
- catch (FormatException){
- Console.WriteLine("{0} does not represent a number.", newString); }
- catch (OverflowException){ Console.WriteLine("{0} is out of range of the integer type.", newString);}