

# Ngôn ngữ lập trình C#

# Nội dung

---

- Giới thiệu về .NET Framework
- Ngôn ngữ lập trình C# :
  - Biến và kiểu dữ liệu
  - Chú thích
  - Hằng
  - Từ khóa
  - Xuất/nhập
  - Biểu thức
  - Các câu lệnh trong C#

# .NET Framework

---

- Là một thành phần phần mềm được thêm vào hệ điều hành (Windows).
  - Chứa đựng những thư viện có sẵn.
  - Quản lý việc thực thi chương trình viết dưới nền tảng .NET
- Phiên bản được sử dụng trong môn học: .NET Framework 4.7 (đi kèm với Visual Studio 2015).

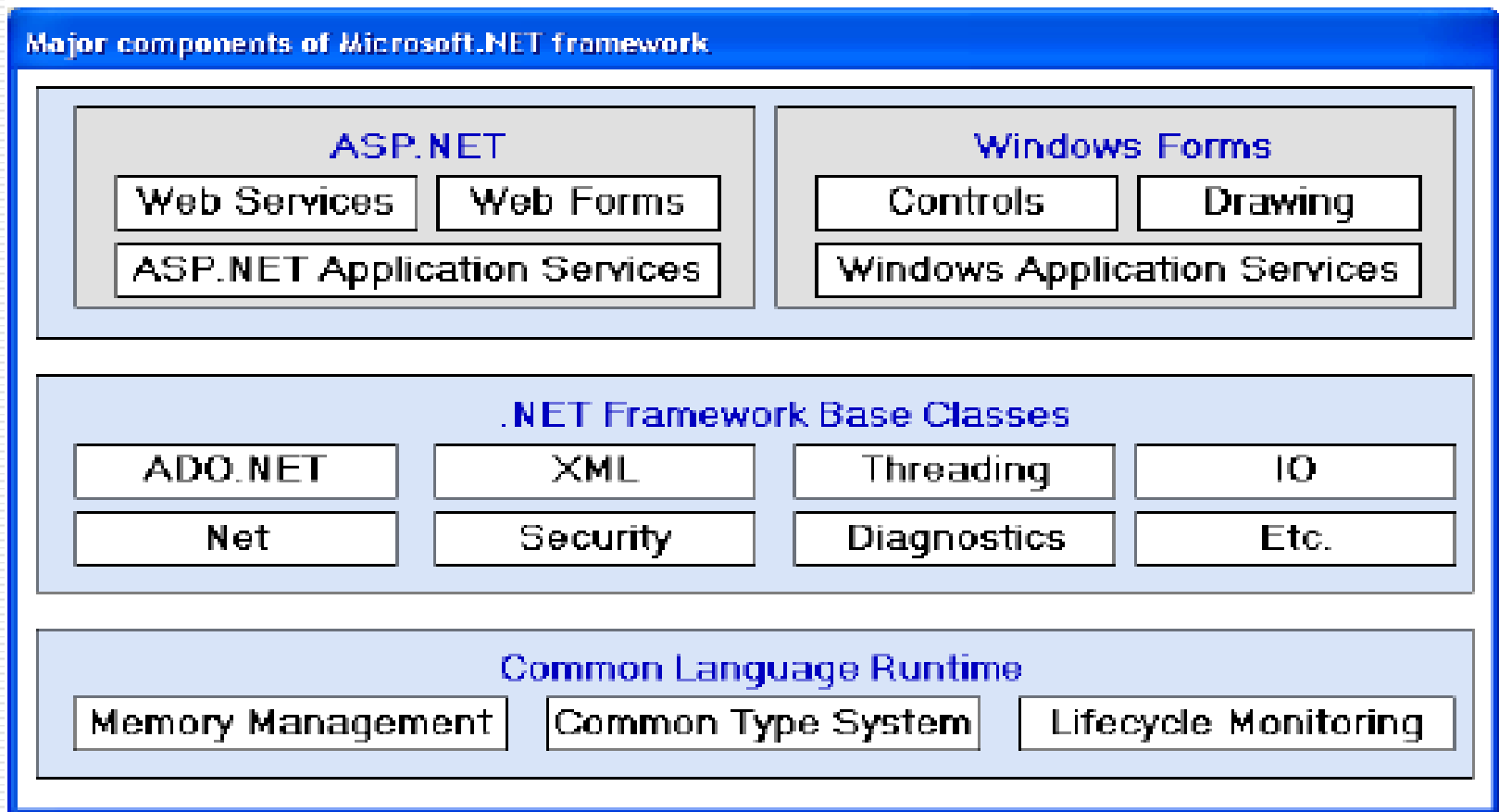
# Các thành phần của .NET Framework [1]

---

- .NET Framework gồm 2 phần chính:
  - Thư viện lớp cơ sở của .NET Framework: tập hợp các lớp thư viện được định nghĩa sẵn.
  - Common Language Runtime (CLR): quản lý việc thực thi chương trình: quản lý bộ nhớ, thực thi mã lệnh, bắt lỗi, cấp phát và thu hồi vùng nhớ, ...

# Các thành phần của .NET Framework [2]

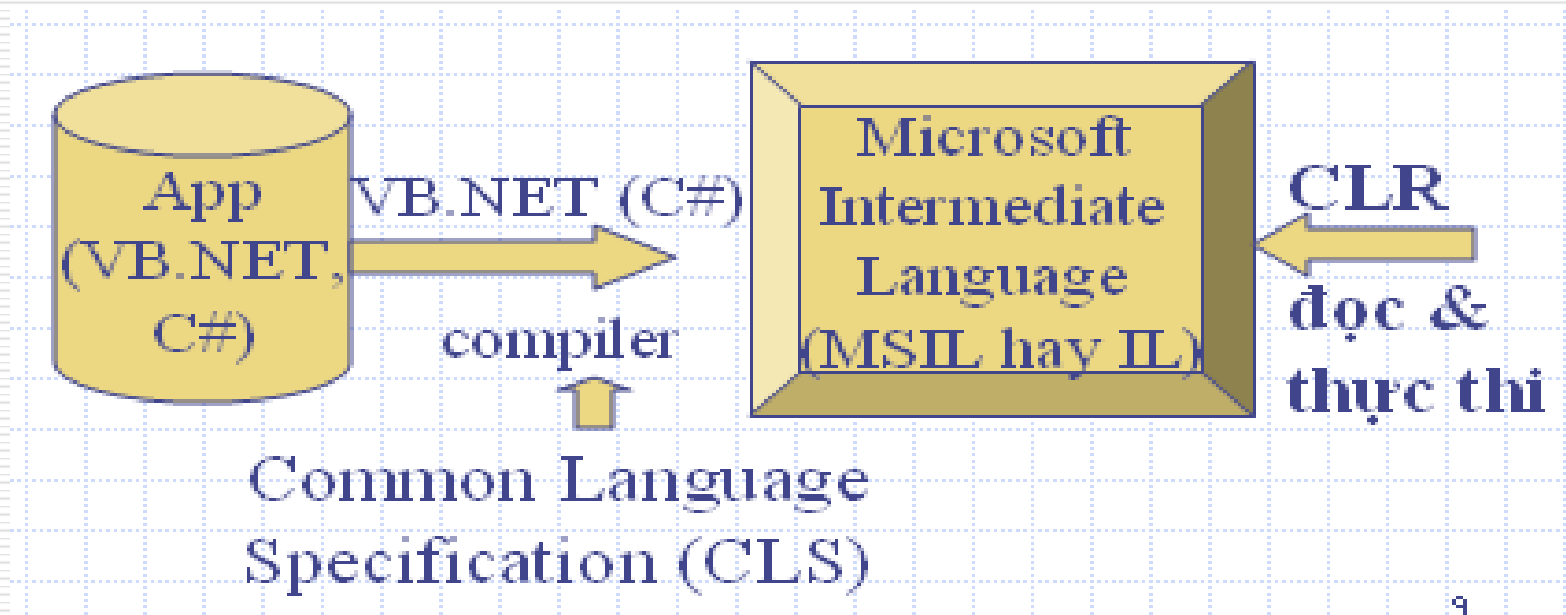
---



# Ngôn ngữ và .NET Framework

---

- Tương thích giữa các ngôn ngữ lập trình với nhau (do CLR).



# Ngôn ngữ lập trình C#

---

- C# là một ngôn ngữ lập trình thuần hướng đối tượng được thiết kế để xây dựng các ứng dụng chạy dưới nền tảng .NET Framework.



# Chương trình C# đầu tiên

---

- ❑ Soạn thảo chương trình **Hello.cs** sau:

```
class Hello
{
    static void Main()
    {
        System.Console.WriteLine("Hello World");
    }
}
```

- ❑ Biên dịch chương trình:  
**csc <Tên chương trình>**
- ❑ Sau khi biên dịch ta được 1 tập tin thực thi (exe), thực thi tập tin này



---

# **Biến và kiểu dữ liệu**

# Biến (Variable)

---

- ❑ Biến là đại lượng chứa dữ liệu trong quá trình tính toán.
- ❑ Giá trị của biến có thể thay đổi.
- ❑ Khai báo biến:
  - <Kiểu dữ liệu> <Tên biến>;
  - **Thí dụ:**     **int** empNumber;  
                  **string** empName;

# Kiểu dữ liệu (Data Types) [1]

---

- Một kiểu dữ liệu là một tập hợp các giá trị và tập hợp các phép toán thao tác trên các giá trị đó.
- Trong .NET, kiểu dữ liệu được chia làm 2 loại:
  - Kiểu giá trị
  - Kiểu tham chiếu

# Kiểu dữ liệu (Data Types) [2]

---

- ❑ Kiểu giá trị: biến có kiểu giá trị lưu giá trị thực sự trong stack.
- ❑ Kiểu tham chiếu: biến có kiểu tham chiếu lưu giá trị thực sự trong heap, bên cạnh đó tham chiếu đến biến đó được ghi nhận trong stack.
- ❑ Cả 2 loại kiểu giá trị hay kiểu tham chiếu có thể là kiểu có sẵn hoặc do người lập trình định nghĩa.

# Các kiểu sơ cấp chuẩn

Tên kiểu	Miền giá trị
byte	0..255
short	-32,768..32,767
int	-2,147,483,648.. 2,147,483,647
long	$-2^{63} .. 2^{63}-1$
float	$\pm 1.5e-45 .. \pm 3.4e38$
double	$\pm 5.0e-324 .. \pm 1.7e308$
decimal	$\pm 1.0 \times 10e-28 .. \pm 7.9 \times 10e28$
char	U+0000 .. U+ffff
bool	true, false

# Các kiểu tham chiếu

---

- ❑ Object: là kiểu cơ sở của mọi kiểu khác trong C#.
- ❑ String: là kiểu tham chiếu có sẵn cho phép các biến kiểu này có thể lưu trữ dữ liệu chuỗi ký tự.
- ❑ Class: là kiểu do người lập trình định nghĩa (chương 2)
- ❑ Delegate: là kiểu do người lập trình định nghĩa cho phép các biến kiểu này tham chiếu đến một hay một số phương thức.
- ❑ Interface: kiểu do người lập trình định nghĩa (chương 2)
- ❑ Array: kiểu do người lập trình định nghĩa cho phép các biến kiểu này chứa các phần tử là những giá trị cùng kiểu

# Quy tắc đặt tên

---

- ❑ Một tên chỉ có thể chứa ký tự, ký số và dấu gạch dưới.
- ❑ Bắt đầu phải là 1 ký tự hoặc là dấu gạch dưới.
- ❑ Tên không được trùng với từ khóa.
- ❑ Ngôn ngữ C# phân biệt ký tự hoa thường, do đó **Count** và **count** là 2 tên khác nhau.

# Tên hợp lệ và không hợp lệ

---

Hợp lệ	Không hợp lệ
Employee	12A2
student	class
Name	Tom&Jerry
_EmpName	Lion King



# Khai báo biến và gán trị cho biến

---

- ❑ Một biến trước khi sử dụng cần được gán trị.

<Tên biến> = <Giá trị>;

■ **Thí dụ:**     **string** emp\_Name;  
                  emp\_Name = "Thomas";

- ❑ Vừa khai báo vừa gán trị:

<Kiểu> <Tên biến> = <Giá trị>;

■ **Thí dụ:**     **string** emp\_Name = "Thomas";

---

# Chú thích

# Chú thích (Comment)

```
1 Program
2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace Thidul
6 {
7     class Program
8     {
9         static void Main(string[] args)
10        {
11            /*
12             * Phương thức WriteLine cho phép hiển thị giá trị
13             * của một biểu thức lên màn hình
14             */
15            Console.WriteLine("Hello World");
16            Console.ReadLine();//Cho phím Enter tu ban phim
17        }
18    }
19 }
```

---

# Hằng

# Hằng (Constant)

---

- Hằng là đại lượng có giá trị không đổi trong suốt quá trình thực hiện của chương trình.
  - Hằng có tên
  - Hằng trực tiếp

# Hằng có tên

---

□ Khai báo:

**const** <Kiểu hằng> <Tên hằng> = <Giá trị>;

□ Thí dụ:

**const float** Pi = 3.14F;

**const float** g = 9.8f;

# Hằng trực tiếp – Literal [1]

---

- ❑ Boolean Literal: gồm có 2 giá trị **true** và **false**.
- ❑ Integer literal: là những giá trị thuộc kiểu int, uint, long, ulong. Các giá trị này được viết theo dạng số nguyên thông thường hoặc kết thúc bằng U (uint, ulong), L (long), UL hay LU (ulong).
  - Ví dụ:        50, 120L, 56UL

# Hằng trực tiếp – Literal [2]

---

- ❑ Real literal: là những giá trị thuộc kiểu float, double, decimal. Các giá trị này được viết như dạng số thực thông thường hoặc kết thúc bằng F (float), D (double), M (decimal).
  - **Thí dụ:** 3.14, 9.8F, 12.0M
- ❑ Character literal: là giá trị thuộc kiểu char. Các giá trị này được viết trong cặp nháy đơn (`)`
  - **Thí dụ:**       `a`, `B`



# Hằng trực tiếp – Literal [3]

---

- ❑ String literal: là những giá trị thuộc kiểu string. Các giá trị này được viết trong cặp dấu nháy kép ("")
  - **Thí dụ:** "Đại học Cần Thơ"
- ❑ Null literal: là giá trị rỗng. Các biến kiểu tham chiếu có thể nhận giá trị này.
  - **Thí dụ:**     **string** email = null;

---

# Từ khóa

# Từ khóa (Keyword)

---

- ❑ Là từ dành riêng của ngôn ngữ.
- ❑ Mỗi từ khóa có 1 ý nghĩa nhất định

# Từ khóa (Keyword)

---

<code>abstract</code>	<code>bool</code>	<code>break</code>	<code>byte</code>	<code>case</code>	<code>catch</code>
<code>char</code>	<code>class</code>	<code>const</code>	<code>continue</code>	<code>default</code>	<code>double</code>
<code>enum</code>	<code>else</code>	<code>false</code>	<code>finally</code>	<code>float</code>	<code>for</code>
<code>foreach</code>	<code>goto</code>	<code>if</code>	<code>int</code>	<code>interface</code>	<code>long</code>
<code>namespace</code>	<code>new</code>	<code>public</code>	<code>private</code>	<code>protected</code>	<code>return</code>
<code>sbyte</code>	<code>short</code>	<code>static</code>	<code>string</code>	<code>struct</code>	<code>switch</code>
<code>throw</code>	<code>true</code>	<code>try</code>	<code>ushort</code>	<code>void</code>	<code>while</code>

---

# **Xuất/Nhập**

# Output - Xuất

---

- ❑ 2 phương thức dùng để xuất giá trị của biểu thức lên màn hình:
  - `Console.Write()`
  - `Console.WriteLine()`
- ❑ Tham số của 2 phương thức này là 1 biểu thức có kiểu bất kỳ

# Output - Xuất

---

```
Console.WriteLine("C# is a powerful programming language");
```

```
Console.WriteLine("C# is a powerful");  
Console.WriteLine("programming language");
```

```
Console.Write("C# is a powerful");  
Console.WriteLine(" programming language");
```

## Output:

```
C# is a powerful programming language  
C# is a powerful  
programming language  
C# is a powerful programming language
```

# Giữ chỗ (Placeholders)

---

- Tổng quát, các phương thức `Console.Write()` và `Console.WriteLine()` có thể có nhiều hơn 1 tham số:
  - Tham số đầu tiên là 1 chuỗi có những điểm đánh dấu trong cặp `{}` chỉ vị trí của giá trị các biểu thức là những tham số kế tiếp sẽ hiển thị.
  - Các tham số kế tiếp là các biểu thức.
  - Các điểm đánh dấu được viết bắt đầu là 0. Chẳng hạn: `{0}`, `{1}`



# Giữ chỗ (Placeholders)

---

```
int number, result;  
number = 5;  
result = 100 * number;  
Console.WriteLine("Result is {0} when 100 is multiplied by {1}",  
    result, number);  
result = 150 / number;  
Console.WriteLine("Result is {0} when 150 is divided by {1}",  
    result, number);
```

Result

Number

# Input - Nhập

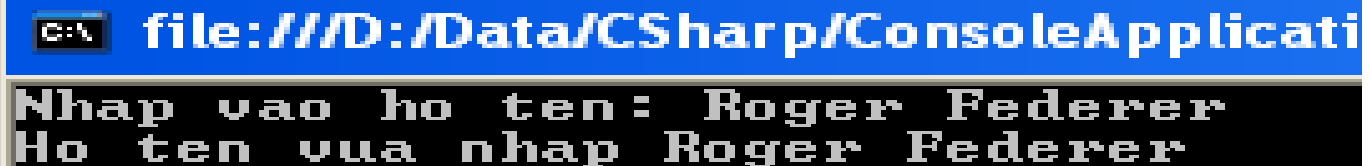
---

- Cho phép nhận giá trị từ bàn phím:
  - `Console.Read()`: nhận 1 ký tự từ bàn phím
  - `Console.ReadLine()`: nhận 1 chuỗi ký tự từ bàn phím.

# Input – Nhập

---

```
class Program
{
    static void Main(string[] args)
    {
        string emp_Name;
        Console.Write("Nhập vào họ tên: ");
        emp_Name = Console.ReadLine();
        Console.WriteLine("Họ tên vừa nhập {0}", emp_Name);
        Console.ReadLine();
    }
}
```



C> file:///D:/Data/CSharp/ConsoleApplicati  
Nhập vào họ tên: Roger Federer  
Họ tên vừa nhập Roger Federer

# Chuyển đổi kiểu - Lớp Convert

---

- ❑ Phương thức `Console.ReadLine()` chỉ nhận 1 chuỗi ký tự từ bàn phím.
  - Cần phải chuyển đổi kiểu nếu muốn nhận giá trị thuộc kiểu khác
- ❑ Một số phương thức của lớp `Convert`:  
`ToInt32()`, `ToSingle()`, `ToDouble()`,  
`ToDecimal()`, `ToBoolean()`

# Chuyển đổi kiểu - Lớp Convert

```
class Program
{
    static void Main(string[] args)
    {
        string emp_Name;
        int emp_Age;
        double emp_Salary;
```

```
        Console.Write("Nhập vào họ tên: ");
        emp_Name = Console.ReadLine();
        Console.Write("Nhập vào tuổi: ");
        emp_Age = Convert.ToInt32(Console.ReadLine());
        Console.Write("Nhập vào lương: ");
        emp_Salary = Convert.ToDouble(Console.ReadLine());

        Console.WriteLine("Họ tên {0}; tuổi {1}; lương {2}", emp_Name, emp_Age, emp_Salary);
        Console.ReadLine();
    }
}
```

C:\ file:///D:/Data/CSharp/ConsoleApplication1/ConsoleAp

```
Nhập vào họ tên: Roger Federer
Nhập vào tuổi: 29
Nhập vào lương: 40000
Họ tên Roger Federer; tuổi 29; lương 40000
```

---

# **Biểu thức (Expression)**

# Biểu thức

---

- Là sự kết hợp giữa các toán hạng và toán tử nhằm tính toán ra 1 giá trị nhất định.
  - Toán hạng: biến, hằng (có tên hoặc trực tiếp).
  - Toán tử: các phép toán.
- Biểu thức là 1 phần của câu lệnh hoặc 1 biểu thức khác
- **Thí dụ:**  $P_i * R * R$

# Các toán tử số học

---

- Các toán tử số học cho phép tính toán trên các toán hạng là số hay chuỗi ký tự. Kết quả của việc tính toán là 1 số hay chuỗi.



# Các toán tử số học

---

Operators	Description	Examples
+ (Addition)	Performs addition. If the two operands are strings, then it functions as a string concatenation operator and adds one string to the end of the other.	40 + 20
- (Subtraction)	Performs subtraction. If a greater value is subtracted from a lower value, the resultant output is a negative value.	100 - 47
* (Multiplication)	Performs multiplication.	67 * 46
/ (Division)	Performs division. The operator divides the first operand by the second operand and gives the quotient as the output.	12000 / 10
% (Modulo)	Performs modulo operation. The operator divides the two operands and gives the remainder of the division operation as the output.	100 % 33

# Các toán tử quan hệ

---

- Các toán tử này nhằm so sánh giữa 2 toán hạng, kết quả là 1 giá trị đúng hoặc sai.

# Các toán tử quan hệ

---

Relational Operators	Descriptions	Examples
<code>==</code>	Checks whether the two operands are identical.	<code>85 == 95</code>
<code>!=</code>	Checks for inequality between two operands.	<code>35 != 40</code>
<code>&gt;</code>	Checks whether the first value is greater than the second value.	<code>50 &gt; 30</code>
<code>&lt;</code>	Checks whether the first value is lesser than the second value.	<code>20 &lt; 30</code>
<code>&gt;=</code>	Checks whether the first value is greater than or equal to the second value.	<code>100 &gt;= 30</code>
<code>&lt;=</code>	Checks whether the first value is lesser than or equal to the second value.	<code>75 &lt;= 80</code>

# Toán tử điều kiện

---

- ❑ AND (&&): toán tử này thực hiện and giữa 2 toán hạng bool, kết quả là 1 giá trị đúng hoặc sai.
- ❑ OR (||): toán tử này thực hiện or giữa 2 toán hạng bool, kết quả là 1 giá trị đúng hoặc sai.
- ❑ NOT (!): toán tử này thực hiện not của 1 toán hạng bool, kết quả là 1 giá trị đúng hoặc sai.

# Toán tử điều kiện

---

```
int num = -5;
if (num < 0 || num > 10)
{
    Console.WriteLine("The number does not exist
between 1 and 10");
}
else
{
    Console.WriteLine("The number exists between 1
and 10");
}
```

## Output:

The number does not exist between 1 and 10

# Toán tử tăng hoặc giảm

---

- ❑ Toán tử  $++$ : tăng giá trị của biến nguyên lên 1 giá trị.
- ❑ Toán tử  $--$ : giảm giá trị của biến nguyên 1 giá trị

# Toán tử tăng hoặc giảm

---

Expression	Type	Result
<code>valueTwo = ++ValueOne;</code>	Pre-Increment	<code>valueTwo = 6</code>
<code>valueTwo = valueOne++;</code>	Post-Increment	<code>valueTwo = 5</code>
<code>valueTwo = --valueOne;</code>	Pre-Decrement	<code>valueTwo = 4</code>
<code>valueTwo = valueOne--;</code>	Post-Decrement	<code>valueTwo = 5</code>

# Độ ưu tiên của các toán tử

Precedence (where 1 is the highest)	Operator	Description	Associativity
1	()	Parentheses	Left to Right
2	++ or --	Increment or Decrement	Right to Left
3	*, / , %	Multiplication, Division, Modulus	Left to Right
4	+, -	Addition, Subtraction	Left to Right
5	<, <=, >, >=	Less than, Less than or equal to, Greater than, Greater than or equal to	Left to Right
6	=, !=	Equal to, Not Equal to	Left to Right
7	&&	Conditional AND	Left to Right
8		Conditional OR	Left to Right
9	=, +=, -=, *=, /=, %=	Assignment Operators	Right to Left



---

# **Câu lệnh - Statement**

# Câu lệnh - Statement

---

- Câu lệnh là 1 nhóm các biến, biểu thức và từ khóa trong C# nhằm thực hiện 1 công việc nào đó.
- Trong C#, mỗi câu lệnh kết thúc bằng dấu ;
- Một dãy liên tiếp các khai báo và câu lệnh đặt trong dấu {} gọi là 1 khối lệnh

# Câu lệnh - Statement

---

```
class Circle
{
    static void Main(string[] args)
    {
        const float _pi = 3.14F;
        float radius = 5;
        float area = _pi * radius * radius;
        Console.WriteLine("Area of the circle is " + area);
    }
}
```

Block

# Câu lệnh if

---

```
if (condition)
{
    // one or more statements;
}
```

```
if (condition)
{
    // one or more statements;
}
else
{
    // one or more statements;
}
```

# Câu lệnh if

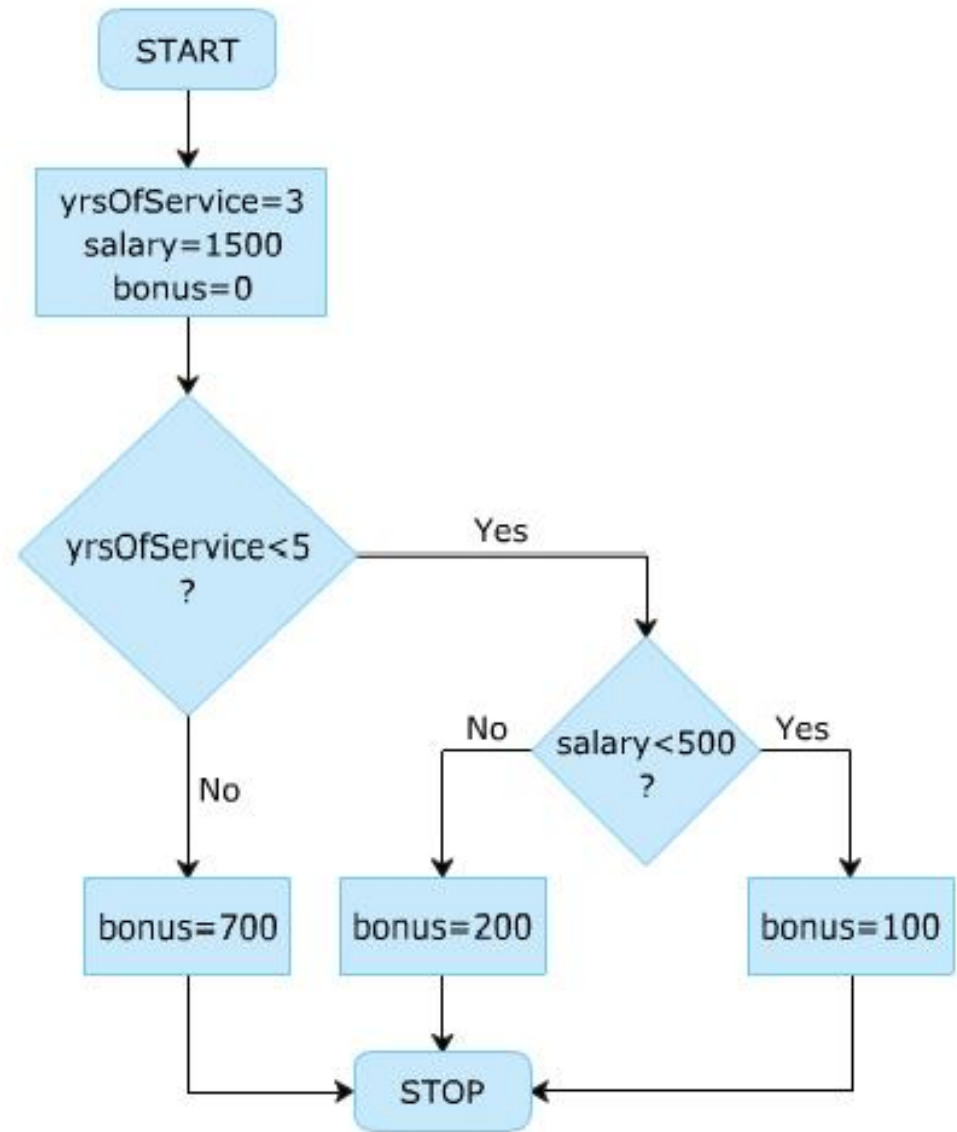
---

```
public int yrsOfService = 3;
public double salary = 1500;
public int bonus = 0;

if (yrsOfService < 5)
{
    if (salary < 500)
    {
        bonus = 100;
    }
    else
    {
        bonus = 200;
    }
}
else
{
    bonus = 700;
}
Console.WriteLine("Bonus amount: " + bonus);
```

# Câu lệnh if

---



# Câu lệnh switch

---

```
switch(expression)
{
  case value1:
    //statement sequence
    break;
  case value2:
    //statement sequence
    break;

  .....

  .....
  case valueN:
    //statement sequence
    break;
  default:
    //default statement sequence
}
```

```
static void Main()  
{  
    int day=5;  
    switch(day)  
    {  
        case 1:  
            Console.WriteLine("Sunday");  
            break;  
        case 2:  
            Console.WriteLine("Monday");  
            break;  
        case 3:  
            Console.WriteLine("Tuesday");  
            break;  
        case 4:  
            Console.WriteLine("Wednesday");  
            break;  
        case 5:  
            Console.WriteLine("Thursday");  
            break;  
        case 6:  
            Console.WriteLine("Friday");  
            break;  
        case 7:  
            Console.WriteLine("Saturday");  
            break;  
        default:  
            Console.WriteLine("Enter the day from 1 to 7");  
            break;  
    }  
}
```



# Vòng lặp while

---

```
while (condition)
{
    // one or more statements;
}
```

# Vòng lặp while

---

□ Liệt kê các số chẵn từ 1 đến 10

```
static void Main()
{
    int Num=1;
    Console.WriteLine("Cac so chan: ");
    while (Num<=10)
    {
        if (Num%2==0)
            Console.WriteLine(Num);
        Num++;
    }
    Console.ReadLine();
}
```

# Vòng lặp do ... while

---

```
do
{
    // one or more statements;
} while (condition);
```

# Vòng lặp do ... while

---

□ Liệt kê các số chẵn từ 1 đến 10

```
static void Main()
{
    int Num=1;
    Console.WriteLine("Cac so chan: ");
    do
    {
        if (Num%2==0)
            Console.WriteLine(Num);
        Num++;
    }
    while (Num<=10);
    Console.ReadLine();
}
```

# Vòng lặp for

---

□ Cú pháp cơ bản của vòng lặp for:

```
for (initialisation; condition; increment/decrement)
{
    // one or more statements;
}
```

# Vòng lặp for

---

□ Liệt kê các số chẵn từ 1 đến 10

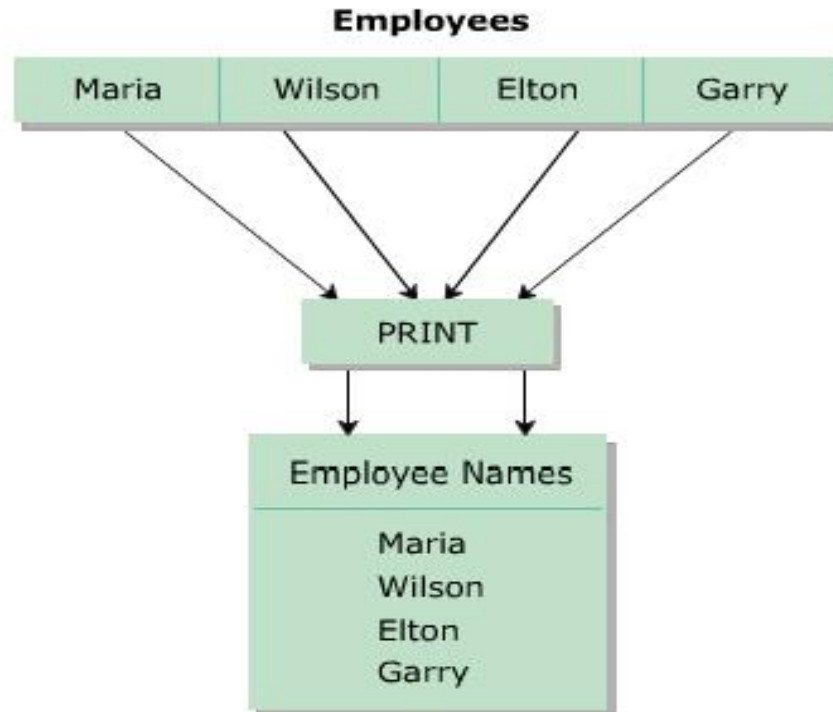
```
static void Main()
{
    Console.WriteLine("Cac so chan: ");
    for(int Num=1;Num<=10;Num++)
    {
        if(Num%2==0)
            Console.WriteLine(Num);
    }

    Console.ReadLine();
}
```

# Vòng lặp for each

---

- Cho phép đi qua 1 tập hợp các phần tử:



# Vòng lặp for each

---

□ Cú pháp:

```
foreach (<datatype> <identifier> in <list>)  
{  
    // one or more statements;  
}
```



# Vòng lặp for each

---

```
static void Main()
{
    string []emp_Names = {"Maria", "Wilson",
                          "Elton", "Garry"};

    foreach(string str in emp_Names)
        Console.WriteLine(str);
    Console.ReadLine();
}
```

# Câu lệnh break

---

- ❑ Cho phép thoát khỏi vòng lặp không cần kiểm tra điều kiện của vòng lặp

```
static void Main()
{
    int i,n;
    n=11;
    for(i=2;i<=n-1;i++)
        if (n%i==0) break;

    if(i==n) Console.WriteLine("{0} la so nguyen to", n);
    else Console.WriteLine("{0} khong la so nguyen to",n);

    Console.ReadLine();
}
```

# Câu lệnh continue

---

- Câu lệnh continue cho phép kết thúc lần lặp hiện hành và chuẩn bị bắt đầu lần lặp kế tiếp.
- **Thí dụ:** Liệt kê các số chẵn từ 1 đến 10:

```
Console.WriteLine("Cac so chan: ");  
for(int Num=1;Num<=10;Num++)  
{  
    if(Num%2!=0)  
        continue;  
    Console.WriteLine(Num);  
}  
  
Console.ReadLine();
```