Тема 2. Основи C#

Лекція №3: Використання змінних, констант та виразів у C#.

План

1. Оголошення та використання змінних;
2. Оголошення та використання констант;
3. Використання статичних змінних та “read only”-змінних;
4. Ідентифікатори змінних та констант;
5. Вирази.

Література

1. Jeff Ferguson, Brian Patterson, Jason Beres, Pierre Boutquin, and Meeta Gupta C# Bible Wiley Publishing, Inc., 2002. — 590 с.: ил.
2. MSDN
3. Microsoft Corporation, C# Language Specification, Version 1.2, 392 с.
4. Microsoft Corporation, C# Version 2.0, 116 с.

Контрольні запитання

1. Які відмінності між оголошенням змінних у C++ та C#?
2. Чи можна використовувати неініціалізовані змінні у C#?
3. Для чого використовують константи?
4. В який момент константі присвоюється значення?
5. Чи можна створити незмінний(оголосити константою) екземпляр об'єкта класу, створеного програмістом(User Defined Type/Class)?
6. Як “виключити” перевірку компілятором спеціальних символів у змінних типу “string”?
7. Чим “string” відрізняється від “String”?
8. Як “пояснити”:) компілятору, що у виразі “a=3.14;” змінній “a” присвоюється значення “double”, а не “float”?
9. Чим використання “enum” відрізняється від використання констант?
10. Для чого у C++ та C# пропонується використання “enum”?
11. Чи правильний вираз “Z=X+1;”(з точки зору компілятора:)), якщо “Z” оголошено як константу?
12. Чим відрізняються вирази C# від виразів C++?
13. Чи можна описати ідентифікатор класу “bool”, якщо “bool” є зарезервованим словом C#?
14. Для чого на практиці дотримуються певних правил опису ідентифікаторів?
15. Що таке camel- та pascal нотації?

Завдання для самостійної роботи

- Дати письмову відповідь на контрольні запитання;
- Подумати над тим, чи можна було б обійтися у програмуванні без констант. Викласти свої аргументи у текстово-графічному вигляді.

Оголошення та використання змінних у C#

У C# використовуються наступні види змінних: “static variables”, “instance variables”, “array elements”, “value parameters”, “reference parameters”, “output parameters”, та “local variables”.

Оголошуються змінні як і у C++, проте при оголошенні змінну можна ініціалізувати.

Наприклад:

```
public class Color
{
     public countItems = 1L;
     ...
```

Ініціалізувати змінну можна пізніше, але це повинно відбутися обов'язково до початку використання, інакше .NET CLR згенерує помилку. Такий підхід також дозволяє уникнути цілого ряду помилок при програмуванні.

При оголошенні змінної текстового типу, у тексті допускається використання спеціальних символів \' \" \\ \0 \a \b \f \n \r \t та \v (як у C++).

Наприклад:

```
string c = "hello \t world";  // hello   world
```

Проте, є випадки, коли бажано не інтерпритувати зазначені символи, як спеціальні. Для таких випадків передбачене використання @.

Наприклад:

```
string d = @"C:\MyProjects\"; // C:\MyProjects\
```

Оголошення та використання констант

Константи у C# використовуються для представлення величин, значення, яких в процесі виконання програми не повинно змінюватися. З одного боку, використання констант, дозволяє уникати складних помилок у великих проектах, з іншого – код програми стає більш читабельним та зрозумілим. У C# константи оголошуються як звичайні змінні, але з використанням ключового слова “const”.

Наприклад:

```
     public const double X = 1.0;
     public const double Y = 2.0;
     public const double Z = X+1;
```

При ініціалізації змінних можна використовувати “type-suffix” для того, щоб вказати якого типу значення.

Наприклад:

```
     public const double Y = 2.0D;
     public const uint A = 1U;
```

Так, для дійсних чисел можна використовувати суфікси F, f, D, d, M та m. Для цілих чисел – U, u, L, l, UL, Ul, uL, ul, LU, Lu, lU та lu.

Використання статичних змінних та “read only”-змінних

Проте, наведений у попередньому розділі підхід може бути використаний тільки у випадку, якщо значення константи відоме програмісту, а пізніше – на етапі компіляції, компілятору. Якщо ні, то для оголошення констант використовуються поєднання “static” та “readonly”.

Наприклад:

```
public class Color
{
     public static readonly Color Black = new Color(0,0,0);
     public static readonly Color Red = new Color(255,0,0);
     public static readonly Color Green = new Color(0,255,0);
     public static readonly Color Blue = new Color(0,0,255);

     private byte red, green, blue;
     public Color(byte r, byte g, byte b) {
          red = r;
          green = g;
          blue = b;
     }
}
```

Ще одним прикладом використання констант є “enum”. Елементи, перелічені в “enum” є константами (named constant).

Наприклад:

```
using System;
enum Color{Red,Green,Blue}
class Test
{
     static void PrintColor(Color color)
     {
          if(color==Color.Red)
               Console.WriteLine("Червоний");
          else
               Console.WriteLine("Інший колір");
     }

     static void Main()
     {
          Color c = Color.Red;
          PrintColor(c);
          PrintColor(Color.Blue);
     }
}
```

Ідентифікатори змінних та констант

Підбираючи назви для змінних, методів та класів (типів) варто дотримуватися кількох правил:

1. Змінні називати, використовуючи camel-нотацію;
2. Методи називати, використовуючи pascal-нотацію;
3. назва методу чи функції повинна максимально відображати суть поняття чи об'єкта, який вони описують;
4. Якщо ідентифікатор співпадає з зарезервованим словом, то використовувати @;
5. Не використовувати у назвах заборонені символи (про це вам більше “розкаже” компілятор).

Вирази

Вирази, як і в інших мовах програмування, складаються з операторів та операндів. Вирази, в основному, служать для обчислення значень. В таблиці наведено можливі конструкції для виразів у C#.

Category	Expression	Description
Primary	x.m	Member access
	x(...)	Method and delegate invocation
	x[...]	Array and indexer access
	x++	Post-increment
	x--	Post-decrement
	new T(...)	Object and delegate creation
	new T[...]	Array creation
	typeof(T)	Obtain System.Type object for T
	checked(x)	Evaluate expression in checked context
	unchecked(x)	Evaluate expression in unchecked context
Unary	+x	Identity
	-x	Negation
	!x	Logical negation
	~x	Bitwise negation
	++x	Pre-increment
	--x	Pre-decrement
	(T)x	Explicitly convert x to type T
Multiplicative	x * y	Multiplication
	x / y	Division
	x % y	Remainder
Additive	x + y	Addition, string concatenation, delegate combination
	x – y	Subtraction, delegate removal
Shift	x << y	Shift left
	x >> y	Shift right
Relational and type testing	x < y	Less than
	x > y	Greater than
	x <= y	Less than or equal
	x >= y	Greater than or equal
	x is T	Return true if x is a T, false otherwise
	x as T	Return x typed as T, or null if x is not a T
Equality	x == y	Equal
	x != y	Not equal
Logical AND	x & y	Integer bitwise AND, boolean logical AND
Logical XOR	x ^ y	Integer bitwise XOR, boolean logical XOR
Logical OR	x | y	Integer bitwise OR, boolean logical OR
Conditional AND	x && y	Evaluates y only if x is true
Conditional OR	x || y	Evaluates y only if x is false
Conditional	x ? y : z	Evaluates y if x is true, z if x is false
Assignment	x = y	Assignment
	x *op*= y	Compound assignment; supported operators are *= /= %= += -= <<= >>= &= ^= |=