Bài giảng môn học

# Bài 3: Kiến thức cơ bản về C#

**Lương Trần Hy Hiến**

**FIT, HCMUP**

*Lập trình Windows Form với C#*

# Nội dung

- Giới thiệu C#
- Những cơ sở ngôn ngữ C#
- Cấu trúc
- Lớp và đối tượng
- Thừa kế và đa hình
- Giao diện
- Array, Indexer và Collection
- Xử lý lỗi & exception (biệt lệ)

# Giới thiệu C#

- **C# là một ngôn ngữ đơn giản:**
  - Loại bỏ những phức tạp có trong Java hay C++ như macro, template, đa kế thừa, virtual base class
  - Giống về diện mạo cú pháp C và C++ nhưng được cải tiến đơn giản hơn (Ví dụ : "**::** , **.** , **→**" chỉ còn "**.**"
- **C# là một ngôn ngữ hiện đại:**
  - Có đầy đủ các tính năng: Xử lý ngoại lệ, thu gom bộ nhớ tự động, kiểu dữ liệu an toàn, bảo mật mã nguồn…

# Giới thiệu C#

- **C# là ngôn ngữ hướng đối tượng:**
  - Đóng gói (encapsulation)
  - Kế thừa (inheritance)
  - Đa hình (polymorphism)
- **C# là một ngôn ngữ mạnh mẽ và mềm dẻo:**
  - Tuỳ thuộc vào bản thân người dùng. Không có giới hạn ở bản chất ngôn ngữ.
  - Tạo các ứng dụng đồ hoạ, xử lý văn bản, trình biên dịch cho các ngôn ngữ khác v.v…

# Những cơ sở ngôn ngữ C#

- **Phân biệt chữ hoa chữ thường**
- **Có các kiểu :**
  - Dựng sẵn : **byte, char, sbyte, int, float, double…**
  - Hằng : const int PI = 3.1416;
  - Liệt kê : enum Ngay {Hai,Ba,Tu,Nam,Sau,Bay,CN};
- **Câu lệnh : if else, switch, for, while, goto**
  - **foreach**: vòng lặp để duyệt tất cả các phần tử của mảng, tập hợp

    VD : int[] intarray; intarray = new int[5];

    foreach(int i in intarray) s+= i.ToString();

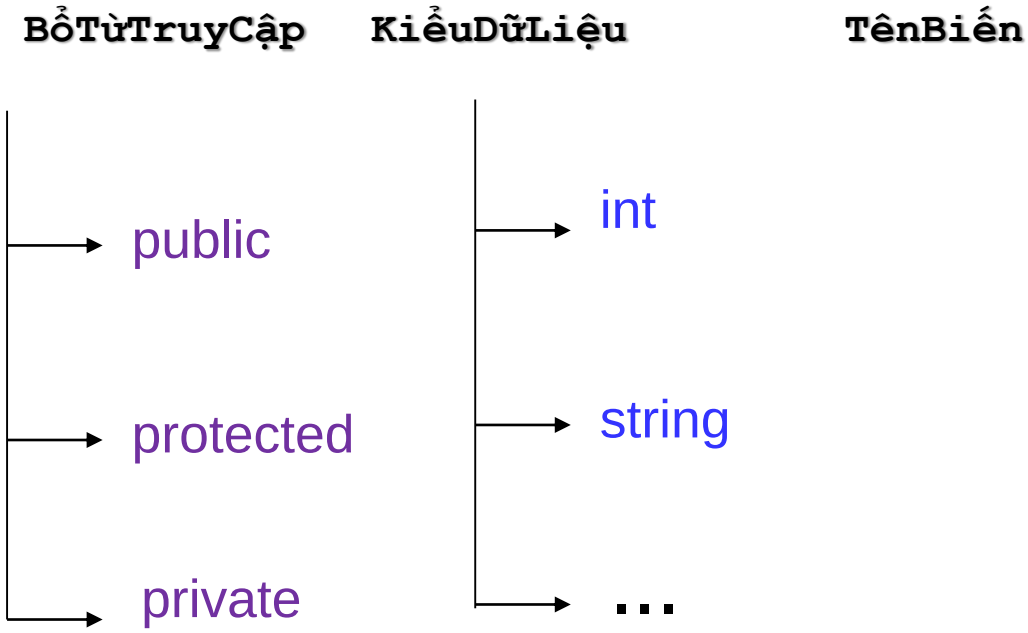
# Khai báo biến, hằng

- **Khai báo biến:**

```
int i;
i = 0;
int  x = 10; y = 20;
bool b = true;
```

- **Khai báo hằng:**

```
const int A = 20;
```



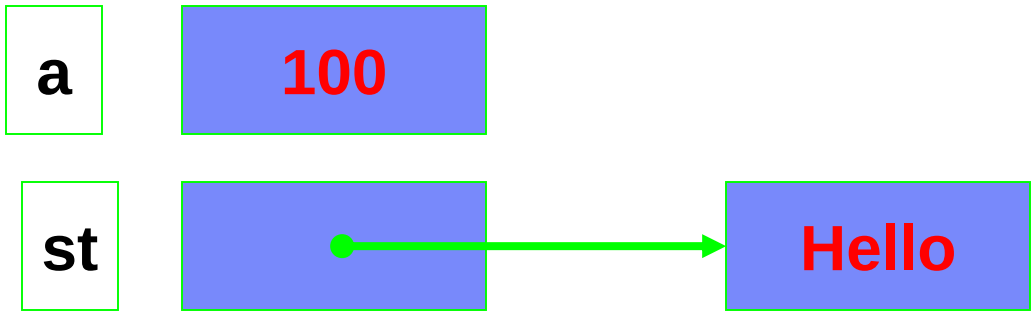
# Các kiểu dữ liệu cơ sở

- **C# chia các kiểu dữ liệu thành 2 loại:**
- **Kiểu giá trị (Value Type): int, char, structures,...**
  - Giữ giá trị trong bộ nhớ
  - Được lưu trong stack
- **Kiểu tham chiếu (Reference Type): classes, interfaces, arrays, string**
  - Chứa địa chỉ của đối tượng trong bộ nhớ heap
  - Có giá trị **null** khi không tham chiếu đến đối tượng nào

# VD: Các kiểu dữ liệu cơ sở

```
int  a = 100;

string st = "Hello";
```



# Value type – số nguyên

| Name | CTS Type | Description | Range (min:max) |
|---|---|---|---|
| byte | System.Byte | 8-bit signed integer | 0:255 (0:$2^8$-1) |
| ushort | System.UInt16 | 16-bit signed integer | 0:65,535 (0:$2^{16}$-1) |
| uint | System.UInt32 | 32-bit signed integer | 0:4,294,967,295 (0:$2^{32}$-1) |
| ulong | System.UInt64 | 64-bit signed integer | 0:18,446,744,073, 709,551,615(0: $2^{64}$-1) |

# Value type – số nguyên

| Name | CTS Type | Description | Range (min:max) |
|---|---|---|---|
| sbyte | System.SByte | 8-bit signed integer | -128:127 ($-2^7$:$2^7$-1) |
| short | System.Int16 | 16-bit signed integer | -32,768:32,767 ($-2^{15}$:$2^{15}$-1) |
| int | System.Int32 | 32-bit signed integer | -2,147,483,648:2,147,483,647 ($-2^{31}$:$2^{31}$-1) |
| long | System.Int64 | 64-bit signed integer | -9,223,372,036,854,775,808: 9,223,372,036,854,775,807 ($-2^{63}$:$2^{63}$-1) |

# Value type – số thực

| Name | CTS Type | Description | Significant Figures | Range (approximate) |
|---|---|---|---|---|
| Float | System.Single | 32-bit single-precision floating- point | 7 | $\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$ |
| Double | System.Double | 64-bit double-precision floating- point | 15/16 | $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ |
| decimal | System.Decimal | 128-bit high precision decimal notation | 28 | ±1.0 × 10-28 to ±7.9 × 1028 |

# Value type - Kiểu Boolean & char

| Name | CTS Type | Value |
|---|---|---|
| Bool | System.Boolean | true or false |
| char | System.Char | Represents a single 16-bit (Unicode) character |

# Các ký tự escape thông dụng

| Escape Sequence | Character |
|---|---|
| \' | Single quote |
| \" | Double quote |
| \\ | Backslash |
| \0 | Null |
| \a | Alert |
| \b | Backspace |
| \f | Form feed |
| \n | New line |
| \r | Carriage return |
| \t | Tab character |
| \v | Vertical tab |

# Reference Type

- **Lớp đối tượng Object (System.Object): là lớp trừu tượng, là lớp cha của tất cả các lớp. Khi định nghĩa một lớp A. Mặc nhiên A sẽ lấy Object làm lớp cha.**
- **Reference Type:**
  - Kiểu lớp: Object, String, CHocSinh, CLopHoc, …

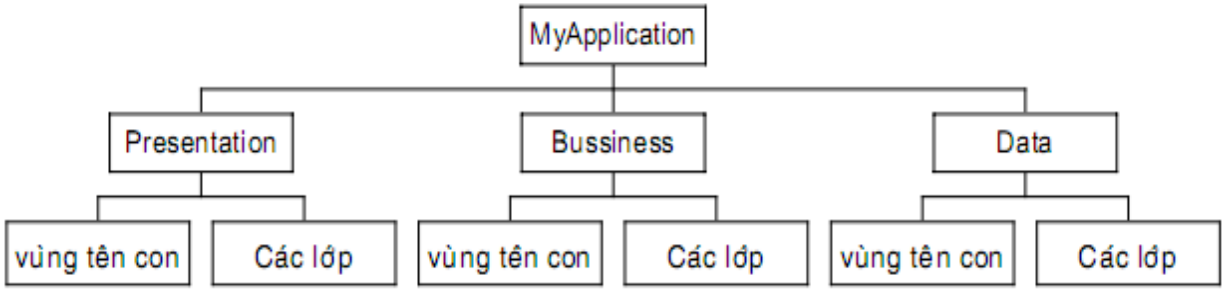
# Namespace

- **Namespace cung cấp cho cách tổ chức quan hệ giữa các lớp và các kiểu khác.**
- **Sử dụng namespace hợp lý thì các class dễ sử dụng và tránh xung đột với các class được viết bởi người khác**

```
namespace CustomerPhoneBook
{
   using System;
   public struct Subscriber
   { // Code for struct here... }
}
```

# Những cơ sở ngôn ngữ C#

- **Tạo vùng tên (namespace)**




**Cách 1**

```
namespace MyApplication
{
  namespace Presentation
  {
    // khai báo lớp
    // khai báo vùng tên con
  }
  namespace Bussiness
  {
    // khai báo lớp
    // khai báo vùng tên con
  }
  namespace Data
  {
    // khai báo lớp
    // khai báo vùng tên con
  }
}
```

**Cách 2**

```
namespace MyApplication.Presentation
{
  // khai báo lớp
  // khai báo vùng tên con
}
namespace MyApplication.Bussiness
{
  // khai báo lớp
  // khai báo vùng tên con
}
namespace MyApplication.Data
{
  // khai báo lớp
  // khai báo vùng tên con
}
```

# Xuất dữ liệu

- **Nhập dữ liệu từ bàn phím và xuất dữ liệu ra màn hình trong C# có thể dùng các phương thức tĩnh trong lớp: System.Console**
- Xuất dữ liệu lên màn hình
  - Cú pháp 1:

```
void Console.Write(data);
void Console.WriteLine(data);
```

# Xuất dữ liệu

- Cú pháp 2:

```
void Console.Write(string format, params object[] arg);
void Console.WriteLine(string format, params object[] arg);
```

- Trong đó:
  - format: chứa chuỗi định dạng
  - arg là mảng các đối tượng sẽ được xuất ra theo chuỗi định dạng

# Xuất dữ liệu

- format là một chuỗi bình thường và có thể có thêm một hay nhiều phần định dạng có cú pháp sau
  - Cú pháp:

```
{index[,alignment][:formatString]}
```

  - Trong đó:
    - index: Số thứ tự của đối số, bắt đầu từ 0
    - alignment: độ rộng, M>0 canh phải, M<0 canh trái
    - formatString: C hay c, D hay d, E hay e, F hay f…

# Nhập dữ liệu

- **Nhập dữ liệu từ bàn phím**
  - Cú pháp:

```
int Console.Read();
string Console.ReadLine();
```

# Nhập dữ liệu – Chuyển kiểu dữ liệu

- **Để chuyển một kiểu dữ liệu sang một kiểu dữ liệu khác chúng ta dùng cú pháp sau**
- **Cú pháp**

```
Kieu.Parse("chuoi");
```

- Ví dụ:

```
string s = "123";
int data = int.Parse(s);
```

# Nhập dữ liệu – Lớp Convert

- **Đây là lớp tiện ích cung cấp các phương thức static giúp chuyển đổi giữa các dữ liệu có các kiểu khác nhau**

| Phương thức | Ý nghĩa |
|---|---|
| ToBoolean | Chuyển một giá trị sang giá trị Boolean |
| ToByte | Chuyển một giá trị sang giá trị số nguyên 8-bit không dấu |
| ToChar | Chuyển một giá trị sang giá trị ký tự unicode |
| ToDateTime | Chuyển một giá trị sang giá trị DateTime. |
| ToDecimal | Chuyển một giá trị sang giá trị Decimal. |
| ToDouble | Chuyển một giá trị sang giá trị số thực có độ chính xác gấp đôi 8 byte |
| ToInt16 | Chuyển một giá trị sang giá trị số nguyên 16-bit có dấu |
| ToInt32 | Chuyển một giá trị sang giá trị só nguyên 32-bit có dấu |
| ToInt64 | Chuyển một giá trị sang giá trị số nguyên 64-bit có dấu |
| ToSByte | Chuyển một giá trị sang giá trị số nguyên 8-bit có dấu |
| ToSingle | Chuyển một giá trị sang giá trị số thực có độ chính xác đơn |
| ToString | Chuyển một giá trị sang giá trị một chuỗi |
| ToUInt16 | Chuyển một giá trị sang giá trị số nguyên 16-bit không dấu |
| ToUInt32 | Chuyển một giá trị sang giá trị số nguyên 32-bit không dấu |
| ToUInt64 | Chuyển một giá trị sang giá trị số nguyên 64-bit không dấu |

# Chuyển đổi kiểu

- **Chuyển đổi kiểu chuỗi sang các kiểu dữ liệu khác**

<Kiểu dữ liệu>.Parse(chuỗi)

Ví dụ:

```
string s;
s ="123.45";
float f = Single.Parse( s);
double d = Double.Parse(s2);
short i = Int16.Parse(s);
int j = Int32.Parse(s);
long k = Int64.Parse(s);
```

# Chuyển đổi kiểu

- **Chuyển đổi kiểu dữ liệu số sang kiểu chuỗi**

  Nguyên tắc

  <tên biến>.ToString( ) ;

  Ví dụ

```
int i = 231 ;
float j = 34.56f  ;  //ngầm định là double !!!
String kq;
kq = " i= "+ i.ToString() + " va j = "+ j.ToString();
```

# Định danh – Identity

- **Định danh – Identity: Tên lớp, tên phương thức, tên biến, tên đối tượng, tên hằng, tên kiểu, …**

- **Quy tắc tạo định danh trong C#:**
  - Ký tự đầu tiên: chữ, ký tự gạch dưới, ký tự @
  - Các ký tự còn lại: chữ, số, ký tự gạch dưới
  - Có thể dùng @ ở đầu từ khóa để tạo định danh

# Định danh – Identity
# Quy tắc đặt tên định danh

- **Một định danh có thể được đặt theo hai quy tắc**
  - Pascal: Ký tự đầu tiên của mỗi từ viết HOA
  - Camel: Giống Pascal nhưng ký đầu tiên của định danh được viết thường

- **Quy tắc đặt tên định danh**
  - Pascal
    - NameSpace
    - Class
    - Interface
    - Public Method/Field
    - Constant
  - Camel
    - Others

# Định danh – Identity
# Quy tắc đặt tên định danh

- **Namespace**
  - Pascal
  - Tên của công ty + tên đề án cách nhau bằng ký tự dấu chấm “.”

# Định danh – Identity
# Quy tắc đặt tên định danh

- **Interface**
  - Pascal
  - Bắt đầu bằng ký tự I

- **Lớp**
  - Pascal
  - Danh từ
  - Ví dụ:
    - Line, AudioSystem

# Định danh – Identity
# Quy tắc đặt tên định danh

- **Field**
  - Camel
  - Không chứa tên lớp
  - Ví dụ:
    - line, audioSystem
  - Những biến có vai trò thì kết hợp vai trò với kiểu
  - Ví dụ:
    - Point startingPoint;
    - Name loginName;

# Định danh – Identity
# Quy tắc đặt tên định danh

- **Field**
  - Những biến chung chung nên có cùng tên với kiểu
  - Ví dụ:
    - Topic topic;
    - Database database;
  - Những biến kiểu control: list, text, …: Nên kèm theo kiểu ở suffix
  - Ví dụ:
    - leftScrollbar

# Định danh – Identity
# Quy tắc đặt tên định danh

- **Field**
  - Mảng hay collection: Dùng danh từ số nhiều
  - Ví dụ:
    - Point[] points;
  - Biến đếm: Dùng prefix: n, cnt
  - Ví dụ:
    - nPoints;
    - cntPoint;

# Định danh – Identity
# Quy tắc đặt tên định danh

- **Phương thức**
  - Pascal
  - Động từ
  - Không chứa tên lớp
  - Khi phương thức truy cập field
    - Get/Set + field
    - Dùng property
  - Ví dụ:
    - employee.getName();

# Định danh – Identity
# Quy tắc đặt tên định danh

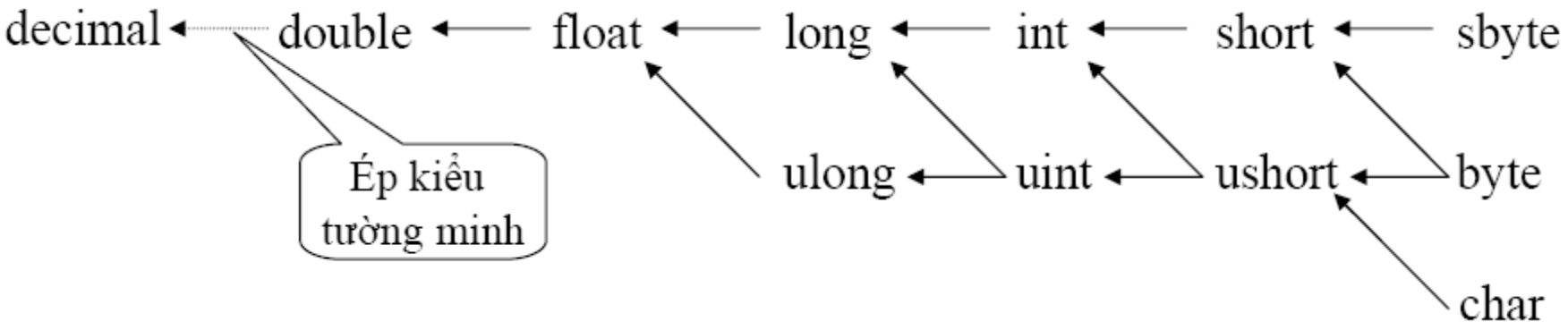
- **Phương thức**
  - Khi phương thức trả về kiểu bool: Dùng prefix: Is, has, can, should
  - Ví dụ:
    - bool IsOpen();
    - bool ShouldAbort();
  - Khi phương thức
    - Tính toán: Dùng prefix: Compute
    - Tìm kiếm: Dùng prefix: Find
    - Thiết lập: Dùng prefix: Iniitialize

# Định danh – Identity
# Quy tắc đặt tên định danh

- **Đối số của phương thức**
  - Pascal
- **Biến lặp trong phương thức**
  - i, j, k, …
- **Hằng giá trị**
  - Viết HOA tất cả ký tự và có thể dùng thêm ký tự Underscore
- **Từ viết tắt**
  - Chỉ viết HOA ký tự đầu
  - Ví dụ:
  - OpenDvdPlayer()

# Chuyển kiểu ngầm định

- **Không có chuyển kiểu ngầm định**
  - Sang kiểu char
  - Giữa số thực và decimal



# Boxing và Unboxing

- **Vấn đề: Việc tách thành hai loại kiểu (kiểu giá trị và kiểu tham chiếu) thì làm thế nào các kiểu tương tác với nhau?**

- **Giải pháp: Dùng 2 kỹ thuật sau đây**
  - Boxing: là chuyển từ kiểu giá trị sang kiểu tham chiếu
  - Unboxing: là chuyển từ kiểu tham chiếu sang kiểu giá trị

# Boxing và Unboxing

- **Boxing**
  - Ví dụ:
    - int a = 55;
    - object o = a;
  - Quá trình hoạt động của boxing
    - Trước hết một vùng nhớ được cấp phát trên vùng nhớ heap để tạo đối tượng o
    - Sau đó giá trị của biến kiểu giá trị được sao chép sang vùng nhớ heap đó
    - Cuối cùng địa chỉ của đối tượng được cấp phát trên heap được đặt vào vùng nhớ trên stack

# Boxing và Unboxing

- **Unboxing**
  - Ví dụ:
    - `int a = 55;`
    - `object o = a;`
    - `int b = (int)o;`

- **Chú ý:**
  - Boxing là không cần ép kiểu
  - Unboxing phải ép kiểu

  - Quá trình hoạt động của boxing
    - Trước hết runtime kiểm tra xem địa chỉ trên stack có trỏ đến đối tượng hợp lệ không và kiểm tra xem kiểu đối tượng có thể được chuyển sang kiểu giá trị không. Nếu không sẽ nén ra một ngoại lệ InvalidCastException
    - Một con trỏ đến giá trị bên trong đối tượng được trả về. Chú ý rằng boxing tạo một bản sao của kiểu được chuyển đổi, còn unboxing thì không làm thế.

# Câu lệnh điều kiện

- **Câu lệnh điều kiện:**
  - if: giống C/C++
  - switch: giống C/C++

# Câu lệnh if, switch

- **Cú pháp 1**

```
if (expression)
{
        Các câu lệnh
}
```

- **Cú pháp 2**

```
if (expression)
{
        Các câu lệnh A;
}
else
{
        Các câu lệnh B;
}
```

- Quy tắc của câu lệnh if: Giá trị của biểu thức phải là một giá trị kiểu bool

# Câu lệnh if, switch

- **Câu lệnh switch**
  - Cú pháp

```
switch (expression)
{
        case const_expression1:
                …
                break;
        case const_expression2:
                …
                break;
        …
        case const_expressionN:
                …
                break;
        default:
                …
                break;
}
```

# Câu lệnh if, switch

- **Quy tắc của câu lệnh switch:**
  - expression phải thuộc một trong các kiểu:
    - số nguyên, char, string, enum
  - Mỗi case (kể cả default) luôn cung cấp "lệnh nhảy" (jump statement) (break, return, goto)
  - Nếu thân case là câu lệnh rỗng thì không cần "lệnh nhảy"
  - Thứ tự các case, default không quan trọng

# Câu lệnh nhảy – jump statement

- **Câu lệnh break, continue, return giống như trong C/C++**
- **Câu lệnh goto**
  - Cú pháp

```
goto label;
goto case constExpression
goto default;
```

- Quy tắc của câu lệnh nhảy: được nhảy ra, không được nhảy vào

# Vòng lặp

- **Vòng lặp:**
  - do… while: giống C/C++
  - while: giống C/C++
  - for: giống C/C++
  - foreach: khác C/C++

# Câu lệnh lặp

- **Câu lệnh for**
  - Cú pháp

```
for (initialization; BooleanExpression; step)
{
        Các câu lệnh
}
```

- Câu lệnh while
  - Cú pháp

```
while (BooleanExpression)
{
        Các câu lệnh
}
```

# Câu lệnh lặp

- **Câu lệnh do...while**
  - Cú pháp

```
do
{
        Các câu lệnh

} while (BooleanExpression);
```

# Câu lệnh lặp

- **Câu lệnh foreach**
  - Cú pháp

```
foreach (type variableName in expression)
{
        Các câu lệnh
}
```

  - Trong đó:
    - type là kiểu của biến variableName
    - expression là đối tượng thuộc collection hay mảng

# Các cấu trúc điều khiển

## Cấu trúc lặp – while

- **Cú pháp**

```
while (<Điều kiện lặp>)
{
        <Tập lệnh>
        [continue;]
        [break;]
}
```

# Các cấu trúc điều khiển

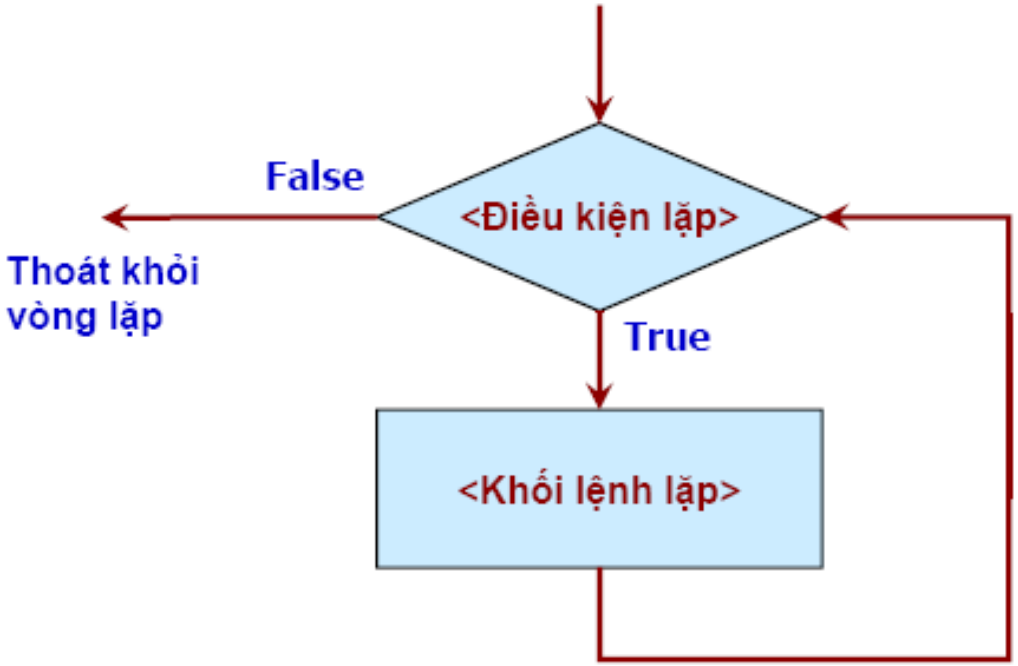
## Cấu trúc lặp – while

- **Ý nghĩa**
  - Điều kiện lặp: là biểu thức logic, trả về true/false
  - Tập lệnh: chỉ có thể được thực hiện và lặp nếu và chỉ nếu <Điều kiện lặp> = true
  - break: thoát khỏi vòng lặp
  - continue: quay trở lên kiểm tra điều kiện của vòng lặp khi cần

# Các cấu trúc điều khiển

## Cấu trúc lặp – while

- Sơ đồ hoạt động:



# Các cấu trúc điều khiển

## Cấu trúc lặp – while

- **Ví dụ 1: Tính tổng các số nguyên từ 1 đến 100**

```
int i=1, tong=0;
while (i <= 100)
{
   tong += i; // ⇔ tong =tong + i
     //Tăng i → tác động đến điều kiện lặp
     i ++;
}
```

# Các cấu trúc điều khiển

## Cấu trúc lặp – for

- Cú pháp

```
for (<biến đếm> = <gt đầu>; <điều kiện lặp>; <tăng/giảm biến đếm>)
{
        <Khối lệnh lặp>
        [continue;]
        [break;]
}
```

# Các cấu trúc điều khiển

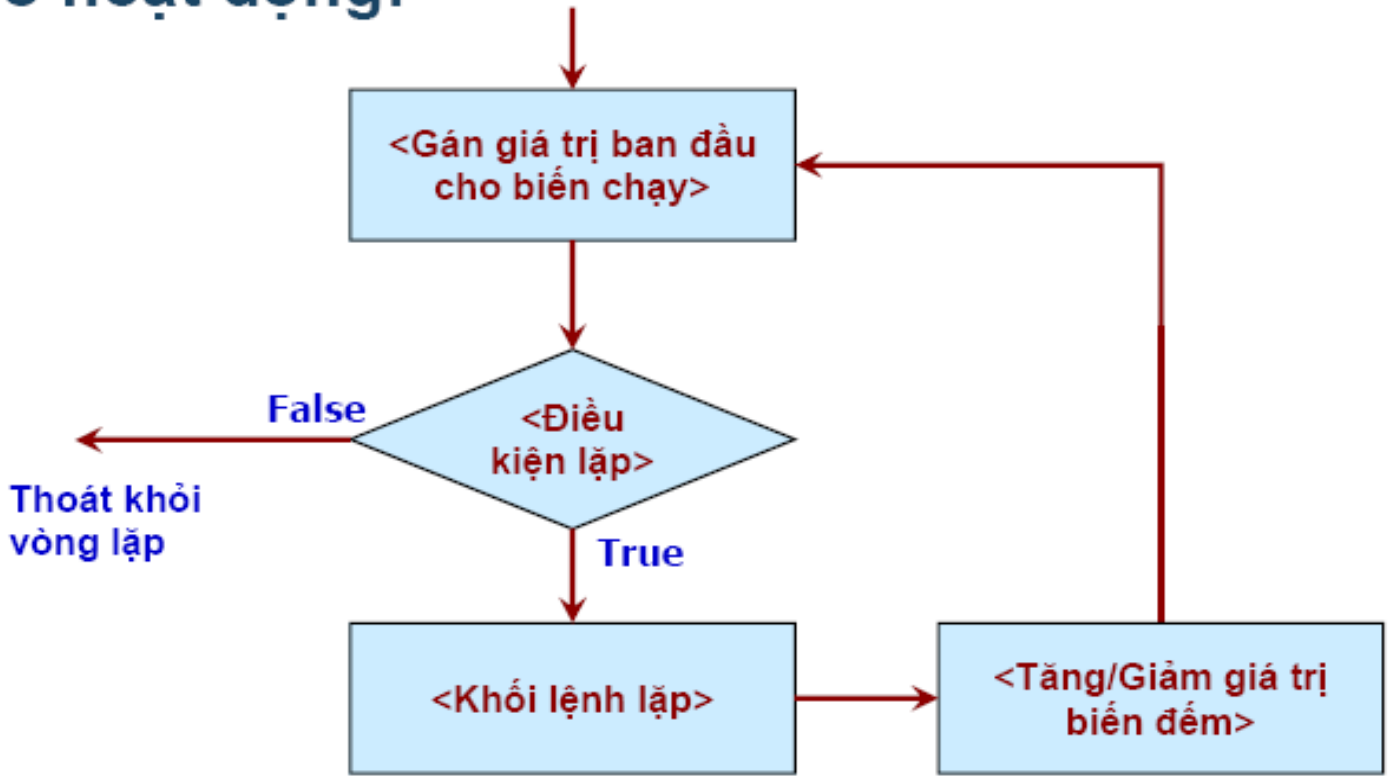
## Cấu trúc lặp – for

- **Ý nghĩa**
  - **Khối lệnh lặp:** chỉ được thực hiện nếu <biến đếm> còn thỏa điều kiện lặp
  - **break:** thoát khỏi vòng lặp khi cần
  - **continue:** bỏ qua các lệnh còn lại (nếu có) và tiếp tục lặp

# Các cấu trúc điều khiển

## Cấu trúc lặp – for

- Sơ đồ hoạt động:



# Các cấu trúc điều khiển

## Cấu trúc lặp – for

- **Ví dụ 1: Tính tổng các số nguyên từ 1 đến 100**

```
int tong =0;
int i;
for(i = 1; i<=100; i++)
{
        tong += i; ⇔ tong =tong + i
}
```

# Các cấu trúc điều khiển

## Cấu trúc lặp – for

- **Ví dụ 2: Tính tổng các số nguyên lẻ từ 1 đến 100**

```
int i, tong;
tong =0;
for(i=1; i<=100; i=i+2)
{
    tong += i;
}
```

Hoặc

```
for(i=1; i<=100; i++)
{
    if(i%2 != 0)
        tong += i;
}
```

# Các toán tử

| Category | Operator |
|---|---|
| Arithmetic | + - * / % |
| Logical | & \| ^ ~ && \|\| ! |
| String concatenation | + |
| Increment and decrement | ++ -- |
| Bit shifting | << >> |
| Comparison | == != < > <= >= |
| Assignment | = += -= *= /= %= &= \|= ^= <<= >>= |

# Các toán tử (tt)

| Category | Operator |
|---|---|
| Member access (for objects and structs) | . |
| Indexing (for arrays and indexers) | [] |
| Cast | () |
| Conditional (the Ternary Operator) | ?: |
| Object Creation | new |
| Type information | sizeof (unsafe code only) is typeof as |
| Overflow exception control | checked unchecked |
| Indirection and Address | * -> & (unsafe code only) [] |

# Các toán tử (tt)

| Shortcut Operator | Tương đương |
|---|---|
| x++, ++x | x = x + 1 |
| x--, --x | x = x - 1 |
| x += y | x = x + y |
| x -= y | x = x – y |
| x *= y | x = x * y |
| x /= y | x = x / y |

# Các toán tử (tt)

| Shortcut Operator | Tương đương |
|---|---|
| x %= y | x = x % y |
| x >>= y | x = x >> y |
| x <<= y | x = x << y |
| x &= y | x = x & y |
| x \|= y | x = x \| y |
| x ^= y | x = x ^ y |

# Phương thức - Method

*Lập trình Windows Form với C#*

# Phương thức

- Khai báo phương thức (hàm)
- Truyền tham số dạng in (ø)
- Truyền tham số dạng out
- Truyền tham số dạng ref

# Khai báo phương thức

```
[modifiers] return_type MethodName([parameters])
{
   // Thân phương thức/hàm
}
```

**Ví dụ:**

```
public static void Xuat (HocSinh hs)
{
   Console.Write("Ma so: {0}. Ho ten: {1}", hs.MaSo,
   hs.HoTen);
   //Cau lenh xuat hoc sinh
}
```

# Gọi phương thức

- [this.]ten_phuong_thuc([danh_sach_tham_so])
- Ví dụ 1:

```
int binhPhuong(int x){
        return x * x;
}
int kq = binhPhuong(7); //Gọi
```

- Ví dụ 2:

```
void Xuat(string s){ Console.WriteLine(s); }
//Lời gọi phải nằm trên 1 dòng lệnh riêng biệt
Xuat("Xin chao cac ban");
```

# Phương thức dạng "in"

- Thân phương thức chỉ tham khảo giá trị của tham số không thay đổi giá trị của tham số
- Ví dụ:

```
public static void Xuat(HocSinh hs)
{
   Console.Write("Ma so: {0}. Ho ten: {1}", hs.MaSo,
   hs.HoTen);
   //Cau lenh xuat hoc sinh
}
```

- Gọi hàm trong hàm Main:

```
Xuat(hs);
```

# Phương thức dạng "out"

- Thân phương thức cấp phát (khởi tạo) giá trị của tham số trước khi sử dụng. Ra khỏi hàm giá trị của tham số thay đổi.
- Ví dụ:

```
public static void Nhap(out HocSinh hs)
{
        hs = new StrHocSinh();
        //Cau lenh nhap hoc sinh
}
```

- Gọi trong hàm Main:

```
Nhap(out hs);
```

# Phương thức dạng "ref"

- Ra khỏi hàm giá trị của tham số sẽ thay đổi
- Ví dụ:

```
public static void TinhDiemTrungBinh(ref HocSinh hs)
{
        hs.DTB = (hs.Toan+ hs.Van)/2;
}
```

- Gọi trong hàm Main:

```
TinhDiemTrungBinh(ref hs);
```

# Phương thức tùy chọn

- **Tham số dạng option,**
  **<kiểu> <tham_số> = <giá_trị>**
- **Ví dụ:**

**Định nghĩa:**

```
void Tinh(int a = 3, int b = 1, int c = 0){
        Console.WriteLine( a + b * c );
}
```

**Gọi hàm:**

```
Tinh(4,2,1);//6
Tinh(4,2);//4
Tinh(4);//4
Tinh();//3
```

# Xử lý ngoại lệ



- Sử dụng try-catch để xử lý ngoại lệ

```
try{
        //viết mã có khả năng ném ngoại lệ
}catch{
        //viết mã xử lý ngoại lệ
}
```

# Chú thích

- Chú thích (comment) được dùng để giải thích về chương trình và các câu lệnh
- Giúp cho chương trình dễ hiểu hơn
- Được bỏ qua khi biên dịch
- Không ảnh hưởng tới kết quả thực thi của chương trình
- Có thể phát sinh ra documentation của chương trình qua chú thích XML

# Hai cách tạo chú thích cơ bản

- Gõ phần chú thích sau cặp ký tự //
- Gõ phần chú thích giữa cặp ký tự /* và */

```
/* Chương trình C# đầu tiên : In ra câu chào "Hello World" */

using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Hello World!"); // Xuất ra câu chào
            Console.ReadLine(); // Chờ nhấn Enter
        }
    }
}
```

# XML Comment

- **Cho phép phát sinh ra sưu liệu dạng XML**
- **Thích hợp cho việc viết sưu liệu của dự án lớn**
- **Chú thích XML bắt đầu với triple slash (“///”) và các tag của XML**
- **Chú thích XML dùng cho**
  - User defined types
  - Class, delegate, enum and struct
  - Member of user defined types

# XML Comment

C# Code without XML Comment

```
using System;
namespace XMLCommentDemo
{
  public class Temperature
  {
    public static int CelsiusToFahrenheit(int degreesCelsius)
    {
      return ((int)((9/5)*degreesCelsius) + 32);
    }

    public static int FahrenheitToCelsius(int degressFahrenheit)
    {
      return ((int)((5/9)*(degressFahrenheit - 32)));
    }
  }
}
```

# XML Comment

- **Comment for class**

```
/// <summary>
/// Class temperature provides functions which convert
/// among various temperature scales.
/// </summary>
public class Temperature
```

- **Comment for method**

```
/// <summary>
/// Converts degrees Celsius to degrees Fahrenheit
/// </summary>
/// <param name="degreesCelsius">Degrees Celsius</param>
/// <returns>Returns degrees Fahrenheit</returns>
public static int CelsiusToFahrenheit(int degreesCelsius)
```

# XML Comment

C# Code with XML Comment

```
using System;
namespace XMLCommentDemo{
  /// <summary>
  /// Class temperature provides functions which convert among various
  /// temperature scales.
  /// </summary>
  public class Temperature {
    /// <summary>
    /// Converts degrees Celsius to degrees Fahrenheit
    /// </summary>
    /// <param name="degreesCelsius">Degrees Celsius</param>
    /// <returns>Returns degrees Fahrenheit</returns>
    public static int CelsiusToFahrenheit(int degreesCelsius) {
      return ((int)((9/5)*degreesCelsius) + 32);
    }
    /// <summary>
    /// Converts degrees Fahrenheit to degrees Celsius
    /// </summary>
    /// <param name="degressFahrenheit">Degrees Fahrenheit</param>
    /// <returns>Returns degrees Celsius</returns>
    public static int FahrenheitToCelsius(int degressFahrenheit) {
      return ((int)((5/9)*(degressFahrenheit - 32)));
    }
  }
}
```

# Một số phương thức toán học

- Math.Abs(biểu thức số)
- Math.Sqrt(biểu thức số)
- Math.Ceiling(biểu thức số)
- Math.Floor(biểu thức số)
- Math.Max(biểu thức số)
- Math.Min(biểu thức số)
- Math.Round(biểu thức số)
- hằng số Math.PI và Math.E

# Một số phương thức của kiểu chuỗi - String

**<biến chuỗi>.ToLower( );**
**<biến chuỗi>.ToUpper( );**
**<biến chuỗi>.Substring(vị trí, số ký tự);**
**<biến chuỗi>.Length ;  //không có ( và )**
**<biến chuỗi>[ vị trí ]**

- **Ví dụ**

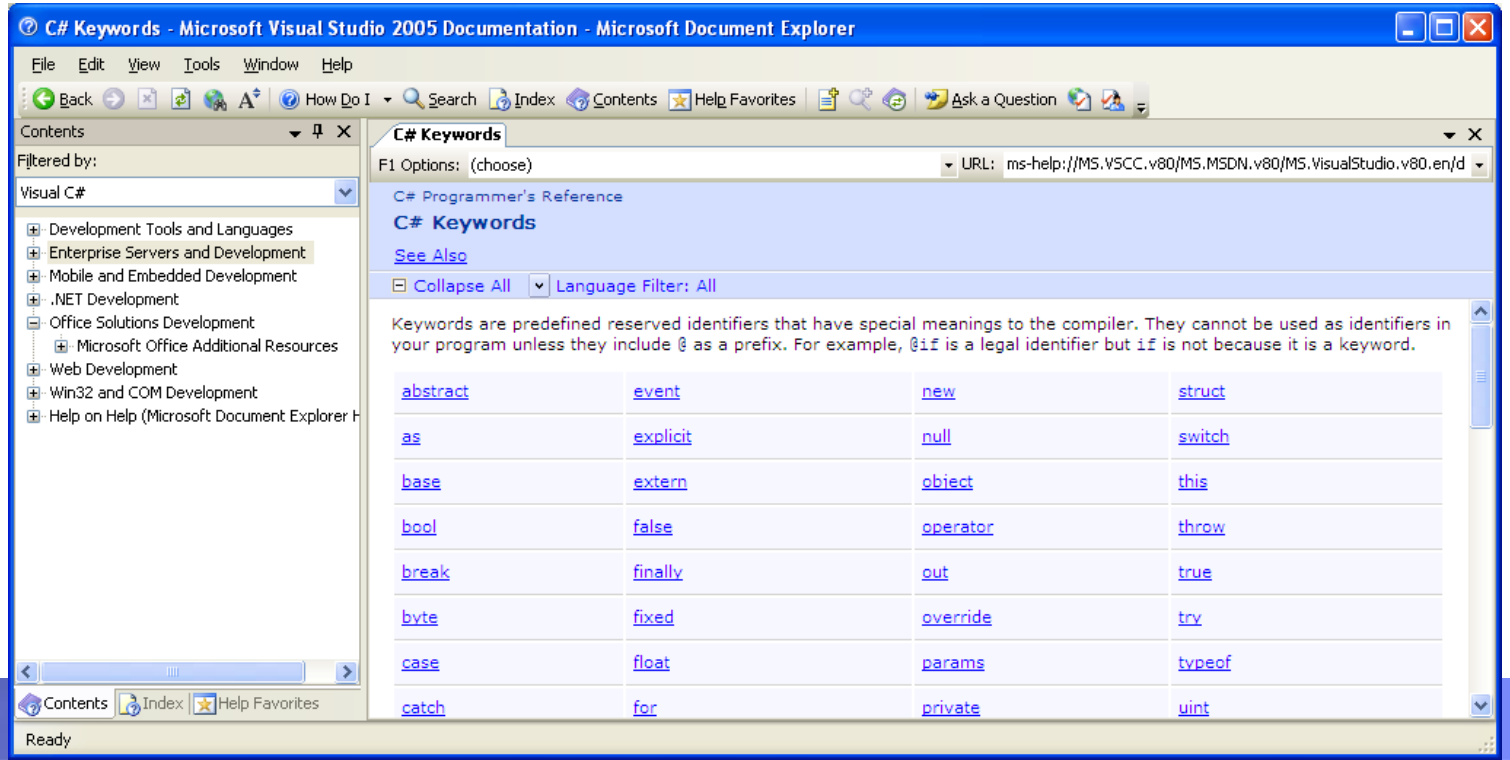
```
string S = “hello woRld”;
string u = S.ToUpper();
char c = S[1]; // c = ‘e’
int    l = S.Substring(0,4).Length ;
//thay vì ghi  (S.Substring(0,4)).Length
```

# Một số phương thức của kiểu chuỗi - String

- **IndexOf(), IndexOfAny(), LastIndexOf(), LastIndexOfAny(): tìm kiếm chuỗi ký tự, hoặc một phần chuỗi ký tự trong một xâu cho trước.**
- **Replace(): thay thế một mẫu trong xâu bởi một chuỗi ký tự khác.**
- **Split(): cắt một xâu thành các xâu con dựa theo ký tự phân cách cho trước.**
- **Trim(), TrimEnd(), TrimStart(): xoá các ký tự trắng ở đầu, cuối xâu.**
- **Insert(), Remove(): chèn vào, xoá đi một xâu con trong một xâu cho trước.**
- **StartsWith(), EndsWith(): kiểm tra xem xâu có bắt đầu, kết thúc bởi một xâu khác.**

# Thao khảo và sử dụng thêm

- **Lớp System.Int32, System.Single, System.String, System.Character, System.Boolean**




STOP