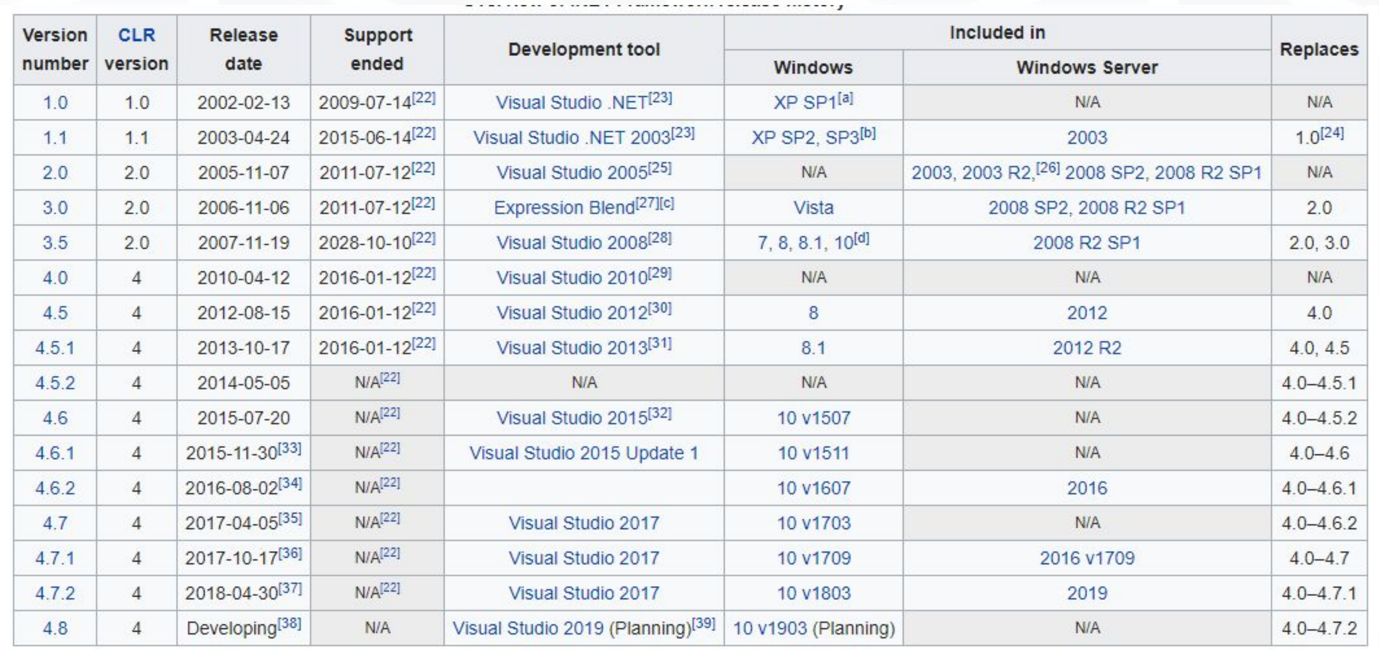
# C# & Giao diện cơ bản

# .Net framework

- Software framework của Microsoft



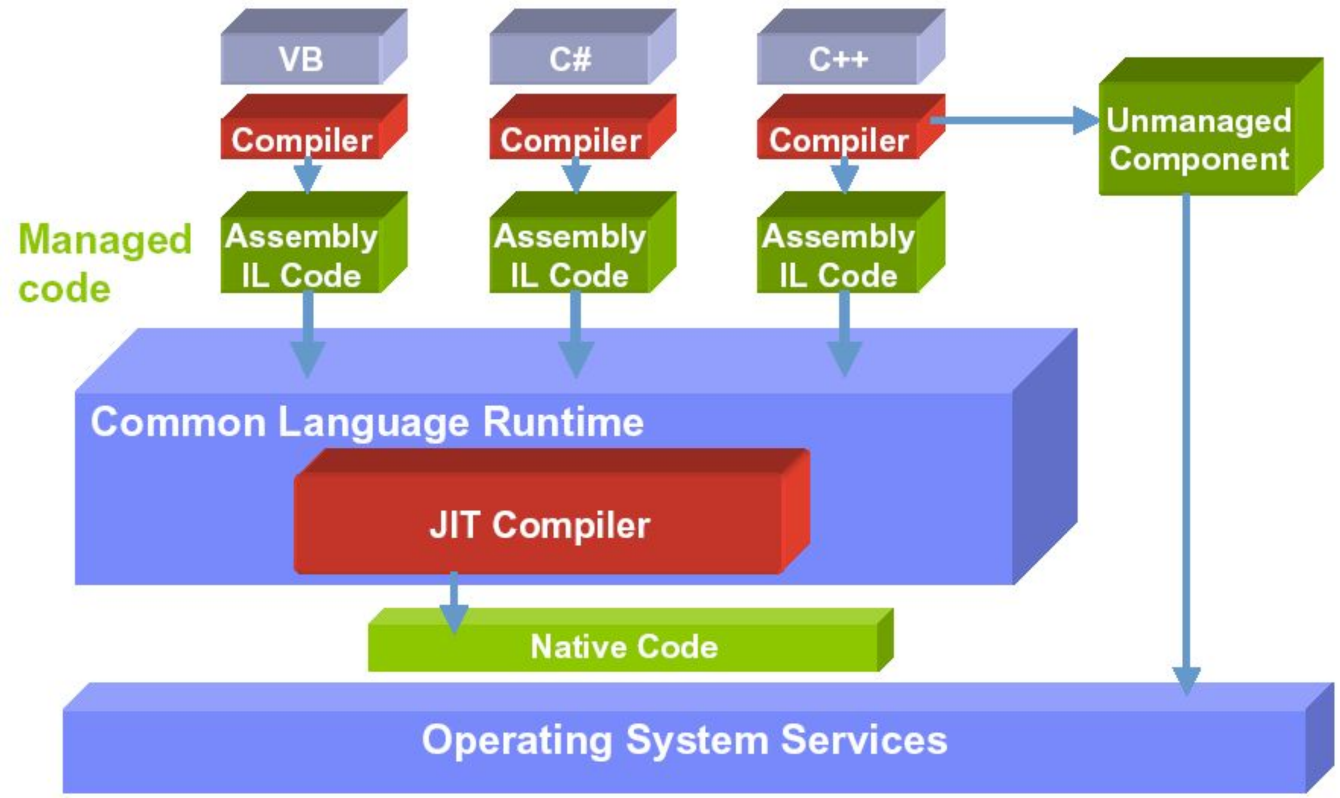
| Version number | CLR version | Release date | Support ended | Development tool | Included in |  | Replaces |
|---|---|---|---|---|---|---|---|
|  |  |  |  |  | Windows | Windows Server |  |
| 1.0 | 1.0 | 2002-02-13 | 2009-07-14[22] | Visual Studio .NET[23] | XP SP1[a] | N/A | N/A |
| 1.1 | 1.1 | 2003-04-24 | 2015-06-14[22] | Visual Studio .NET 2003[23] | XP SP2, SP3[b] | 2003 | 1.0[24] |
| 2.0 | 2.0 | 2005-11-07 | 2011-07-12[22] | Visual Studio 2005[25] | N/A | 2003, 2003 R2,[26] 2008 SP2, 2008 R2 SP1 | N/A |
| 3.0 | 2.0 | 2006-11-06 | 2011-07-12[22] | Expression Blend[27][c] | Vista | 2008 SP2, 2008 R2 SP1 | 2.0 |
| 3.5 | 2.0 | 2007-11-19 | 2028-10-10[22] | Visual Studio 2008[28] | 7, 8, 8.1, 10[d] | 2008 R2 SP1 | 2.0, 3.0 |
| 4.0 | 4 | 2010-04-12 | 2016-01-12[22] | Visual Studio 2010[29] | N/A | N/A | N/A |
| 4.5 | 4 | 2012-08-15 | 2016-01-12[22] | Visual Studio 2012[30] | 8 | 2012 | 4.0 |
| 4.5.1 | 4 | 2013-10-17 | 2016-01-12[22] | Visual Studio 2013[31] | 8.1 | 2012 R2 | 4.0, 4.5 |
| 4.5.2 | 4 | 2014-05-05 | N/A[22] | N/A | N/A | N/A | 4.0–4.5.1 |
| 4.6 | 4 | 2015-07-20 | N/A[22] | Visual Studio 2015[32] | 10 v1507 | N/A | 4.0–4.5.2 |
| 4.6.1 | 4 | 2015-11-30[33] | N/A[22] | Visual Studio 2015 Update 1 | 10 v1511 | N/A | 4.0–4.6 |
| 4.6.2 | 4 | 2016-08-02[34] | N/A[22] |  | 10 v1607 | 2016 | 4.0–4.6.1 |
| 4.7 | 4 | 2017-04-05[35] | N/A[22] | Visual Studio 2017 | 10 v1703 | N/A | 4.0–4.6.2 |
| 4.7.1 | 4 | 2017-10-17[36] | N/A[22] | Visual Studio 2017 | 10 v1709 | 2016 v1709 | 4.0–4.7 |
| 4.7.2 | 4 | 2018-04-30[37] | N/A[22] | Visual Studio 2017 | 10 v1803 | 2019 | 4.0–4.7.1 |
| 4.8 | 4 | Developing[38] | N/A | Visual Studio 2019 (Planning)[39] | 10 v1903 (Planning) | N/A | 4.0–4.7.2 |

*Từ Wikipedia, last update: Feb 2019*

# Things to look forward

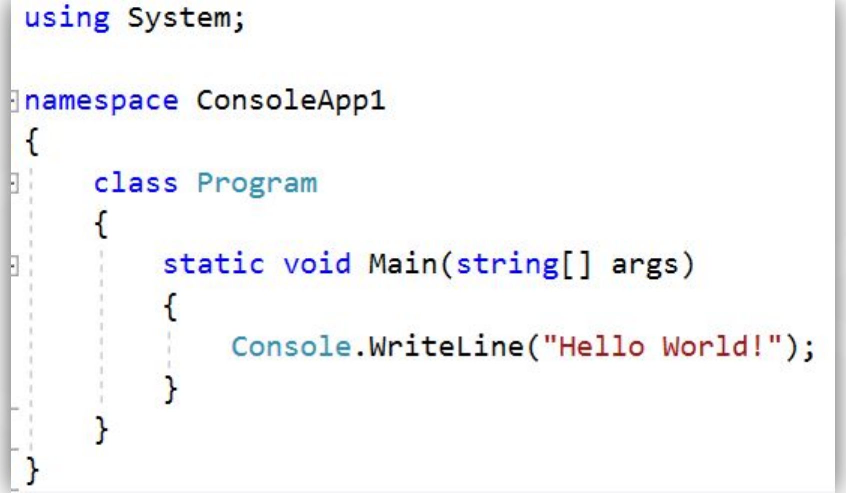
- .Net 5

# Cơ chế

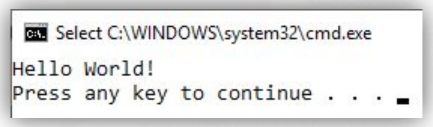

VB
C#
C++
Compiler
Compiler
Compiler
Unmanaged Component
Managed code
Assembly IL Code
Assembly IL Code
Assembly IL Code
Common Language Runtime
JIT Compiler
Native Code
Operating System Services

# Hello world

- Create a new console app



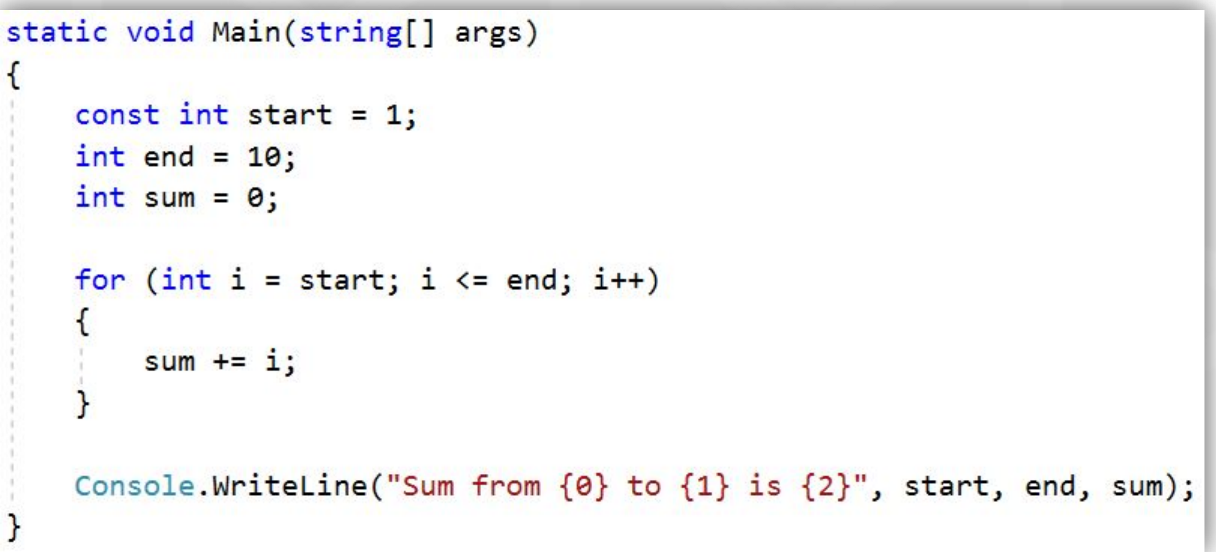
```
using System;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```



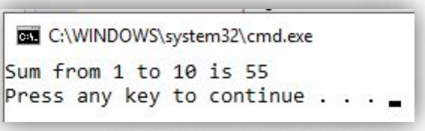
```
Hello World!
Press any key to continue . . .
```

# Constants & Variables

- Write a program that calculate sum from 1 to 10



```
static void Main(string[] args)
{
    const int start = 1;
    int end = 10;
    int sum = 0;

    for (int i = start; i <= end; i++)
    {
        sum += i;
    }

    Console.WriteLine("Sum from {0} to {1} is {2}", start, end, sum);
}
```



C:\WINDOWS\system32\cmd.exe

```
Sum from 1 to 10 is 55
Press any key to continue . . .
```

# Messing around

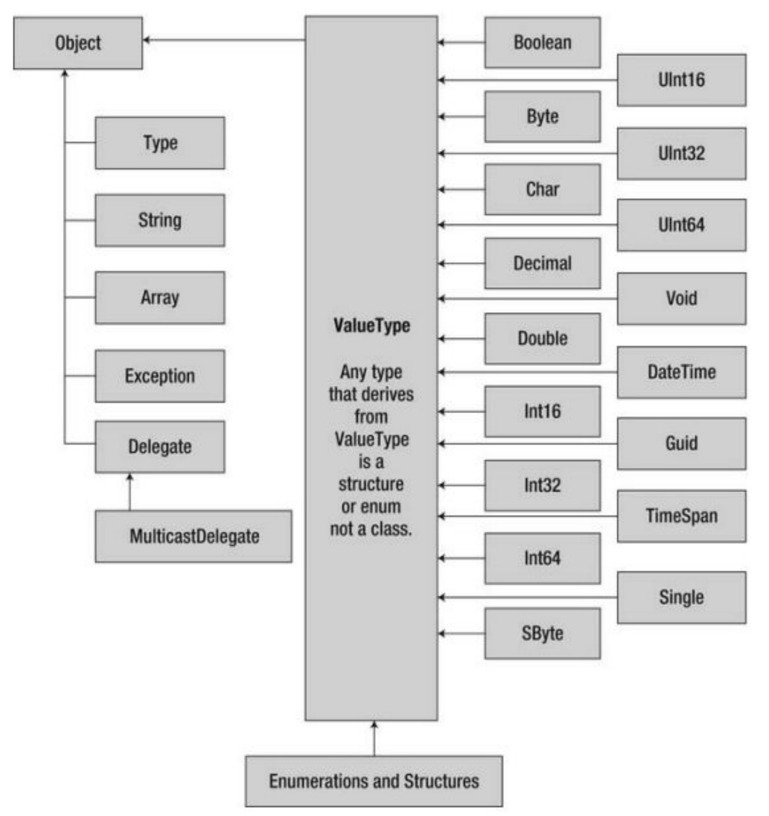
- Position of parameter for string format
- Function examples
  - ref
  - out

# Artifact

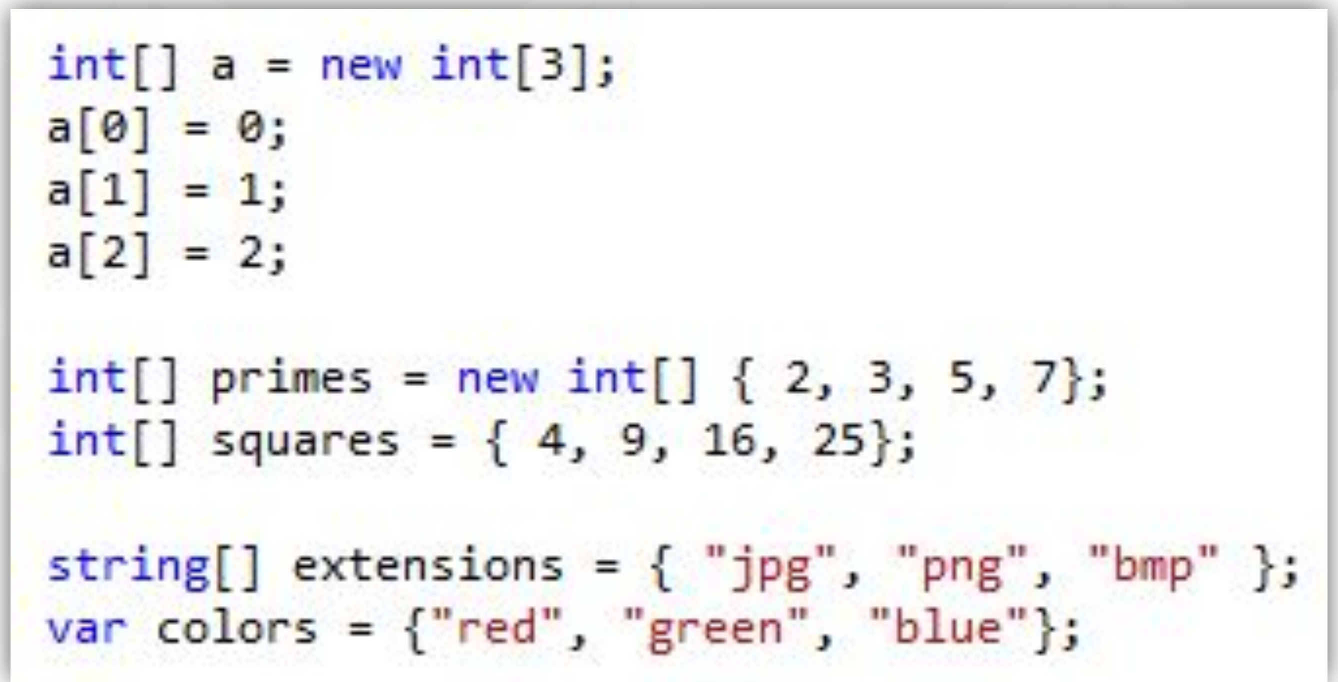
- Build => Error
- Run vs Run without debugging
- Release
- Debug
  - Breakpoint & watch
  - Step in & Step over
  - Watch

- Set as startup project
- Open project location to get build artifacts (exe file)

# Types

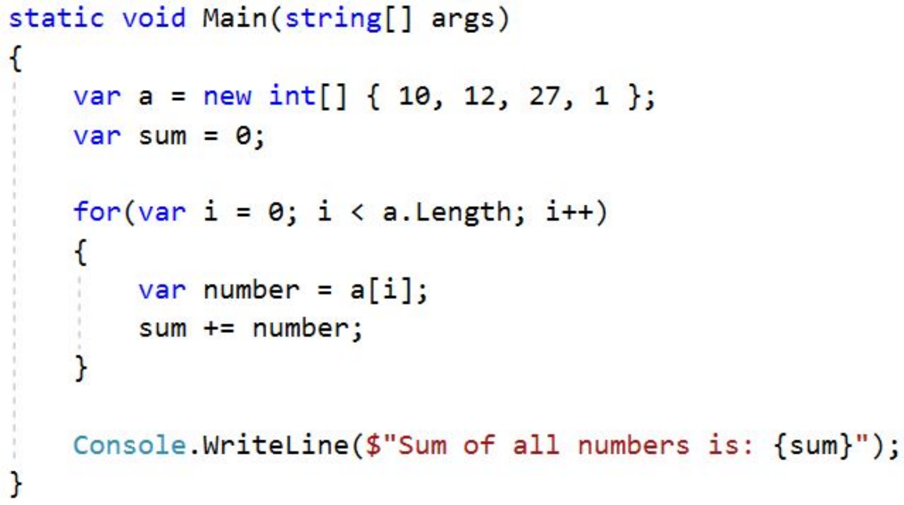

Object
Type
String
Array
Exception
Delegate
MulticastDelegate
ValueType
Any type that derives from ValueType is a structure or enum not a class.
Boolean
UInt16
Byte
UInt32
Char
UInt64
Decimal
Void
Double
DateTime
Int16
Guid
Int32
TimeSpan
Int64
Single
SByte
Enumerations and Structures

# Array

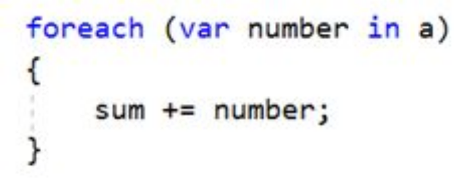


```
int[] a = new int[3];
a[0] = 0;
a[1] = 1;
a[2] = 2;

int[] primes = new int[] { 2, 3, 5, 7};
int[] squares = { 4, 9, 16, 25};

string[] extensions = { "jpg", "png", "bmp" };
var colors = {"red", "green", "blue"};
```

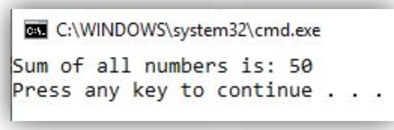
# Loop through an array



```
static void Main(string[] args)
{
    var a = new int[] { 10, 12, 27, 1 };
    var sum = 0;

    for(var i = 0; i < a.Length; i++)
    {
        var number = a[i];
        sum += number;
    }

    Console.WriteLine($"Sum of all numbers is: {sum}");
}
```



```
foreach (var number in a)
{
    sum += number;
}
```



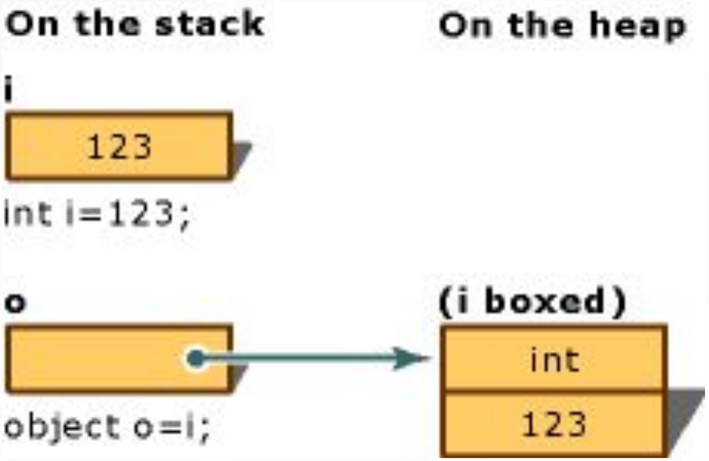
```
Sum of all numbers is: 50
Press any key to continue . . .
```

# Number casting

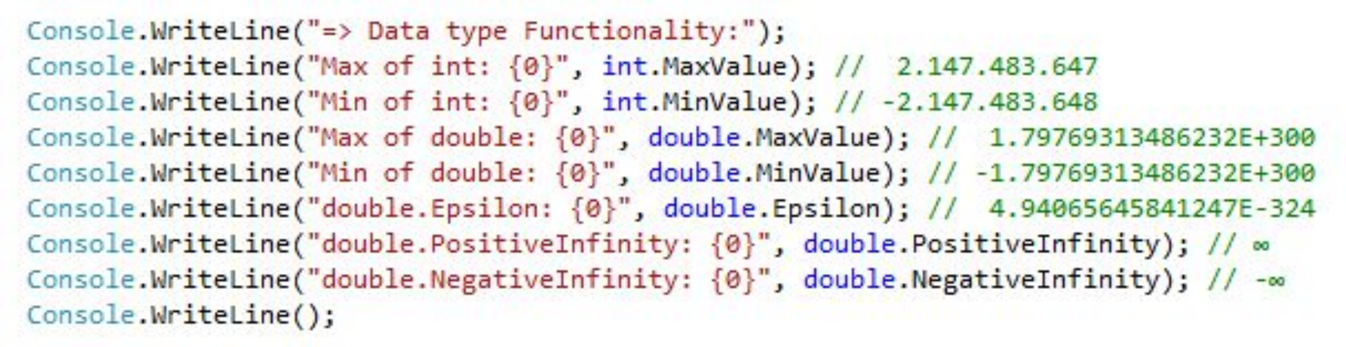
```
// Small to big -> OK
byte b = 5; // implicit int to byte
int i = b;  // i = 5
// Big to small -> NOT OK
int i = 500;
byte j = i; // compile error
byte j = (byte)i; // i = 244
```

# Boxing / Unboxing int/float/string Int16 String

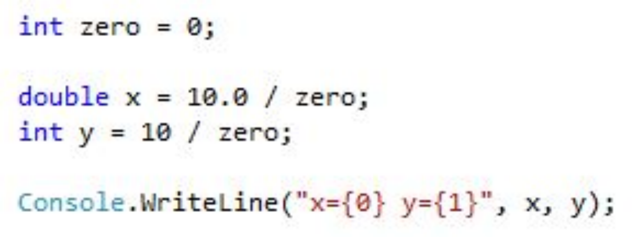
- ❑ Boxing: convert value type to object
- ❑ Unboxing: extract value type from object



# Minimum & maximum value



```
Console.WriteLine("=> Data type Functionality:");
Console.WriteLine("Max of int: {0}", int.MaxValue); //  2.147.483.647
Console.WriteLine("Min of int: {0}", int.MinValue); // -2.147.483.648
Console.WriteLine("Max of double: {0}", double.MaxValue); //  1.79769313486232E+300
Console.WriteLine("Min of double: {0}", double.MinValue); // -1.79769313486232E+300
Console.WriteLine("double.Epsilon: {0}", double.Epsilon); //  4.94065645841247E-324
Console.WriteLine("double.PositiveInfinity: {0}", double.PositiveInfinity); // ∞
Console.WriteLine("double.NegativeInfinity: {0}", double.NegativeInfinity); // -∞
Console.WriteLine();
```



```
int zero = 0;

double x = 10.0 / zero;
int y = 10 / zero;

Console.WriteLine("x={0} y={1}", x, y);
```

Result?

# Operation 1/2

- ❑ References: . () [] new ->
- ❑ Arithmetic: + ++ - -- * / % sizeof
- ❑ Logical: & | ^ !
- ❑ Conditional: && (&) || ! == != > >= < <=
- ❑ Type verification: is as typeof
- ❑ Bitwise: ~ >> <<
- ❑ Assignment: = += -= *= /= %= &= |= ^= >>= <<=
- ❑ Selection: ?: ?? (not null)
- ❑ Lambda expression definition: =>

# Operation 2/2

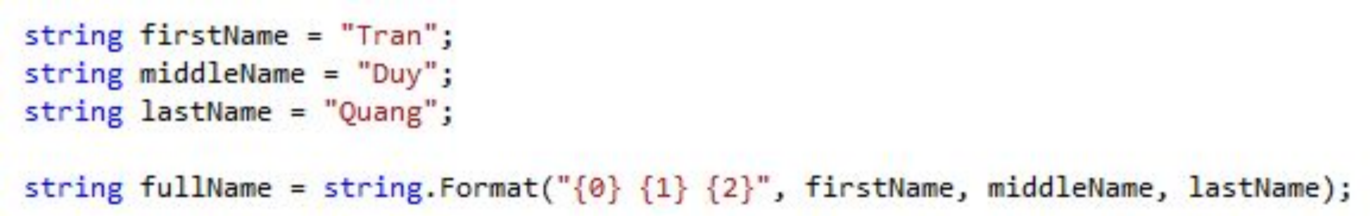
```
int i = 5;
// Selection ?:
string x = i == 5 ? "Yes": "No";

int? x = 5;     // nullable type
int y = x ?? 0; // Selection ?? operator
int z = x;      // Error: nullable type
     // cannot assign to non-nullable type

// Lambda expression - anonymous method
(int x) => x * 2; <=>
public int Double(int x) => x * 2;}
```

# String Format

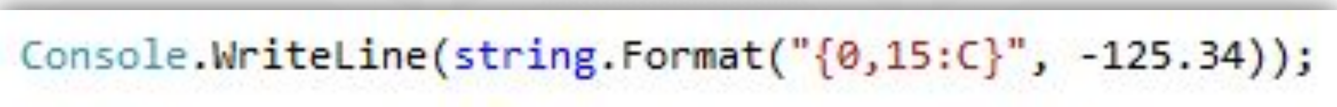
```
string header = String.Format(
  "{0,-12}{1,8}{2,12}{1,8}{2,12}{3,14}\n",     // pattern string
  "City", "Year", "Population", "Change (%)"); // argument list

string body = String.Format(
"{0,-12}{1,8:yyyy}{2,12:N0}{3,8:yyyy}{4,12:N0}{5,14:P1}",
Name, BaseYear, BasePopulation, ObserveYear, ObservePopulation,
ObservePopulation - BasePopulation)/ (double) BasePopulation);

// Sample output
City            Year  Population    Year  Population    Change (%)
------------------------------------------------------------------
Los Angeles     1940   1,504,277    1950   1,970,358        31.0 %
New York        1940   7,454,995    1950   7,891,957         5.9 %
Chicago         1940   3,396,808    1950   3,620,962         6.6 %
Detroit         1940   1,623,452    1950   1,849,568        13.9 %
```

# String format

- Positive: right align, negative: left align



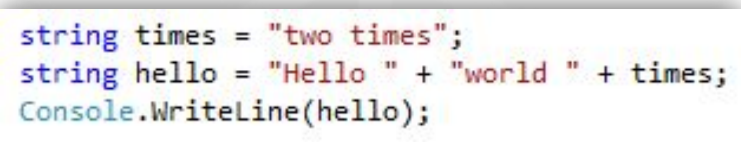
```
string firstName = "Tran";
string middleName = "Duy";
string lastName = "Quang";

string fullName = string.Format("{0} {1} {2}", firstName, middleName, lastName);
```



```
Console.WriteLine(string.Format("{0,15:C}", -125.34));
```

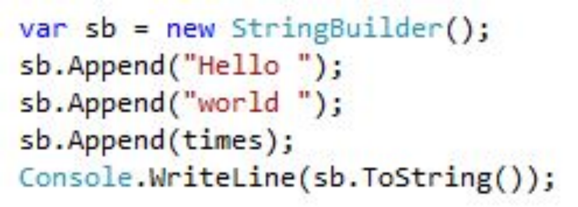
# String concatenation

- Use “+” character



```
string times = "two times";
string hello = "Hello " + "world " + times;
Console.WriteLine(hello);
```

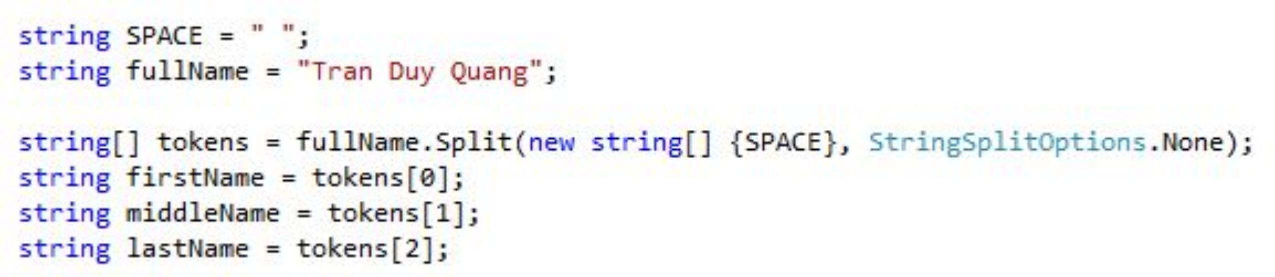
- Use StringBuilder



```
var sb = new StringBuilder();
sb.Append("Hello ");
sb.Append("world ");
sb.Append(times);
Console.WriteLine(sb.ToString());
```

- Question: what should we use, + or StringBuilder?

# String Split



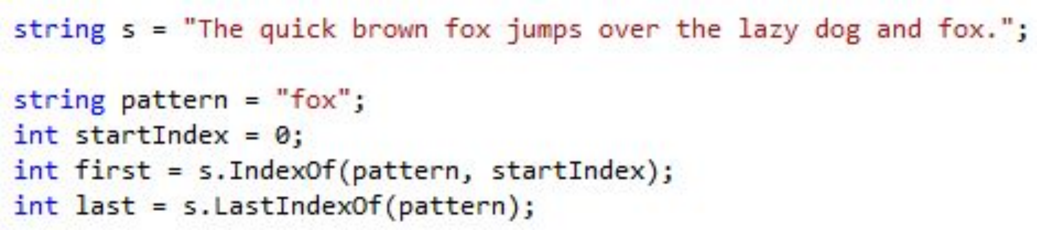
```
string SPACE = " ";
string fullName = "Tran Duy Quang";

string[] tokens = fullName.Split(new string[] {SPACE}, StringSplitOptions.None);
string firstName = tokens[0];
string middleName = tokens[1];
string lastName = tokens[2];
```

## Exercise

- **Calculate sum of string numbers = “5, 3, 8, 11, -12, 3”**
- Split String fraction = “3/4” into int numerator and denominator

  What if we meet 3//4?

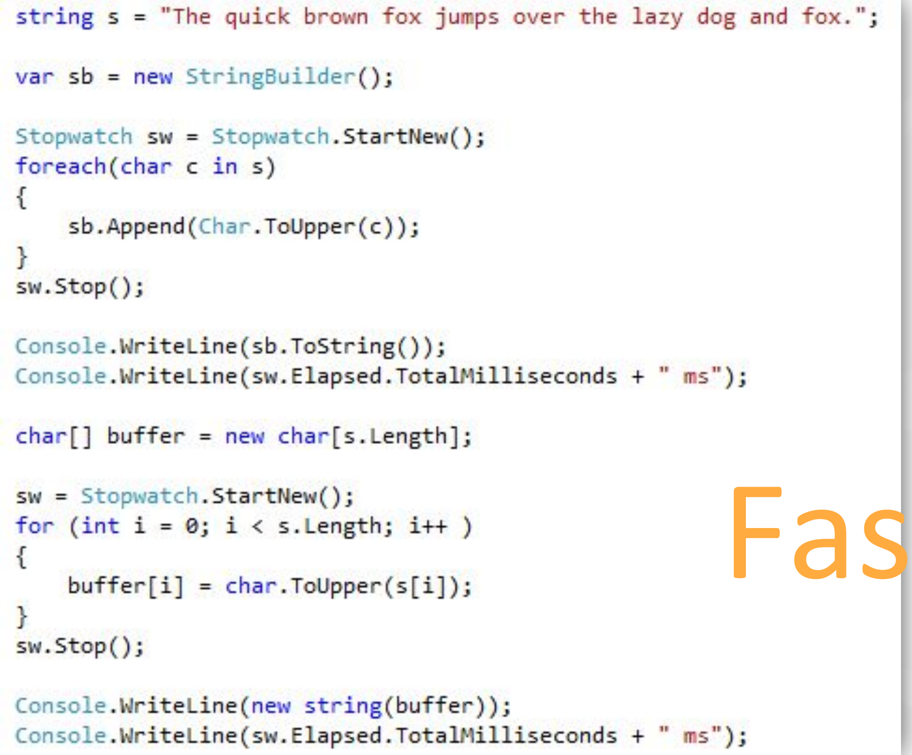
# String search



```
string s = "The quick brown fox jumps over the lazy dog and fox.";

string pattern = "fox";
int startIndex = 0;
int first = s.IndexOf(pattern, startIndex);
int last = s.LastIndexOf(pattern);
```

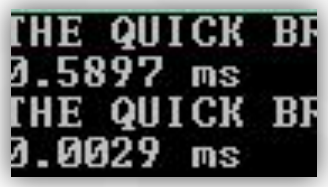
- Exercise
  - Given string s = “She sells seashells by the seashore. The shells she sells are seashells”
  - Calculate the number of occurrence of the word “sells” and “she”
- Further reading: replace and regular expression

# String as char array



```
string s = "The quick brown fox jumps over the lazy dog and fox.";

var sb = new StringBuilder();

Stopwatch sw = Stopwatch.StartNew();
foreach(char c in s)
{
    sb.Append(Char.ToUpper(c));
}
sw.Stop();

Console.WriteLine(sb.ToString());
Console.WriteLine(sw.Elapsed.TotalMilliseconds + " ms");

char[] buffer = new char[s.Length];

sw = Stopwatch.StartNew();
for (int i = 0; i < s.Length; i++ )
{
    buffer[i] = char.ToUpper(s[i]);
}
sw.Stop();

Console.WriteLine(new string(buffer));
Console.WriteLine(sw.Elapsed.TotalMilliseconds + " ms");
```

Faster!



```
THE QUICK BR
0.5897 ms
THE QUICK BR
0.0029 ms
```

# String Exercises

1. Read a string and give statistics about the number of occurrence for each of the word in the string.

2. Normalize a string of full name and print out on the screen: no more than one spaces between words, the first letter is capitalized meanwhile the rest are in lower case, no space in the beginning and the end of the string.

3. Split String fullpath = “C:\Documents\Photos\Test.jpg” into a) Containing directory b) File name c) Extension

# DateTime

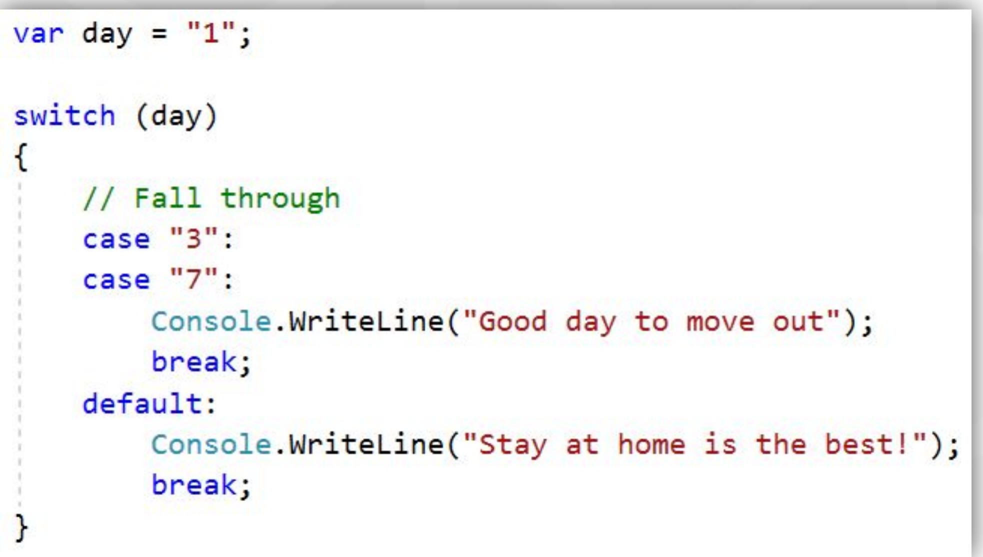
```
using System;
DateTime a = new DateTime();

// Full construction
DateTime b = new DateTime(2013, 06, 15, 15, 28, 31, 927);

// Current time
DateTime c = DateTime.Now;
```

# Flow control

- ❑ Branching
  - ❑ Selection: ?: ??
  - ❑ if… else if … else
  - ❑ switch … case … default
- ❑ Iteration
  - ❑ for
  - ❑ foreach
  - ❑ do while
  - ❑ while

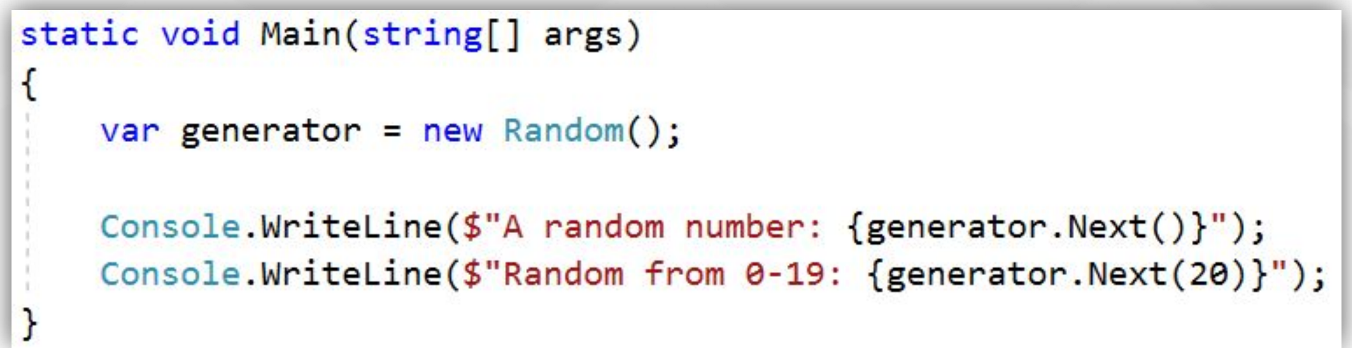
- ❑ Ignore & breaking
  - ✔ continue
  - ✔ break

# switch case example



```
var day = "1";

switch (day)
{
    // Fall through
    case "3":
    case "7":
        Console.WriteLine("Good day to move out");
        break;
    default:
        Console.WriteLine("Stay at home is the best!");
        break;
}
```

# Random generator



```
static void Main(string[] args)
{
    var generator = new Random();

    Console.WriteLine($"A random number: {generator.Next()}");
    Console.WriteLine($"Random from 0-19: {generator.Next(20)}");
}
```

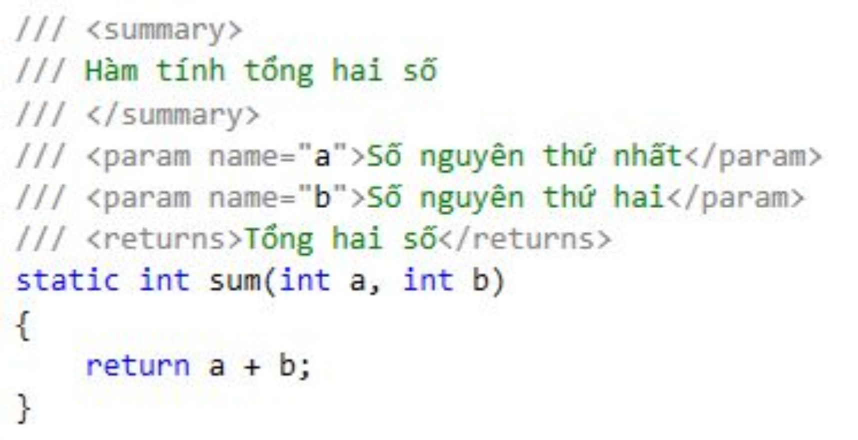
# Basic coding convention

# if for do while

- ❑ Add an empty line before and after
- ❑ Always use brackets even if there is one line of code

# Comment

- Should provide purpose of a block of function
- XML comment for document generation (doxygen)



```
/// <summary>
/// Hàm tính tổng hai số
/// </summary>
/// <param name="a">Số nguyên thứ nhất</param>
/// <param name="b">Số nguyên thứ hai</param>
/// <returns>Tổng hai số</returns>
static int sum(int a, int b)
{
    return a + b;
}
```

# Function name

- Start with a verb
- Private: camelCase
- Public: PascalCase

- Ex: isPrime, doSomething

- Quiz: Check if a element exists in an array
  - kiemTraTonTai, checkExistence

# Constants

- PascalCase

# Exercises

- Check if a number is a prime number
  - bool IsPrime(int number) / LaSoNguyenTo()
- Calculate $x^n$
  - double Pow(double x, int n) / LuyThua

# Exercises

- ❑ Given a valid date in 3 variables: day, month, year
- ❑ Write a function that print out the next day

# Basic UI

# WPF

- ❑ Create WPF app
- ❑ UI is written in xaml language
- ❑ Code behind is C#

# 4 basic controls

- ❑ Label
- ❑ TextBox
- ❑ Button (Click event)
- ❑ Image

- ❑ MessageBox

# Homework 01 - C# cơ bản bằng console

Nhập vào 3 số nguyên vào 3 biến day, month, year

1. Kiểm tra 3 biến này tạo thành một ngày hợp lệ
2. In ra màn hình ngày kế

# Homework 02 - UI (chọn 1 trong 2)

1. Hiển thị ngẫu nhiên một câu trích dẫn tạo động lực kèm hình (chọn từ tối thiểu mảng 10 phần tử). Ví dụ: "Không có gì là không thể".
Nguồn: https://motivationping.com/quotes/

2. Hiển thị ngẫu nhiên một cặp hình & từ tiếng Anh (chủ đề từ chọn, ví dụ: Động vật, chọn từ tối thiểu mảng 10 phần tử)