



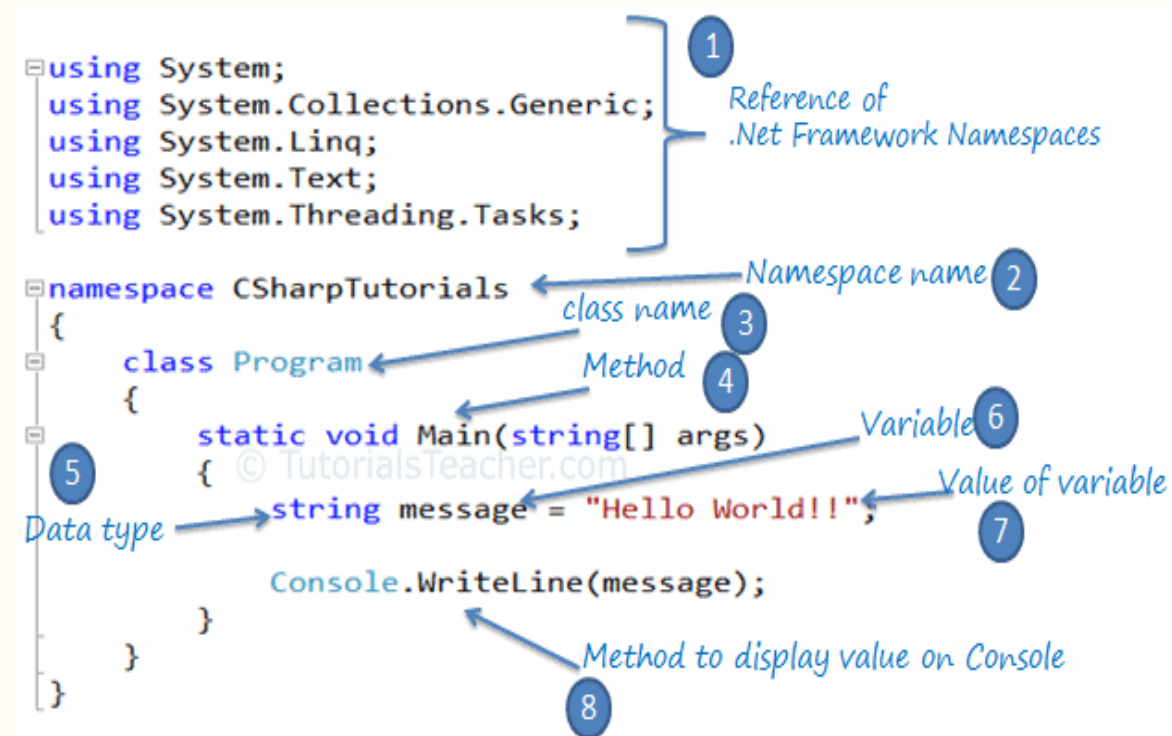
СТРУКТУРА ПРОГРАМИ МОВОЮ С#

Питання 1.3.

Історія версій C#

Версія C#	Версія .NET Framework	Версія Visual Studio	Дата релізу
1.0	.NET Framework 1.0	Visual Studio .NET 2002	Січень 2002р.
1.1, 1.2	.NET Framework 1.1	Visual Studio .NET 2003	Квітень 2003р.
C# 2.0	.NET Framework 2.0	Visual Studio 2005	Листопад 2005р.
C# 3.0	.NET Framework 3.0	Visual Studio 2008	Листопад 2006р.
C# 3.5	.NET Framework 3.5	Visual Studio 2010	Листопад 2007р.
C# 4.0	.NET Framework 4.0	Visual Studio 2010	Квітень 2010р.
C# 5.0	.NET Framework 4.5	Visual Studio 2012, 2013	Серпень 2012р.
C# 6.0	.NET Framework 4.6	Visual Studio 2015	Липень 2015р.
C# 7.0	.NET Framework 4.6.2	Visual Studio 2017	Березень 2017р.
C# 7.1	.NET Framework 4.7	Visual Studio 2017 v 15.3	Серпень 2017р.
C# 7.2	.NET Framework 4.7.1	Visual Studio 2017 v15.5	Листопад 2017р.
C# 7.3	.NET Framework 4.7.2	Visual Studio 2017 v15.7	Травень 2018р.
C# 8.0	.NET Framework 4.8	Visual Studio 2019	Вересень 2019р.

Структура простої C#-програми



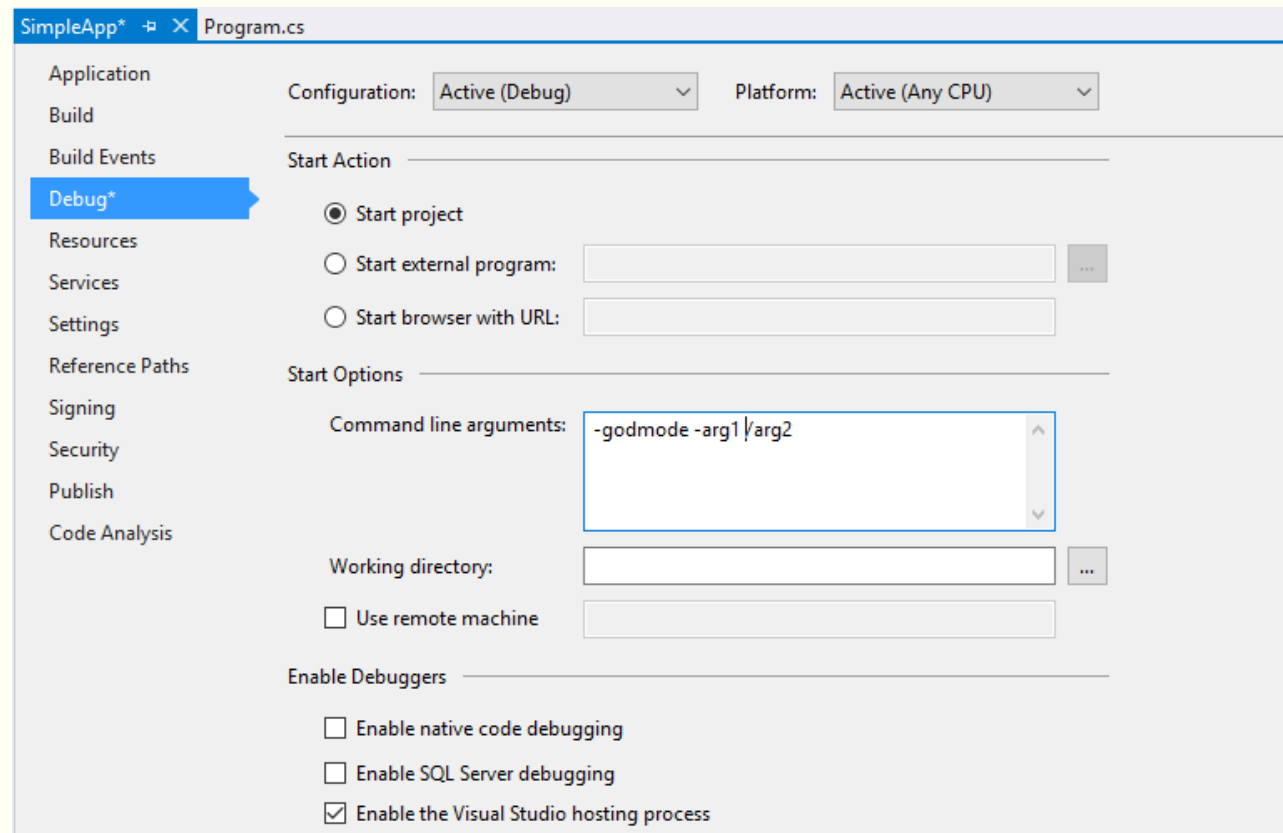
- (1) Блок посилань на інші простори імен .NET
- (2) Назва простору імен для Вашого додатку
- (3) Назва класу
- (4) Метод `Main()` – точка входу в додаток
 - Приймає аргументи командного рядка
 - Нічого не повертає
- (5) Рядковий тип даних
- (6) Змінна `message`
- (7) Рядкове значення, присвоюване змінній `message`
- (8) Метод `WriteLine()` для виводу в консоль

Аргументи командного рядка

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SimpleApp
{
    class Program
    {
        static int Main(string[] args)
        {
            // Вивести просте повідомлення для користувача
            Console.WriteLine("***** My First C# App *****");
            Console.WriteLine("Hello World!");

            // Обробити будь-які вхідні аргументи
            for (int i = 0; i < args.Length; i++)
                Console.WriteLine("Arg: {0}", args[i]);
            Console.ReadLine();
            return -1;
        }
    }
}
```



```
file:///c:/users/spuasson/documents/visual s
***** My First C# App *****
Hello World!
Arg: -godmode
Arg: -arg1
Arg: /arg2
```

Ключове слово `using` в мові C#

- Інструкція `using` визначає область, після завершення якої об'єкт видаляється.
- Директива `using` створює псевдонім для простору імен або імпортує типи, визначені в інших просторах імен.
- Директива `using static` імпортує елементи з одного класу.

Ключові слова мови C#

- Категорії ключових слів у C#:
 - **Модифікатори (Modifier Keywords)** – вказують на те, хто може змінювати типи або їх члени, дозволяють або забороняють вносити зміни іншим частинам програми: *abstract, async, const, event, extern, new, override, partial, readonly, sealed, static, unsafe, virtual, volatile*.
 - **Модифікатори доступу (Access Modifier Keywords)** – застосовуються в оголошенні класу та його членів, визначають їх доступність: *public, private, internal, protected*.
 - **Інструкції (Statement Keywords)** – ключові слова, пов'язані з ходом виконання (flow) програми: *if, else, switch, case, do, for, foreach, in, while, break, continue, default, goto, return, yield, throw, try, catch, finally, checked, unchecked, fixed, lock*.
 - **Ключові слова для параметрів методів (Method Parameter Keywords)**: *params, ref, out*.
 - **Ключові слова для просторів імен (Namespace Keywords)**: *using, оператори . та ::, extern alias*.
 - **Оператори (Operator Keywords)** – виконують різні дії: *as, await, is, new, sizeof, typeof, stackalloc, checked, unchecked*.
 - **Ключові слова для доступу (Access Keywords)** – застосовуються для доступу до батьківського класу або класу-контейнера щодо об'єкта чи класу: *base, this*.
 - **Літерали (Literal Keywords)** – застосовуються до значення об'єкта або посилання на нього: *null, false, true, value, void*.

Ключові слова мови C#

- Категорії ключових слів у C#:
 - Ключові слова типів (Type Keywords)** – позначають типи даних: *bool, byte, char, class, decimal, double, enum, float, int, long, sbyte, short, string, struct, uint, ulong, ushort*.
 - Контекстні ключові слова (Contextual Keywords)** – використовуються у специфічних випадках, слова не зарезервовані (не підсвічуються синім): *add, var, dynamic, global, set, value*.

```
public class @class
{
    public static int var { get; set; }
    public static int this { get; set; }
}
```

The image shows a C# code snippet with three annotations:

- An arrow points from the text "Reserved keyword used with @ prefix" to the `@class` identifier.
- An arrow points from the text "Contextual keyword" to the `var` keyword.
- An arrow points from the text "Reserved keyword" to the `this` keyword.

© TutorialsTeacher.com

- Ключові слова для запитів (Query Keywords)** – контекстні ключові слова в LINQ-запитах: *from, where, select, group, into, orderby, join, let, in, on, equals, by, ascending, descending*.

Ідентифікатори (назви змінних, класів, інтерфейсів тощо)

- Зарезервовані ключові слова не можуть використовуватись як ідентифікатори.

- Проте можуть використовуватись з префіксом '@'.
- Наприклад, class – зарезервоване слово, проте @class можна використовувати:

```
public class @class { public static int MyProperty { get; set; } }  
@class.MyProperty = 100;
```

- Правила іменування ідентифікаторів у мові C#.

- Угоди щодо іменування ідентифікаторів:

- Імена інтерфейсів починаються з великої літери I: IList, IEnumerable.
- Типи атрибутів закінчуються словом Attribute: [Obsolete] -> ObsoleteAttribute.
- Типи перелічень використовують для іменування прапорців іменники в множині, а не для прапорців – іменники в однині.
- Ідентифікатори не повинні містити два послідовних символи _. Ці імена зарезервовані для ідентифікаторів, створених компілятором.

Змінні та константи в мові C#

Змінні

- Оголошення:
 - тип назваЗмінної;
 - `string name;`
 - `int x;`
- Оголошення з ініціалізацією:
 - `string name = "John";`
 - `int x = 5;`
- Неявний «тип» `var` (визначається компілятором за присвоєним значенням)
 - `var i = 10; // неявно типізована змінна`
 - `int i = 10; // явно типізована змінна`

Константи

- Оголошення з ініціалізацією:
 - `const int myNum = 15;`
- Літерали:
 - `85` /* decimal */
 - `0x4b` /* hexadecimal */
 - `30` /* int */
 - `30u` /* unsigned int */
 - `30l` /* long */
 - `30ul` /* unsigned long */
 - `3.14159` /* Legal */
 - `314159E-5F` та ін.

Неявно типізовані локальні змінні

```
class ThisWillNeverCompile
{
    // Ошибка! var не может применяться к полям!
    private var myInt = 10;
    // Ошибка! var не может применяться к возвращаемому значению
    // или типу параметра!
    public var MyMethod(var x, var y){}
}
```

```
// Ошибка! Должно быть присвоено значение!
var myData;

// Ошибка! Значение должно присваиваться в самом объявлении!
var myInt;
myInt = 0;

// Ошибка! Нельзя присваивать null в качестве начального значения!
var myObj = null;
```

```
// Допустимо, если SportsCar является переменной ссылочного типа!
var myCar = new SportsCar();
myCar = null;

// Также допустимо!
var myInt = 0;
var anotherInt = myInt;
string myString = "Wake up!";
var myData = myString;
```

- Неявна типізація застосовна тільки до локальних змінних всередині контексту методу або властивості.
 - Застосовувати ключеве слово var для визначення значень, що повертає метод, параметрів чи даних полів у спеціальному типі не допускається.

Локальним var-змінним повинно присвоюватись початкове значення в самому оголошенні.

- Присвоювати null як початкове значення заборонено.
- Проте дозволяється присвоювати null локальній змінній з виведенням після початкової ініціалізації типом (за умови нулабельності типу)
- Значення неявно типізованої локальної змінної можна присвоювати іншим змінним, як явно, так і неявно типізованим.

Неявно типізовані дані є строго типізованими

- Коли використовувати var?
 - Особливої користі не принесе при оголошенні локальних var-змінних заради інтересу.
 - Ускладнюється швидке визначення типу даних, а звідси, й призначення змінної.
 - У наборі технологій LINQ використовуються вирази запитів, які можуть видавати динамічно створювані результуючі набори, засновані на форматі самого запиту.
 - Тоді неявна типізація дуже зручна, оскільки не потрібно явно визначати тип, який запит може повернути, що в деяких випадках взагалі стає неможливо явно.

```
static void LinqQueryOverInts()
{
    int[] numbers = { 10, 20, 30, 40, 1, 2, 3, 8 };
    // Запрос LINQ!
    var subset = from i in numbers where i < 10 select i;
    Console.WriteLine("Values in subset: ");
    foreach (var i in subset)
    {
        Console.WriteLine("{0} ", i);
    }
    Console.WriteLine();
    // К какому типу относится subset?
    Console.WriteLine("subset is a: {0}", subset.GetType().Name);
    Console.WriteLine("subset is defined in: {0}", subset.GetType().Namespace);
}
```

Вирази в мові C#. Основні вирази (первинний пріоритет)

- Вирази складаються з операторів та операндів.
 - Оператори групуються за пріоритетами

Вираз	Опис
<code>x.m</code>	Доступ до членів
<code>x(...)</code>	Виклик метода чи делегата
<code>x[...]</code>	Доступ до масива та індексатора
<code>x++</code>	Постфіксний інкремент
<code>x--</code>	Постфіксний декремент
<code>new T(...)</code>	Створення об'єкта та делегата
<code>new T(...){...}</code>	Створення об'єкта з ініціалізатором
<code>new {...}</code>	Анонімний ініціалізатор об'єкта
<code>new T[...]</code>	Створення масива
<code>typeof(T)</code>	Отримання об'єкта <code>System.Type</code> для <code>T</code>
<code>checked(x)</code>	Обчислення виразу в перевіреному контексті
<code>unchecked(x)</code>	Обчислення виразу в неперевіреному контексті
<code>default(T)</code>	Отримання значення за умовчанням для типу <code>T</code>
<code>delegate {...}</code>	Анонімна функція (анонімний метод)

Вирази в мові C#. Нижчі пріоритети

Унарний	+x	Значення (Identity)
	-x	Від'ємне значення
	!x	Логічне заперечення
	~x	Порозрядне заперечення
	++x	Префіксний інкремент
	--x	Префіксний декремент
	(T)x	Явне зведення x у тип T
	await x	асинхронне очікування завершення x
Мультиплікативний	x * y	Множення
	x / y	Ділення
	x % y	Остача від ділення
Адитивний	x + y	Додавання, конкатенація рядків, об'єднання делегатів
	x - y	Віднімання, видалення делегатів

Зсув	x << y	Зсув вліво
	x >> y	Зсув вправо
Перевірка типу та відношення	x < y	Менше
	x > y	Більше
	x <= y	Менше або дорівнює
	x >= y	Більше або дорівнює
	x is T	повертає true, якщо x є T, інакше – false
	x as T	повертає x з типом T або null, якщо x не є T
Рівність	x == y	Дорівнює
	x != y	Не дорівнює

Вирази в мові C#. Нижчі пріоритети

Логічне І	$x \& y$	Порозрядне логічне І для цілочисельних операндів, логічне І для операндів логічного типу
Логічне виключне АБО	$x \wedge y$	Порозрядне виключне АБО для операндів цілочисельного типу, логічне виключне АБО для операндів логічного типу
Логічне АБО	$x y$	Поразрядне АБО для операндів цілочисельного типу, логічне АБО для операндів логічного типу
Умовне логічне І (Conditional logical AND)	$x \&\& y$	
Умовне логічне АБО	$x y$	
Об'єднання із значенням NULL	$x ?? y$	Приймає значення x (або null, якщо воно допустиме), інакше - y
Умова	$x ? y : z$	повертає y, якщо x має значення true, або z, якщо x має значення false
Присвоєння або анонімна функція	$x = y$	Присвоєння значення
	$x \text{ op} = y$	$* = / = \% = + = - = << = >> = \& = \wedge = =$
	$(T \ x) \Rightarrow y$	Анонімна функція (лямбда-вираз)

Базовий ввід-вивід

- Підтримується класом Console.
 - Забезпечує зчитування та запис символів у консоль.
 - Стандартні потоки вводу, виводу та помилок представлені у вигляді властивостей.
 - Додаток може перенаправити ці властивості в інші потоки, наприклад, пов'язані з файлами.
 - За умовчанням методи зчитування використовують стандартний потік вводу (клавіатуру), а методи запису – стандартний потік виводу (монітор).

- Деякі члени класу Console:

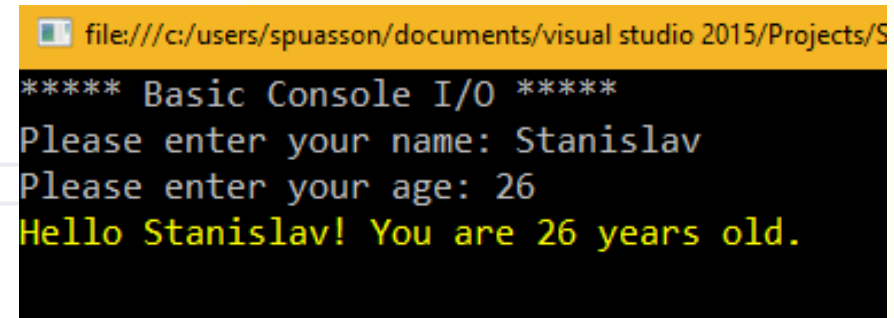
Член	Описание
Beep()	Этот метод заставляет консоль выдать звуковой сигнал указанной частоты и длительности
BackgroundColor ForegroundColor	Эти свойства устанавливают цвета фона и переднего плана для текущего вывода. Им может быть присвоен любой член перечисления ConsoleColor
BufferHeight BufferWidth	Эти свойства управляют высотой и шириной буферной области консоли
Title	Это свойство устанавливает заголовок текущей консоли
WindowHeight WindowWidth WindowTop WindowLeft	Эти свойства управляют размерами консоли по отношению к установленному буферу
Clear()	Этот метод позволяет очищать установленный буфер и область отображения консоли

Демонстрація базового вводу-виводу

```
class Program
{
    static void GetUserData()
    {
        // Получить информацию об имени и возрасте.
        Console.Write("Please enter your name: "); // Запрос на ввод имени
        string userName = Console.ReadLine();
        Console.Write("Please enter your age: "); // Запрос на ввод возраста
        string userAge = Console.ReadLine();
        // Изменить цвет переднего плана, просто ради интереса.
        ConsoleColor prevColor = Console.ForegroundColor;
        Console.ForegroundColor = ConsoleColor.Yellow;
        // Вывести полученные сведения на консоль.
        Console.WriteLine("Hello {0}! You are {1} years old.",
            userName, userAge);
        // Восстановить предыдущий цвет переднего плана.
        Console.ForegroundColor = prevColor;
    }

    static void Main(string[] args)
    {
        Console.WriteLine("***** Basic Console I/O *****");
        GetUserData();
        Console.ReadLine();
    }
}
```

- WriteLine() дозволяє помістити в потік виводу рядок тексту.
- Write() поміщає в потік виводу текст без символу повернення каретки.
- ReadLine() дозволяє отримати інформацію з потоку вводу до натиснення клавіші <Enter>.
- Read() використовується для захоплення з потоку вводу одиночного символу.

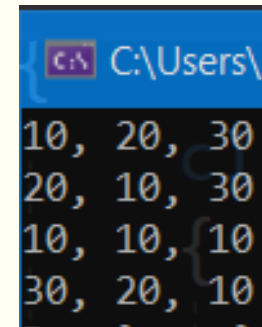


```
file:///c:/users/spuasson/documents/visual studio 2015/Projects/S
***** Basic Console I/O *****
Please enter your name: Stanislav
Please enter your age: 26
Hello Stanislav! You are 26 years old.
```


Форматування консольного виводу

- Підтримується стиль, який трохи нагадує стиль printf() з мови C.
 - Перший параметр методу WriteLine() є рядковим літералом, який містить заповнювачі виду {0}, {1}, {2} і т.д. (нумерація з 0).
 - Решта параметрів WriteLine() – це значення, які повинні підставлятись на місце заповнювачів.
 - Якщо унікальних заповнювачів більше, ніж аргументів, під час виконання буде згенеровано виняток форматування.
 - Інакше, зайві аргументи просто ігноруються.

```
Console.WriteLine("{0}, {1}, {2}", 10, 20, 30);  
Console.WriteLine("{1}, {0}, {2}", 10, 20, 30);  
Console.WriteLine("{0}, {0}, {0}", 10, 20, 30);  
Console.WriteLine($"{30}, {20}, {10}");
```



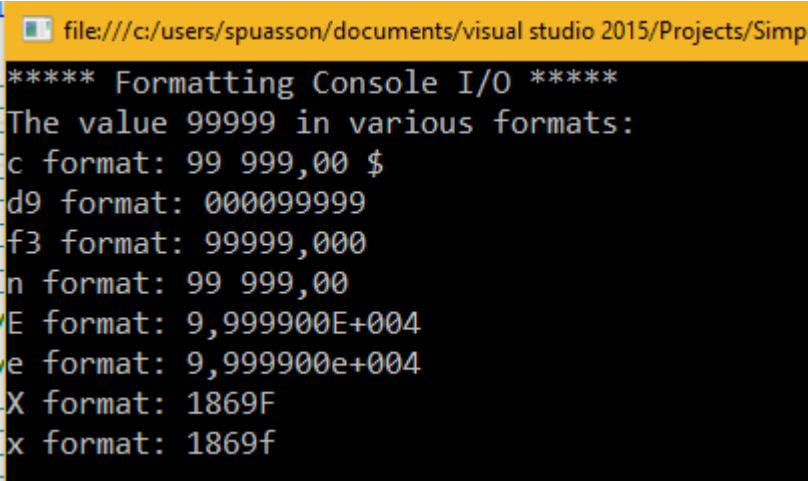
Форматування числових даних

Символ форматирования	Описание
C или c	Используется для форматирования денежных значений. По умолчанию значение предваряется символом локальной валюты (например, знаком доллара (\$)) для культуры US English)
D или d	Используется для форматирования десятичных чисел. В этом флаге можно также указывать минимальное количество цифр для представления значения
E или e	Используется для экспоненциального представления. Регистр этого флага указывает, в каком регистре должна представляться экспоненциальная константа — в верхнем (E) или в нижнем (e)
F или f	Используется для форматирования с фиксированной точкой. В этом флаге можно также указывать минимальное количество цифр для представления значения
G или g	Означает <i>general</i> (общий). Этот флаг может использоваться для представления чисел в формате с фиксированной точкой или экспоненциальном формате
N или n	Используется для базового числового форматирования (с запятыми)
X или x	Используется для шестнадцатеричного форматирования. В случае символа X в верхнем регистре шестнадцатеричное представление будет содержать символы верхнего регистра

Демонстрація виводу

```
static void FormatNumericalData()
{
    Console.WriteLine("The value 99999 in various formats:");
    Console.WriteLine("c format: {0:c}", 99999);
    Console.WriteLine("d9 format: {0:d9}", 99999);
    Console.WriteLine("f3 format: {0:f3}", 99999);
    Console.WriteLine("n format: {0:n}", 99999);
    // Обратите внимание, что использование верхнего или нижнего регистра для x
    // определяет, в каком регистре отображаются символы в шестнадцатеричном формате
    Console.WriteLine("E format: {0:E}", 99999);
    Console.WriteLine("e format: {0:e}", 99999);
    Console.WriteLine("X format: {0:X}", 99999);
    Console.WriteLine("x format: {0:x}", 99999);
}
```

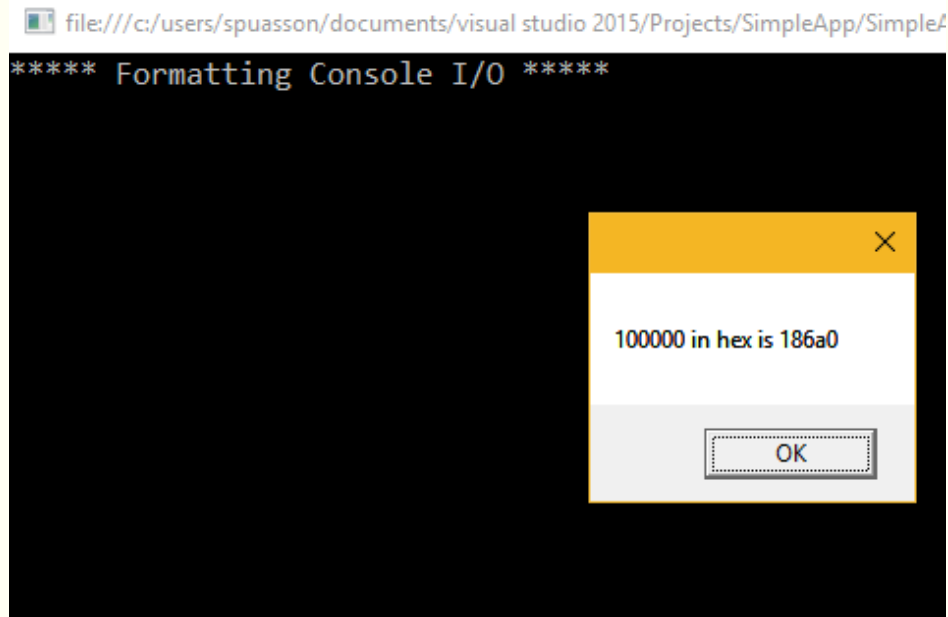
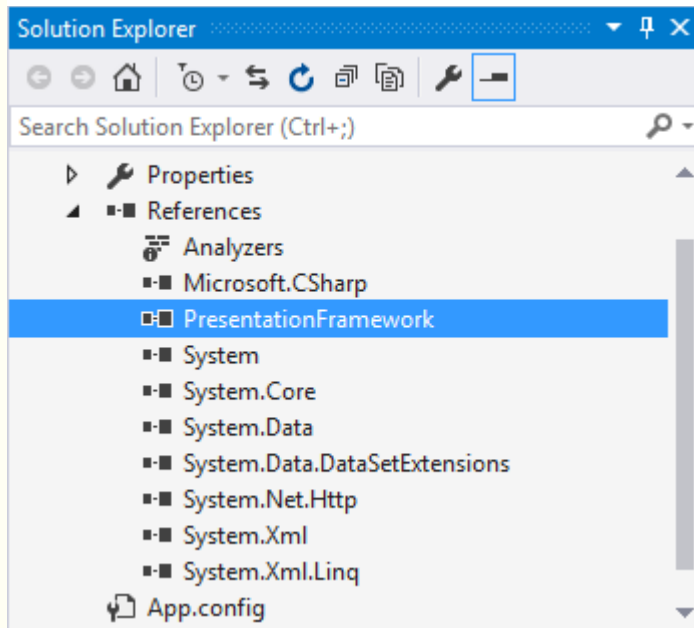
- Вивід грошових позначок залежить від регіональних стандартів комп'ютера користувача



```
file:///c:/users/spuasson/documents/visual studio 2015/Projects/Simp
***** Formatting Console I/O *****
The value 99999 in various formats:
c format: 99 999,00 $
d9 format: 000099999
f3 format: 99999,000
n format: 99 999,00
E format: 9,999900E+004
e format: 9,999900e+004
X format: 1869F
x format: 1869f
```

Форматування за допомогою методу `string.Format()`

```
static void DisplayMessage()
{
    // Использование string.Format() для форматирования строкового литерала,
    string userMessage = string.Format("100000 in hex is {0:x}", 100000);
    // Для компиляции этой строки кода требуется
    // ссылка на PresentationFramework.dll!
    System.Windows.MessageBox.Show(userMessage);
}
```



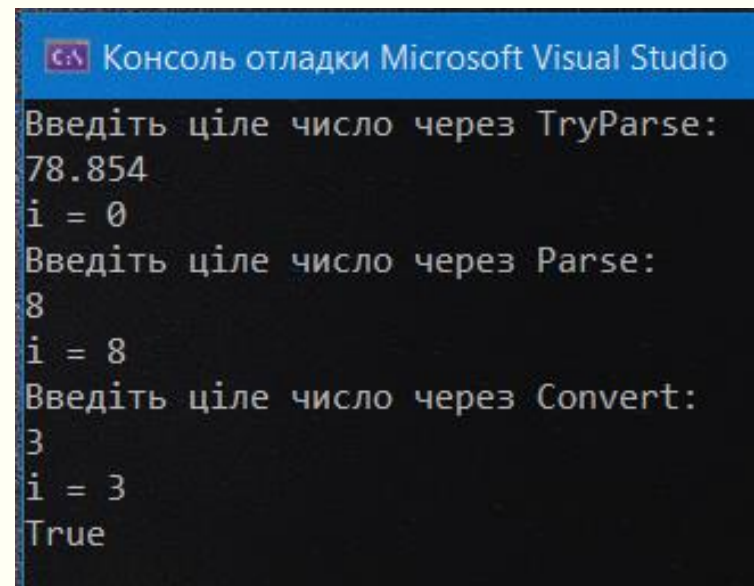
- Той же синтаксис форматування можна застосовувати при виклику статичного методу `string.Format()`.
 - Може виявитись зручним при динамічному компонуванні текстових даних для використання в додатку будь-якого типу.

Парсинг вводу та перетворення типів

```
int i;  
Console.WriteLine("Введіть ціле число через TryParse: ");  
int.TryParse(Console.ReadLine(), out i);  
Console.WriteLine($"i = {i}", i);  
  
Console.WriteLine("Введіть ціле число через Parse: ");  
i = int.Parse(Console.ReadLine());  
Console.WriteLine($"i = {i}", i);  
  
Console.WriteLine("Введіть ціле число через Convert: ");  
i = Convert.ToInt32(Console.ReadLine());  
Console.WriteLine($"i = {i}", i);  
  
// ParseExact і TryParseExact додатково приймають формат розбору  
string dateString = "09-01-2020";  
CultureInfo provider = CultureInfo.InvariantCulture;  
DateTime dateTime;  
bool isSuccess = DateTime.TryParseExact(dateString,  
    new string[] { "MM/dd/yyyy", "MM-dd-yyyy", "MM.dd.yyyy" },  
    provider,  
    DateTimeStyles.None,  
    out dateTime);  
Console.WriteLine(isSuccess);
```

■ Для перетворення введених у консоль даних з рядкового типу в інший потрібний тип можуть застосовуватись:

- відповідні методи потрібного типу: Parse(), TryParse(), ParseExact(), TryParseExact().
- відповідні методи класу Convert: Convert.ToInt32(), Convert.ToBoolean() тощо. Більш загальна реалізація.



```
Консоль отладки Microsoft Visual Studio  
Введіть ціле число через TryParse:  
78.854  
i = 0  
Введіть ціле число через Parse:  
8  
i = 8  
Введіть ціле число через Convert:  
3  
i = 3  
True
```



ДЯКУЮ ЗА УВАГУ!

Наступне питання: Система типів мови C#