Тема 2. Основи C#

Лекція №2: Типи у C# та .NET.

План

1.Типи у C# та їх використання;
2.Види типів C#;
3.Конвертація (приведення) типів;
4.Упакування та розпакування типів;
5.Вивід за форматом та ініціалізація;
6.Аспекти сумісності типів C# та типів .NET.

Література

1. Jeff Ferguson, Brian Patterson, Jason Beres, Pierre Boutquin, and Meeta Gupta C# Bible Wiley Publishing, Inc., 2002. — 590 с.: ил..
2. Рейли Д. Создание приложений Microsoft ASP.NET/Пер. с англ. — М.; Издательско-торговыи дом «Русская Редакция», 2002. — 480 с.: ил.
3. MSDN.
4. Microsoft Visual Studio 2005 Express Edition Documentation: C# Programmer's Reference.

Контрольні запитання

1. Що таке тип даних?
2. Яке відношення мають типи .NET до типів C#?
3. Опишіть випадки, коли бажано використовувати типи .NET, а не C#-типи.
4. Що таке ініціалізація об'єктів у C#?
5. Коли використовується default-ініціалізація у C#?
6. Для чого у C# використовуються Reference Types?
7. Чим у C# відрізняються Reference Types від Value Types?
8. Для чого у C# призначені user-defined типи?
9. Що таке конвертація типів і коли вона використовується?
10. Коли виконуєтся implicit-конвертація?
11. Коли виконуєтся explicit-конвертація?
12. Що таке Boxing & Unboxing і коли вони використовуються?
13. Чому існує implicit- та explicit-конвертація?
14. Для чого у .NET використовуються типи, якщо деякі мови програмування їх не використовують?

Завдання для самостійної роботи

- Дати письмову відповідь на контрольні запитання
- Створити програмний код на C# для ілюстрації explicit та implicit техніки приведення значення однієї змінної до значення змінної іншого типу.

Типи у C# та їх використання

.NET CLS зобов'язує програміста явно вказувати типи даних перед їх використанням у програмі. Тому C# та інші сумісні з .NET CLS мови програмування передбачають процедуру опису типів даних. Тип вказується перед ідентифікатором змінної чи екземпляра об'єкта.

Наприклад:

```
int i;
int i=0;
MyInt i;
MyInt i = new MyInt();
```

Види типів у C#

C# використовує базові типи(intrinsic) та типи описані користувачем(user-defined – як в JAVA чи C++). Крім того, C# використовує типи-значення (Value Types), типи-посилання (Reference Types) та типи-вказівники (pointers).

Кожен базовий тип відповідає деякому CTS-типу. До базових відносяться: sbyte, byte, short, ushort, int, uint, long, char, float, ulong, decimal та double.

Типи-значення використовуються для опису атомарних невеликих об'єктів.

Наприклад:
```
int i;
```

Типи-посилання використовуються для опису об'єктів, розмір, яких не відповідає розміру ідентифікатора типу. Для оголошення типу-посилання використовують class, interface та delegate. Є тако вбудовані типи-посилання object та string.

Наприклад:
```
int [] i = new int[100];
```

Типи-вказівники використовуються дуже рідко і тільки у тих випадках, коли додатку потрібно мати безпосередній доступ до пам'яті.

Наприклад: int* i, a; (але не int *i, *a;!)

Конвертація (приведення) типів

Значення змінною одного типу може бути в деяких випадках приведене до значення змінної іншого типу. Розрізняють два види приведення до типу: implicit та explicit. Implicit виконується автоматично та можливе тільки для наступних ситуацій:

```
sbyte   ->  short, int, long, float, double, decimal;
byte    ->  short, ushort, int, uint, long, ulong, float, double,
            decimal;
short   ->  int, long, float, double, decimal;
ushort  ->  int, uint, long, ulong, float, double, decimal;
int     ->  long, float, double, decimal;
uint    ->  long, ulong, float, double, decimal;
long    ->  float, double, decimal;
char    ->  ushort, int, uint, long, ulong, float, double, decimal;
float   ->  double;
ulong   ->  float, double, or decimal.
```

Наприклад:
```
short sh = 2;
sbyte sb = 1;
sh = sb;
```

Explicit використовується у випадку, коли програміст має потребу явно повідомити компілятор про конвертацію типу у інший для випадків не передбачених implicit-приведенням. Застосування explicit можливе для наступних випадків:

```
sbyte   ->  byte, ushort, uint, ulong, char;
```

```
byte    ->  sbyte, char;
short   ->  sbyte, byte, ushort, uint, ulong, char;
ushort  ->  sbyte, byte, short, char;
int     ->  sbyte, byte, short, ushort, uint, ulong, char;
uint    ->  sbyte, byte, short, ushort, int, char;
long    ->  sbyte, byte, short, ushort, int, uint, ulong, char;
ulong   ->  sbyte, byte, short, ushort, int, uint, long, char;
char    ->  sbyte, byte, short;
float   ->  sbyte, byte, short, ushort, int, uint, long, ulong, char,
            decimal;
double  ->  sbyte, byte, short, ushort, int, uint, long, ulong, char,
            float, decimal;
decimal ->  sbyte, byte, short, ushort, int, uint, long, ulong, char,
            float, double;
```

Наприклад:

```
byte bt = 2;
sbyte sb = 1;
bt = (byte)sb;
```

Упакування та розпакування типів (Boxing & Unboxing)

Будь-який об'єкт можна "запакувати" у об'єкт "object". А потім розпакувати його з запакованого вигляду до його "рідного" типу. Це дозволяє організовувати колекції різнотипних об'єктів.

Наприклад:

```
int i = 123;
object o = (object) i;  // boxing
o = 123;
i = (int) o;  // unboxing
```

Проте, при використанні boxing/unboxing слід пам'ятати, що процес упакування та розпакування є потенційно небезпечним з точки зору продуктивності.

Вивід за форматом та ініціалізація

При створення змінної чи екземпляра їх значення є неініціалізованим і використовувати його у такому вигляді для виводу не рекомендується. Статичні змінні ініціалізуються по замовчуванню наступними значеннями: bool=false; byte=0; char='\0'; decimal=0.0M; double=0.0D; float=0.0F; int=0; long=0L; sbyte=0; short=0; uint=0; ulong=0; ushort=0;

Для форматування при виводі використовуються наступні шаблони:

Код	Опис	Приклад	Результат
C	Currency	Console.Write("{0:C}", 2.5);	$2.50
D	Decimal	Console.Write("{0:D5}", 25);	00025
E	Scientific	Console.Write("{0:E}", 250000);	2.500000E+005
F	Fixed-point	Console.Write("{0:F2}", 25);	25.00
G	General	Console.Write("{0:G}", 2.5);	2.5
N	Number	Console.Write("{0:N}", 2500000);	2,500,000.00
X	Hexadecimal	Console.Write("{0:X}", 250);	FA

Аспекти сумісності типів C# та типів .NET

Будь-яка .NET CLS-сумісна мова програмування може мати будь-які свої базові типи. Але вони відображаються на типи, регламентовані .NET

CLS та .NET CTS. Це дає змогу використовувати в одному додатку різні мови програмування не ризикуючи втратити дані при виконання над ними операцій засобами конкретної мови програмування. Проте, якщо код планується використовувати у різних мовних середовищах, то варто використовувати “рідні” (.NET Framework) типи. Типи C# відображаються у типи .NET наступним чином:

C# Type	**.NET Framework Type**
bool	System.Boolean
byte	System.Byte
sbyte	System.SByte
char	System.Char
decimal	System.Decimal
double	System.Double
float	System.Single
int	System.Int32
uint	System.UInt32
long	System.Int64
ulong	System.UInt64
object	System.Object
short	System.Int16
ushort	System.UInt16
string	System.String