



CHỦ ĐỀ 2



MICROSOFT .NET và C#

GVGD: PHẠM THỊ KIM NGOAN

Email: ptkngoan@gmail.com

Nội dung

Tổng quan về .Net, Visual Studio

Các loại ứng dụng dùng .Net Framework

Ngôn ngữ lập trình C#

Xử lý ngoại lệ

1. Tổng quan về .Net, Visual Studio

➤ .Net

- Cung cấp một môi trường **hướng đối tượng** nhất quán cho **nhiều loại ứng dụng**.
- Cung cấp một nền tảng phát triển chung cho **nhiều ngôn ngữ** lập trình khác nhau của Microsoft: C#, Visual J#, Visual Basic...

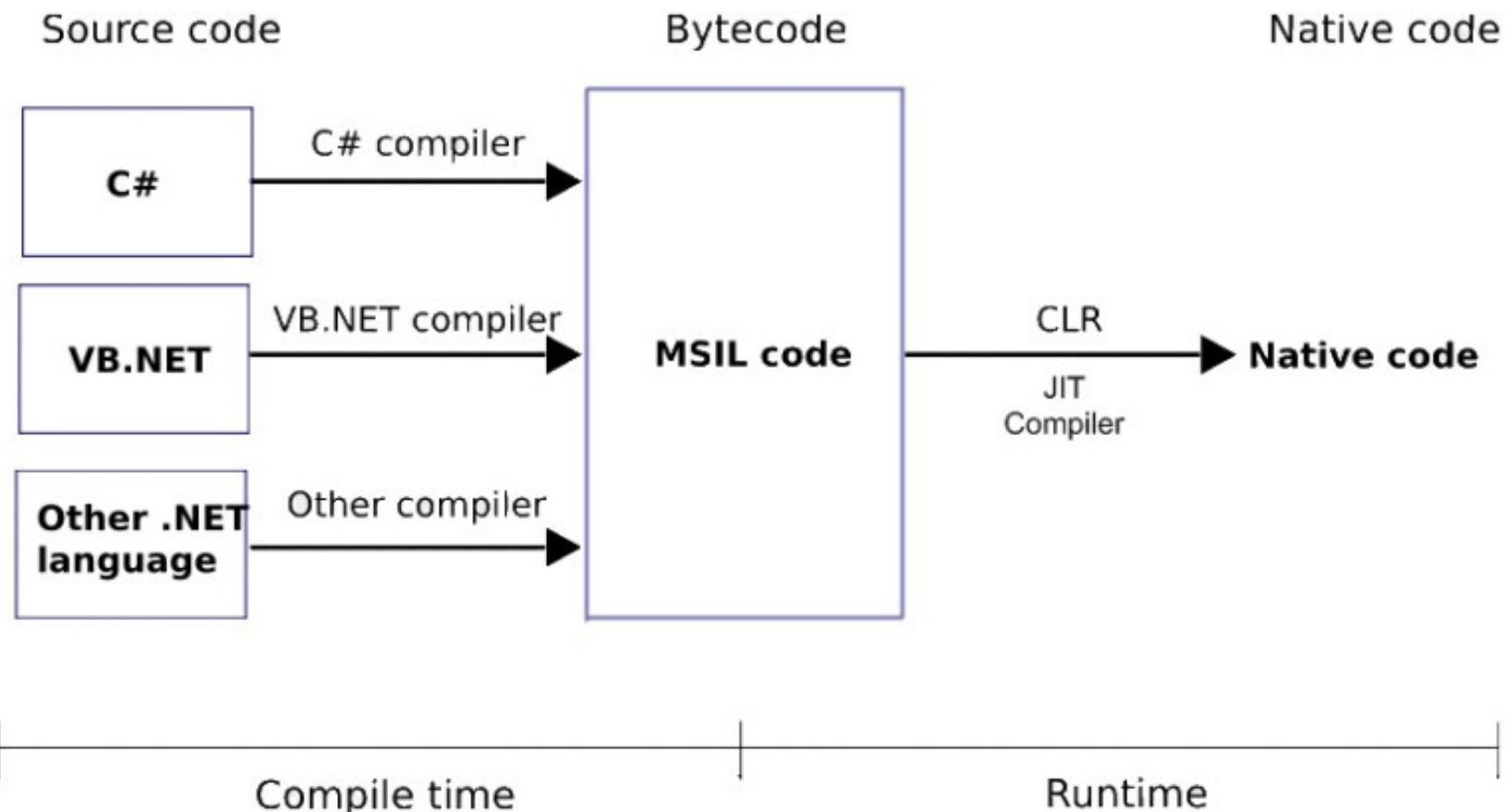
➤ .Net gồm 3 thành phần:

- **Runtime** (môi trường hoạt động)
- **Libraries** (thư viện)
- **Toolings** (công cụ phát triển)



1. Tổng quan về .Net, Visual Studio

➤ Môi trường hoạt động



1. Tổng quan về .Net, Visual Studio

➤ Thư viện

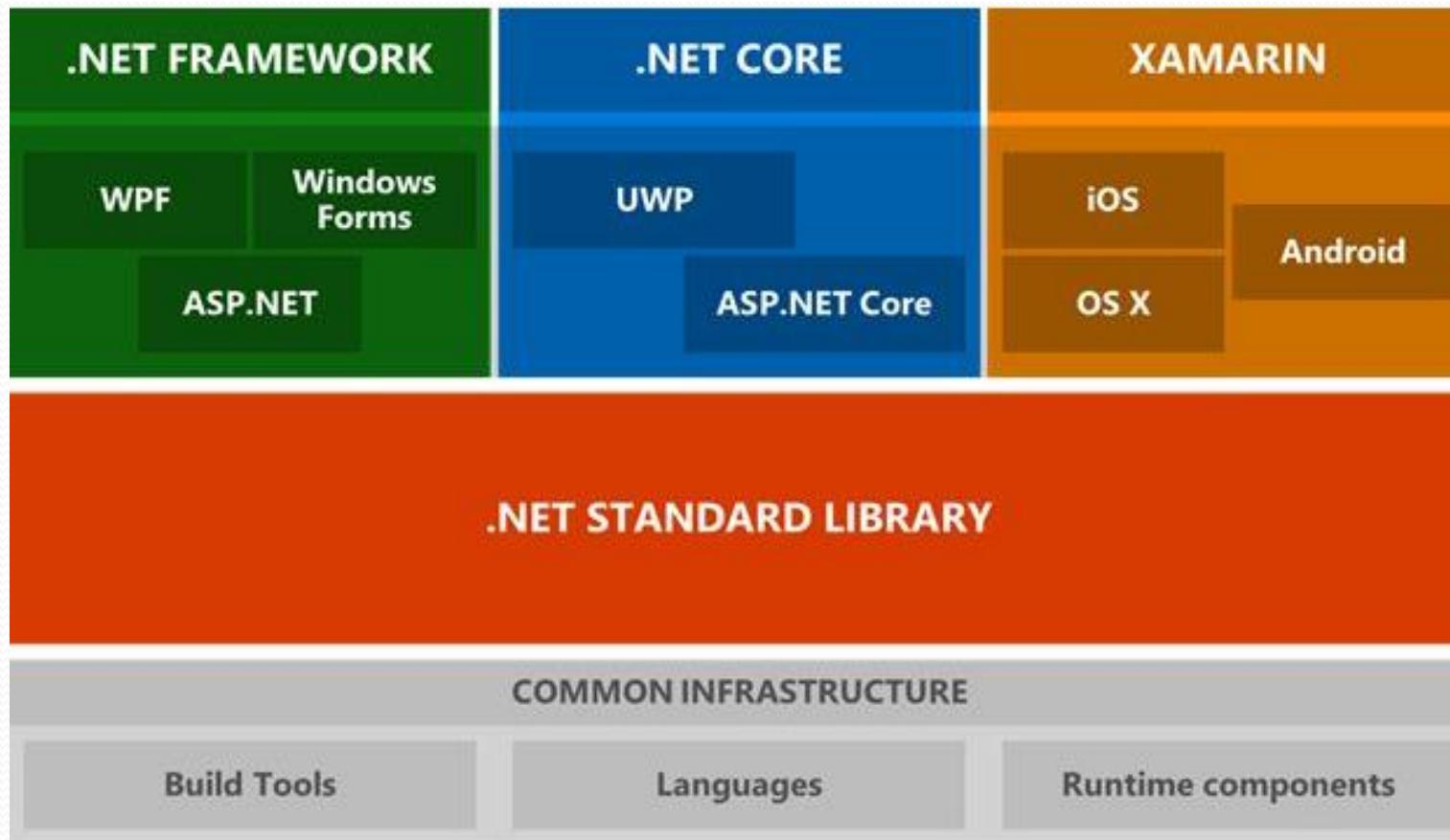
- Các class được định nghĩa trong hệ thống thư viện cơ bản của .NET gọi tắt là BCL (Base class libraries)

➤ Công cụ phát triển

- Các công cụ của .NET bao gồm compiler và Visual Studio .NET.
- Đối với nền tảng .NET core mới thì có thêm công cụ dòng lệnh (dotnet cli)

1. Tổng quan về .Net, Visual Studio

➤ 3 phiên bản của .Net



1. Tổng quan về .Net, Visual Studio

➤ 3 phiên bản của .Net:

- **Xamarin (Mono)**: hỗ trợ phát triển các ứng dụng di động (iOS, Android, Windows mobile) trên nền tảng khác Windows.
- **.Net Framework** hỗ trợ phát triển các ứng dụng Windows và Web trên nền tảng Windows.
- **.Net Core** là một framework mã nguồn mở và đa nền tảng để xây dựng các ứng dụng chạy trên nhiều hệ điều hành bao gồm Windows, Mac OS hay Linux. .Net Core chỉ hỗ trợ UWP và ASP.Net Core.

1. Tổng quan về .Net, Visual Studio

- Môi trường thực thi ứng dụng .NET: Microsoft .NET Framework

<http://www.microsoft.com/downloads/>

- Trình soạn thảo và biên dịch:
 - Visual Studio .NET IDE
 - Trình soạn thảo văn bản (Notepad, UltraEdit...) & Trình biên dịch bằng dòng lệnh (Commandline compiler)

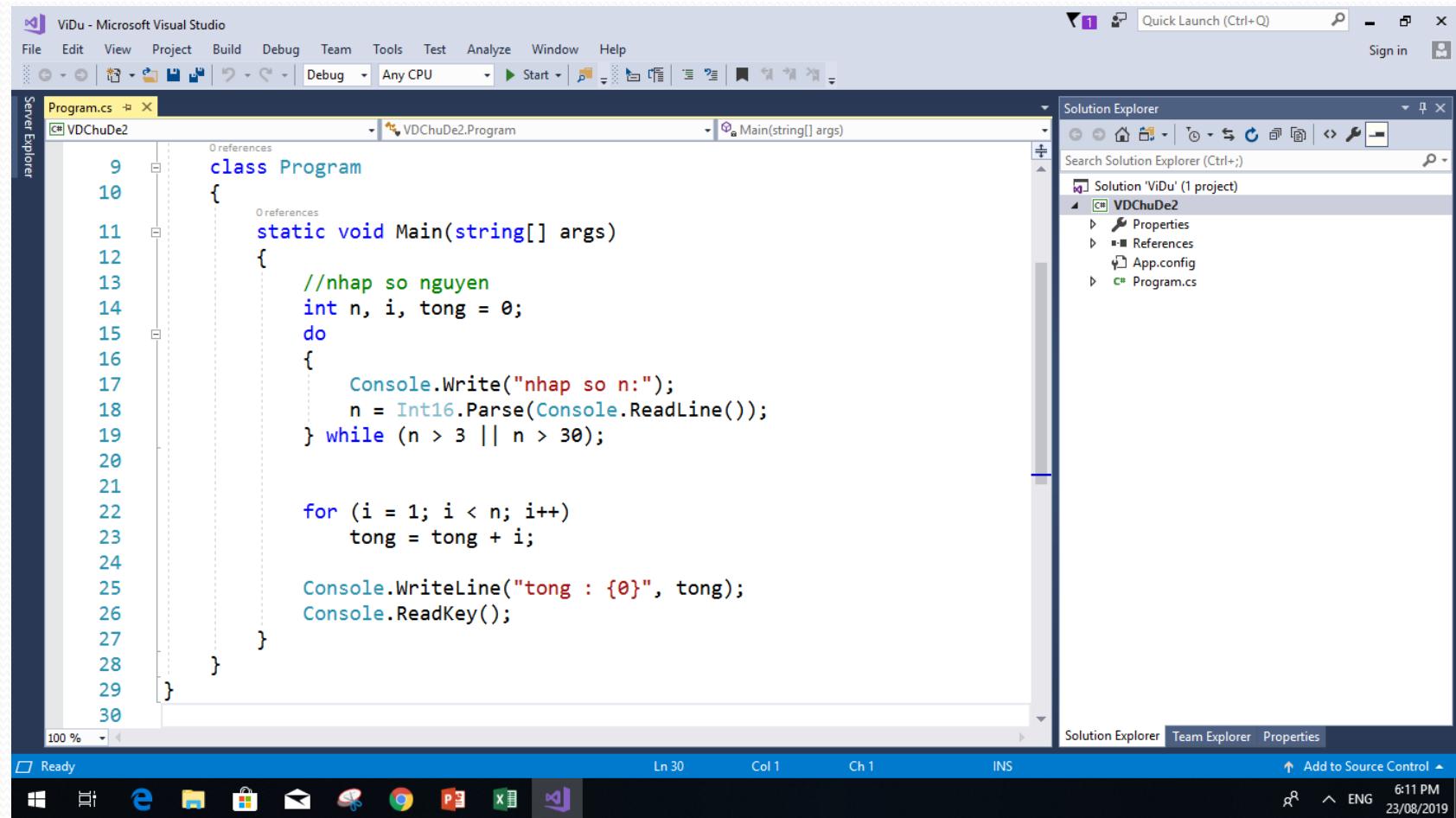
1. Tổng quan về .Net, Visual Studio

➤ **Microsoft Visual Studio (VS)** là một IDE cung cấp:

- Các chức năng cơ bản: viết mã, build và debug.
- Làm việc nhóm thông qua Team Foundation Server của Microsoft.
- Các phím tắt và plugins hỗ trợ người dùng thao tác nhanh trong việc viết mã.
- Tùy chỉnh liên kết các project và thư viện, tập tin liên quan.
- ...

1. Tổng quan về .Net, Visual Studio

➤ Giao diện

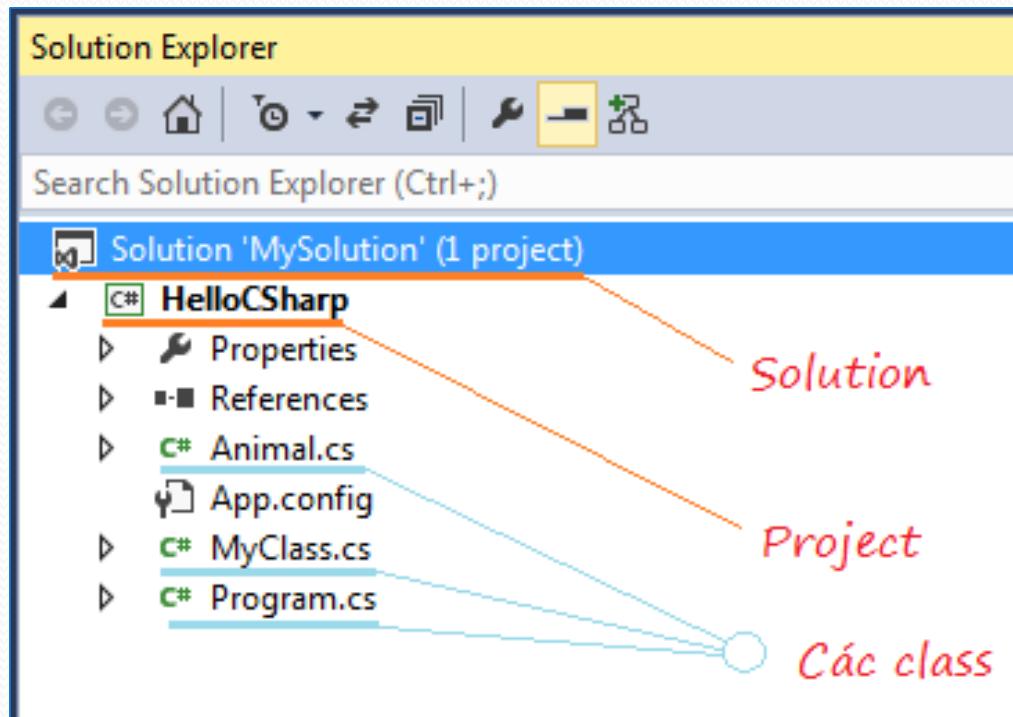


The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays a C# code editor with the file `Program.cs` open. The code implements a simple program that reads an integer from the user, calculates the sum of its digits, and prints the result. The Solution Explorer on the right shows a single project named `VDChuDe2` containing files `Properties`, `References`, `App.config`, and `Program.cs`. The status bar at the bottom provides information about the current file, line, column, and character.

```
9 class Program
10 {
11     static void Main(string[] args)
12     {
13         //nhap so nguyen
14         int n, i, tong = 0;
15         do
16         {
17             Console.WriteLine("nhap so n:");
18             n = Int16.Parse(Console.ReadLine());
19         } while (n > 3 || n > 30);
20
21
22         for (i = 1; i < n; i++)
23             tong = tong + i;
24
25         Console.WriteLine("tong : {0}", tong);
26         Console.ReadKey();
27     }
28 }
29 }
```

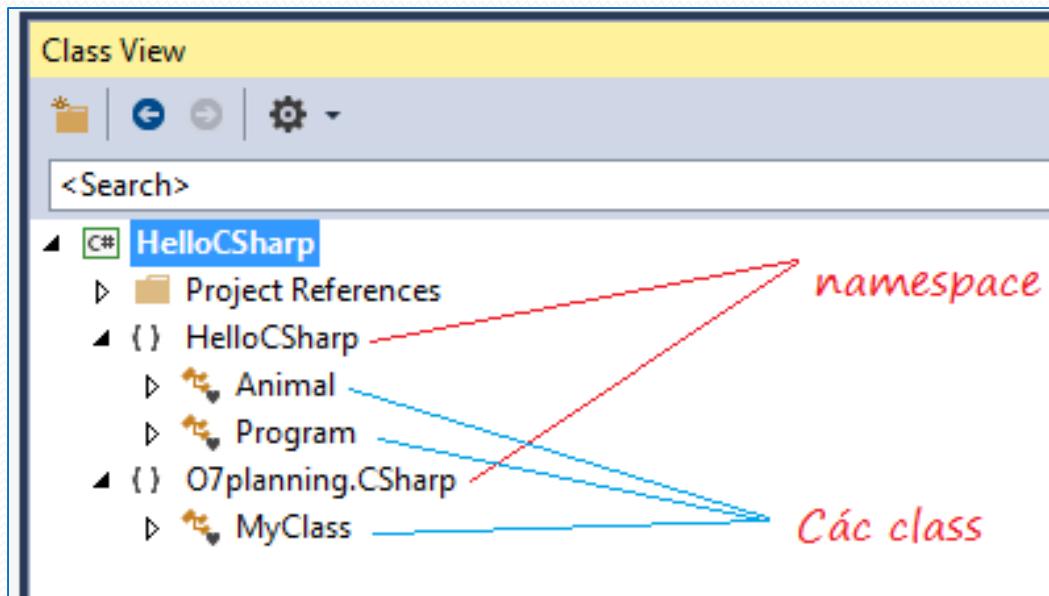
1. Tổng quan về .Net, Visual Studio

- Cách tổ chức: Một giải pháp (Solution) có thể chứa trong nó nhiều dự án (Project). Trong các Project chứa các lớp (class).



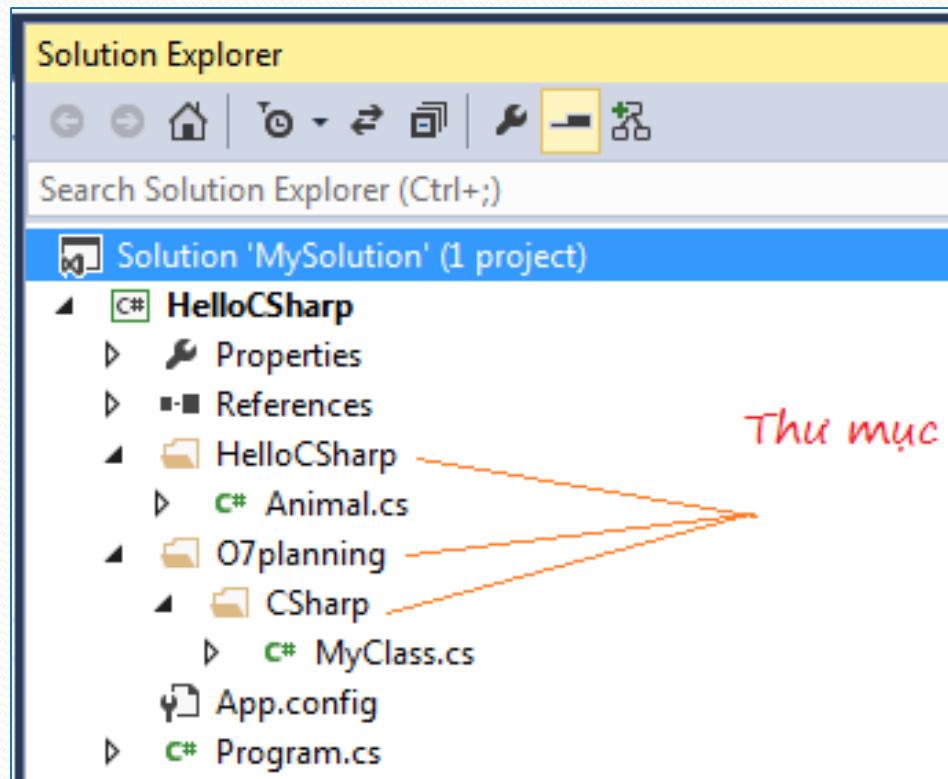
1. Tổng quan về .Net, Visual Studio

- Cách tổ chức: Khi nhìn trên "Class view" có thể thấy các lớp thuộc vào không gian tên (namespace) nào.

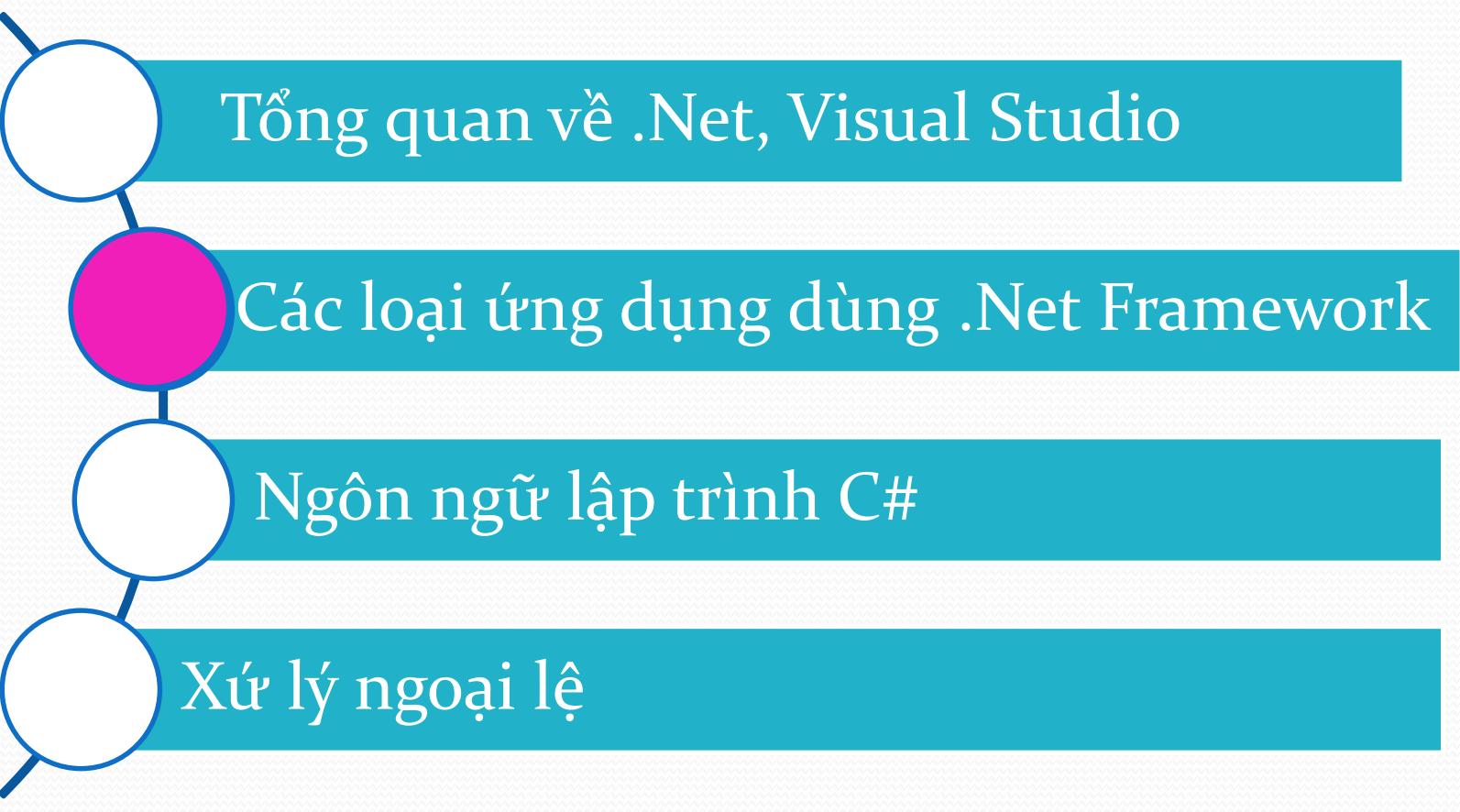


1. Tổng quan về .Net, Visual Studio

- Cách tổ chức: Có thể tạo ra các thư mục khác nhau để chứa các class.



Nội dung



Tổng quan về .Net, Visual Studio

Các loại ứng dụng dùng .Net Framework

Ngôn ngữ lập trình C#

Xử lý ngoại lệ

2. Các loại ứng dụng dùng .Net Framework

- Sử dụng .NET Framework để phát triển những kiểu ứng dụng và dịch vụ sau:
 - Ứng dụng Console
 - Ứng dụng giao diện GUI trên Windows (Windows Forms)
 - Ứng dụng Web (Web Forms)
 - Dịch vụ XML Web
 - ...

2. Các loại ứng dụng dùng .Net Framework

➤ Ứng dụng Console

```
E:\Lecturers\Lap trinh HDT C#\K60\BT Nhóm\OOP-60.CNTT-Bài tập nhóm - 60CNTT-3-32670\M  
Cap do choi:  
1) de  
2) trung binh  
3) kho  
Ban chon cap do choi:1  
----- huong dan -----  
- chon 0 lam moc  
- nhan len xuong qua ve de di chuyen so 0  
- tu do di chuyen cac con so con lai de sap xep ma tran theo mau  
- thoat - an Esc  
- nhan 1 phím bat ki de bat dau
```

2. Các loại ứng dụng dùng .Net Framework

➤ Ứng dụng Windows Forms

Admin

30/11/2019 7:13:35 CH

Quản Lý Nhà Trọ

Danh sách các phòng

Thông Tin Chi Tiết

Số người	<input type="text"/>	<input type="button" value="−"/>	Ngày Thuê	<input type="text"/>
Tiền Điện	<input type="text"/> vnd			
Tiền Nước	<input type="text"/> vnd			
Tiền Nhà	1.500.000 vnd			
Tổng Chi Trả	<input type="text"/> vnd			

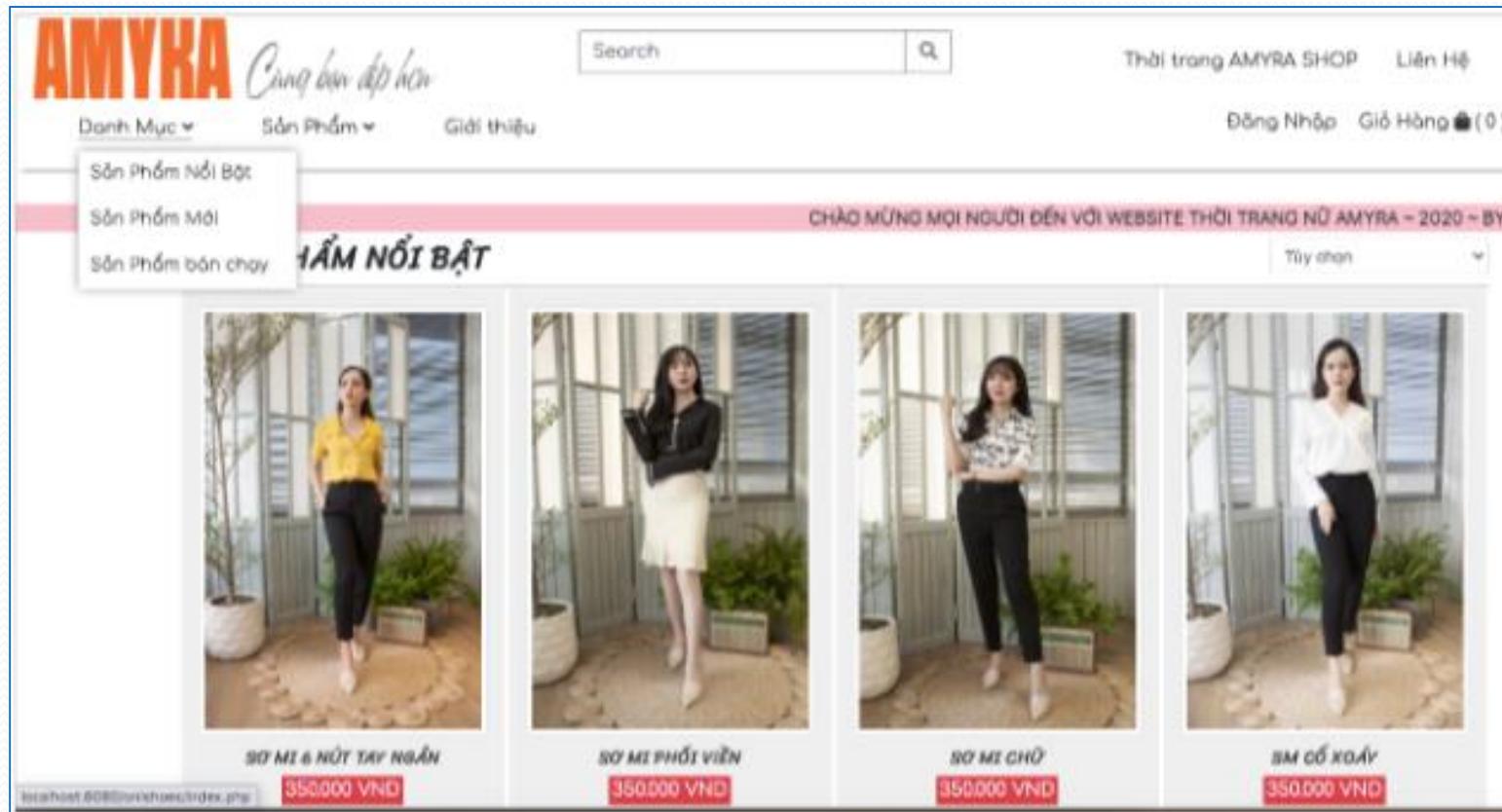
Các nút: Cập Nhật, Xác Nhận, Đóng Tiền, Xóa



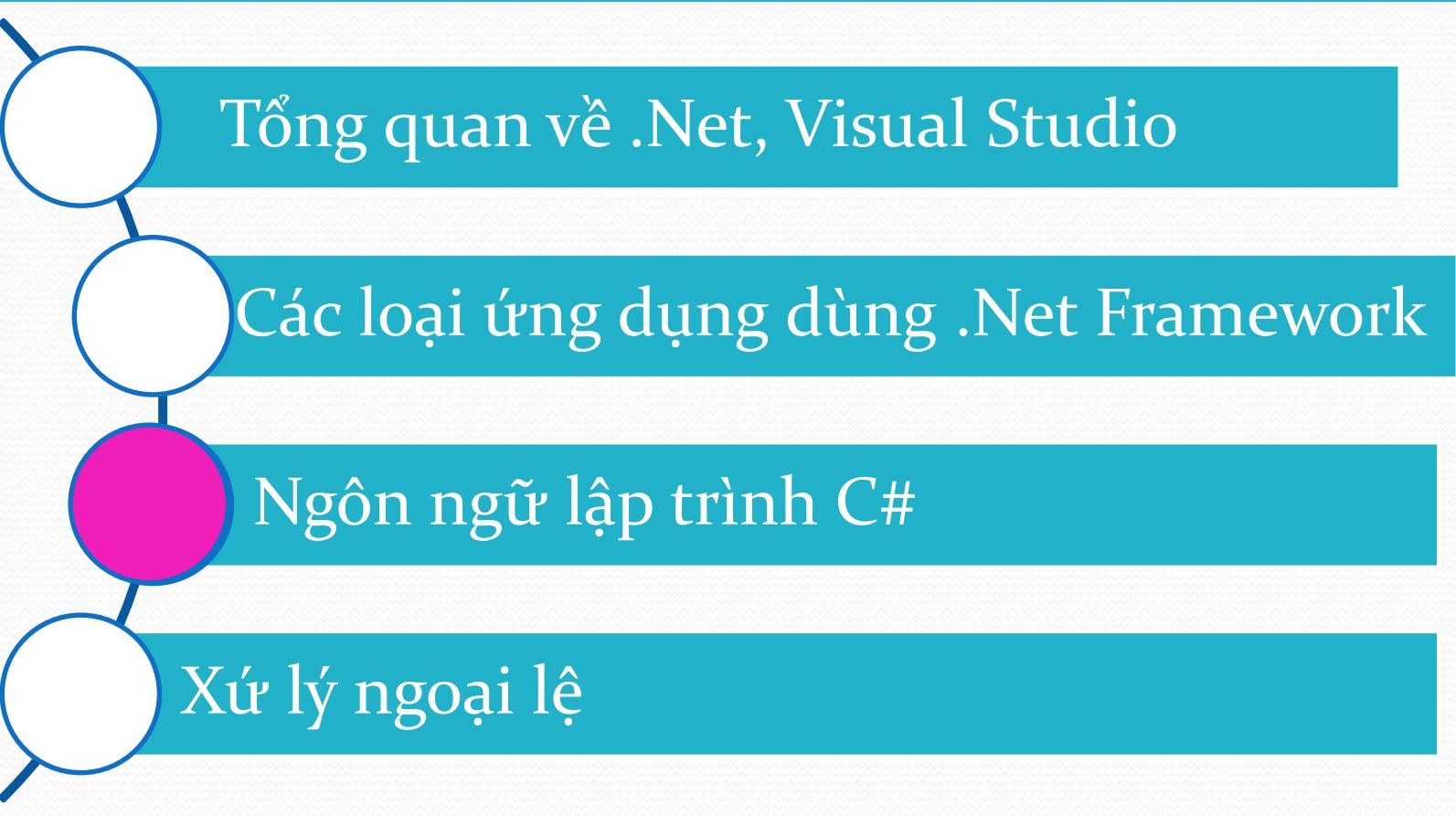
Legend:
Blue square: Đang trống
Red square: Đã có chủ
Yellow square: Tới hạn thu tiền

2. Các loại ứng dụng dùng .Net Framework

➤ Ứng dụng Web Forms



Nội dung



Tổng quan về .Net, Visual Studio

Các loại ứng dụng dùng .Net Framework

Ngôn ngữ lập trình C#

Xử lý ngoại lệ

3. Ngôn ngữ lập trình C#

➤ Cấu trúc chương trình C#

```
using <namespace sử dụng>
...
namespace <Tên Namespace>
{
    [Khóa truy xuất] class <Tên lớp>
    {
        public static void Main()
        {
            ...
        }
        // thành viên khác ...
    }
    // lớp khác ...
}
```

3. Ngôn ngữ lập trình C#

➤ Ví dụ chương trình HelloWorld

Using statement

```
using System;
using System.Collections.Generic;
using System.Text;
```

Namespace

```
namespace HelloWorld
```

class

```
class Program
```

Static function

```
static void Main(string[] args)
```

Code statement

```
Console.WriteLine("Hello World");
```

3. Ngôn ngữ lập trình C#

➤ Không gian tên (Namespaces)

- Cung cấp một cách để giữ một tập hợp các tên được phân biệt riêng rẽ nhau.
- Ưu điểm:
 - Tránh được sự trùng lặp tên giữa các lớp
 - Cho phép tổ chức mã nguồn một cách khoa học và hợp lý
 - Cho phép dễ dàng tái sử dụng mã
- Trong thư viện .NET framework có nhiều không gian tên, phải tham chiếu tới để sử dụng qua từ khóa using

Ví dụ: using System



3. Ngôn ngữ lập trình C#

➤ Các Namespaces cơ bản

Namespace	Description
System	Chứa lớp toán học, chuyển đổi dữ liệu
System.IO	Các lớp cho thao tác Input và Output
System.Net	Các lớp liên quan đến network protocol
System.Collections	Chức các lớp liên quan đến xử lý tập hợp
System.Data	Các lớp của ADO.NET
System.Drawing	Các lớp thực thi chức năng GUI
System.Threading	Các lớp lập trình MultiThread
System.Web	Các lớp liên quan đến HTTP protocol
System.Xml	Các lớp liên quan XML

4. Ngôn ngữ lập trình C#

- **class Program { ... }**
- **Phương thức Main():** là điểm bắt đầu để chạy ứng dụng
 - Phương thức có đối số

```
static void Main(string[] args)
```
 - Phương thức không có đối số

```
static void Main()
```
 - Phương thức trả về giá trị kiểu int

```
static int Main(string[] args)
```

```
static int Main()
```

3. Ngôn ngữ lập trình C#

➤ Nhập trong ứng dụng Console (Console I/O)

- Đọc ký tự văn bản từ cửa sổ console:
 - **Console.Read()**: Đọc **một ký tự** từ bàn phím và trả về kiểu **số nguyên** là mã ASCII của ký tự đó.
 - **Console.ReadLine()**: Đọc dữ liệu từ bàn phím cho đến khi gặp **ký tự xuống dòng** thì dừng, giá trị đọc được luôn là **một chuỗi**.
 - **Console.ReadKey()**: Đọc **một ký tự** từ bàn phím nhưng trả về kiểu **ConsoleKeyInfo**.

3. Ngôn ngữ lập trình C#

➤ Xuất trong ứng dụng Console (Console I/O)

- Xuất chuỗi ký tự:
 - `Console.WriteLine(<giá trị cần xuất ra màn hình>)`
 - `Console.WriteLine(<giá trị cần xuất ra màn hình>)`

3. Ngôn ngữ lập trình C#

➤ Định dạng xuất

- Xuất giá trị biến theo thứ tự: Dùng cặp dấu { } và số thứ tự bên trong {i}

```
int i=5; j=10;
```

```
Console.WriteLine("i={0}, j={1}", i, j);
```

- Sử dụng hàm static của lớp String là String.Format

```
float a = 3.12345F;
```

```
String.Format ("{0:0.0}", a);
```

```
// String.Format ("{0:0.#}", a);
```

3. Ngôn ngữ lập trình C#

➤ Kiểu dữ liệu:

- Phân loại dữ liệu:
 - Phân theo phương thức **định nghĩa**: **build-in** (có sẵn) và **user-defined** (người dùng tự định nghĩa)
 - Phân theo cách thức **lưu trữ**: **value** (tham trị) và **reference** (tham chiếu)

3. Ngôn ngữ lập trình C#

➤ Kiểu dữ liệu có sẵn:

Name	CTS Type	Size	Range
sbyte	System.SByte	8	-128..127
short	System.Int16	16	(-32768 .. 32767)
int	System.Int32	32	-2 ³¹ ..2 ³¹ -1
long	System.Int64	64	-2 ⁶³ ..2 ⁶³ -1
byte	System.SByte	8	0..255
ushort	System.UInt16	16	(0 .. 65535)
uint	System.UInt32	32	0..2 ³² -1
ulong	System.UInt64	64	0..2 ⁶⁴ -1
float	System.Single	32	xấp xỉ từ 3,4E - 38 đến 3,4E+38
double	System.Double	64	1,7E-308 đến 1,7E+308
decimal	System.Decimal	128	Có độ chính xác đến 28 con số
bool	System.Boolean		Kiểu true/false
char	System.Char	16	Ký tự unicode

3. Ngôn ngữ lập trình C#

➤ Kiểu dữ liệu:

Kiểu người dùng tự định nghĩa: class,...

Ví dụ: class SinhVien

3. Ngôn ngữ lập trình C#

➤ Biến (variable):

- Một vùng nhớ có định kiểu
- Có thể gán và thay đổi được giá trị
- Các biến phải được khởi gán trước khi sử dụng, nếu không, trình biên dịch sẽ báo lỗi
- Khuyến cáo đặt tên các biến, các phương thức, các lớp:
 - Tên biến: bắt đầu bằng chữ thường (VD: **someName**)
 - Tên phương thức và các thành phần khác: bắt đầu bằng chữ hoa (VD: **SomeOtherMethod**)

3. Ngôn ngữ lập trình C#

➤ Biến (variable):

- Khai báo

[tầm vực] (kiểu dữ liệu)[?] tên biến [= giá trị khởi tạo];

- Ví dụ:

int i;

float x = 0;

int? j = null;

3. Ngôn ngữ lập trình C#

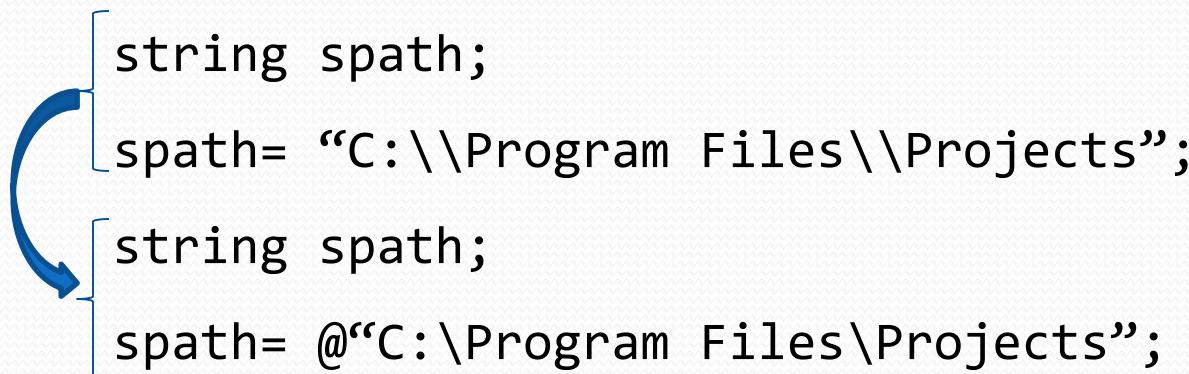
➤ Hằng (Constant):

- Là biến số nhưng không thể thay đổi giá trị sau khi khởi gán.

Ví dụ: const int myConst = 32;

- Ký tự đặc biệt @ trong giá trị dạng chuỗi

Ví dụ



```
string spath;  
spath= "C:\\Program Files\\Projects";  
  
string spath;  
spath= @"C:\\Program Files\\Projects";
```

3. Ngôn ngữ lập trình C#

➤ **Các toán tử (Operators):** Toán tử được phân thành nhiều loại khác nhau

- Toán tử gán: `=, -=, +=, /=, *=, &=, |=, <<=, ...`
- Toán tử số học: `+, -, *, /, %`
- Toán tử tăng, giảm: `++, -`
- Toán tử quan hệ: `==, !=, <, >, <=, >=`
- Toán tử logic: `!, &&, || ...`
- Toán tử ba thành phần

`conditional-expression ? expression1 : expression2`

- Một số phép toán khác: `as, is, sizeof, typeof` để kiểm tra loại đối tượng.

3. Ngôn ngữ lập trình C#

➤ Chuyển đổi kiểu dữ liệu:

- Chuyển đổi **ngầm** (implicity)
- Chuyển đổi **tường minh** (explicity)
- Dùng lớp **Convert**: Convert.ToDateTime(Sourcevalue)
- Dùng phương thức **ToString()**
- Dùng phương thức **Parse()**
`<DataType>.Parse(<Sourcevalue>);`
- Dùng phương thức **TryParse()**
`<DataType>.TryParse(<Sourcevalue>, out <var_rs>);`

3. Ngôn ngữ lập trình C#

➤ Chuyển đổi kiểu dữ liệu:

- Chuyển đổi **ngầm** (implicity): quá trình chuyển đổi diễn ra tự động và đảm bảo không bị mất mát dữ liệu

Ví dụ: short x = 5; int y = x;

- Chuyển đổi **tường minh** (explicity) sử dụng toán tử chuyển đổi (cast operator)

Ví dụ: double a = 34.5; int b = (int) a;

- Dùng lớp **Convert**: Convert.ToDateTime(Sourcevalue)

Ví dụ: string s1 = "50.5";

double x = Convert.ToDouble(s1);

int y = Convert.ToInt32(s1);

3. Ngôn ngữ lập trình C#

➤ Chuyển đổi kiểu dữ liệu:

- Phương thức **ToString()**

Ví dụ:

```
Console.WriteLine("x+y=" + Convert.ToString(x+y));  
Console.WriteLine("x+y=" + (x+y).ToString());
```

- Dùng phương thức **Parse()**

```
<DataType>.Parse(<Sourcevalue>);
```

Ví dụ:

```
string s1 = "50.5";  
double x = double.Parse(s1);
```

3. Ngôn ngữ lập trình C#

➤ Chuyển đổi kiểu dữ liệu:

- Dùng phương thức TryParse()

```
<DataType>.TryParse(<Sourcevalue>, out <var_rs>);
```

Ví dụ:

```
string s1 = "50.5";  
  
double num;  
  
bool result = double.TryParse(s1, out num);  
  
if (result)
```

...

3. Ngôn ngữ lập trình C#

➤ Lệnh nhảy có điều kiện:

- Câu lệnh: if...else

```
if(Biểu thức điều kiện) Công việc1;  
[else Công việc 2;]
```

- Câu lệnh switch:

```
switch (biểu thức cần kiểm tra) {  
    case trường_hợp: {  
        Các câu lệnh  
        Lệnh nhảy(break, continue) }  
    [default: Các câu lệnh cho trường hợp mặc định]  
}
```

3. Ngôn ngữ lập trình C#

➤ Lệnh nhảy không điều kiện:

- break: thoát hay chuyển hướng của lệnh switch hay lệnh lặp.
- continue: chuyển việc thực hiện xử lý đến lần lặp tiếp.
- goto: chuyển điều khiển của chương trình trực tiếp đến nhãn.
- return: kết thúc phương thức trả điều khiển đến nơi phương thức được gọi.

3. Ngôn ngữ lập trình C#

➤ Lệnh lặp:

- Vòng lặp for

```
for ( [Khởi tạo]; [Biểu thức kiểm tra]; [Lệnh lặp])  
    { các câu lệnh }
```

- Vòng lặp while

```
while (Biểu thức kiểm tra) {các câu lệnh}
```

- Vòng lặp do...while

```
do Công việc while Biểu_thức_kiểm_tra  
{ các câu lệnh}
```

3. Ngôn ngữ lập trình C#

➤ Lệnh lặp:

- Vòng lặp foreach ..in: Lặp lại một hay nhiều câu lệnh ứng với mỗi phần tử duyệt qua trong một mảng hay tập đối tượng.

```
foreach (Kiểu_dữ_liệu biến in mảng/tập_đối_tượng)
    { các câu lệnh }
```

3. Ngôn ngữ lập trình C#

➤ Phương thức (hàm):

- Cú pháp [phạm vi] Kiểu_DL_trả_về Tên_PT(các tham số){thân phương thức}
- Truyền tham số: Mặc định, tham số truyền cho phương thức là kiểu tham trị
 - Một bản sao của tham số đó được tạo ra
 - Bản sao đó sẽ bị hủy khi kết thúc phương thức
 - Giá trị của tham số được truyền không thay đổi sau khi kết thúc phương thức

Ví dụ: public int AddValue(int value1, int value2) { return value1+value2; }

3. Ngôn ngữ lập trình C#

➤ Phương thức (hàm):

- Truyền tham chiếu: C# hỗ trợ truyền tham chiếu sử dụng các từ khóa
 - ref: truyền tham chiếu, biến được tham chiếu phải được khởi gán trước khi truyền
 - out: truyền tham chiếu, biến được tham chiếu không cần khởi gán trước khi truyền

3. Ngôn ngữ lập trình C#

➤ Truyền tham chiếu ref:

```
class Phuong_thuc {  
    public static void Cong(ref int a, ref int b){  
        a = a+5; b = b+10; }  
    static void Main(string[] args) {  
        Console.WriteLine("nhap x, y:");  
        int x = int.Parse(Console.ReadLine());  
        int y = int.Parse(Console.ReadLine());  
        Console.WriteLine("Tham chieu ref:");  
        Cong(ref x, ref y);  
        Console.WriteLine("{0}, {1}", x, y);  
        Console.ReadLine(); }  
}
```

3. Ngôn ngữ lập trình C#

➤ Truyền tham chiếu out:

```
class Phuong_thuc {  
    public static void Cong(out int a, out int b){  
        a = 0; b = 0;  
        a = a+5; b = b+10; }  
    static void Main(string[] args) {  
        Console.WriteLine("nhap x, y:");  
        int x = int.Parse(Console.ReadLine());  
        int y = int.Parse(Console.ReadLine());  
        Cong(out x, out y);  
        Console.WriteLine("{0}, {1}", x, y);  
        Console.ReadLine(); }  
}
```

3. Ngôn ngữ lập trình C#

➤ Kiểu dữ liệu mảng:

- Khai báo:

```
Kiểu_DL[] Tên_mảng;
```

- Khởi tạo

```
Tên_mảng= new Kiểu_DL[số_phần_tử];
```

- Khai báo kết hợp khởi tạo

```
Kiểu_DL[] Tên_mảng= new Kiểu_DL[số_phần_tử];
```

3. Ngôn ngữ lập trình C#

➤ Kiểu dữ liệu mảng:

- Khai báo và có giá trị mặc định

```
Kiểu_DL[] Tên_mảng={giátrị1, gt2, gt3, ght4, gt5};
```

- Truy xuất phần tử

```
Tên_mảng[chỉ số]
```

- Lấy chiều dài mảng

```
Tên_mảng.Length
```

```
Tên_mảng.GetLength(0)
```

3. Ngôn ngữ lập trình C#

➤ Ví dụ mảng 1 chiều:

```
class Program
{
    static void Main(string[] args)
    {
        int[] A = new int[10];
        Console.WriteLine("Nhập số phần tử mảng:");
        byte n = Convert.ToByte(Console.ReadLine());
        //nhập mảng
        for (int i = 0; i < n; i++)
        {
            Console.WriteLine("A[{0}]:",i);
            A[i] = Convert.ToInt16(Console.ReadLine());
        }
        //tính tổng các phần tử
        int tong = 0;
        foreach (int x in A)
            tong += x;
        Console.WriteLine("Tổng mảng:{0}",tong);
        Console.ReadLine();
    }
}
```

```
for (int i = 0; i < A.Length; i++)
    tong += A[i];
```

3. Ngôn ngữ lập trình C#

➤ Kiểu dữ liệu mảng:

- Khai báo mảng 2 chiều:

Kiểu_DL[,] Tên_mảng;

- Khởi tạo

Tên_mảng = new Kiểu_DL[số phần tử1, số phần tử2];

- Khai báo kết hợp khởi tạo

Kiểu_DL[,] Tên_mảng = new Kiểu_DL[số pt1, số pt2];

3. Ngôn ngữ lập trình C#

➤ Ví dụ mảng 2 chiều:

```
class Mang_n_chieu
{
    static void Main(string[] args)
    {
        int[,] A = new int[2,3];
        //nhập mảng
        for (int i = 0; i < 2; i++)
            for(int j=0;j<3;j++)
        {
            Console.WriteLine("A[{0},{1}]:",i,j);
            A[i,j] = Convert.ToInt16(Console.ReadLine());
        }
        A.SetValue(15, 1,2);
        //in mảng
        for (int i = 0; i < A.GetLength(0); i++)
        {
            for (int j = 0; j < A.GetLength(1); j++)
            {
                Console.Write("{0}    ",A[i,j]);
            }
            Console.WriteLine();
        }
        Console.ReadLine();
    }
}
```

3. Ngôn ngữ lập trình C#

➤ Kiểu dữ liệu mảng:

- Mỗi phần tử của mảng có kiểu dữ liệu khác nhau.

```
object[] Tên_mảng = new object[ số phần tử];
```

- Các lớp cung cấp các phương thức dung để tạo, thao tác, tìm kiếm và sắp xếp mảng cho các loại mảng: Array, ArrayList, List,...

3. Ngôn ngữ lập trình C#

➤ Ví dụ mảng các phần tử là object:

```
class Mang_object
{
    static void Main(string[] args)
    {
        object[] A = new object [3];

        //gán giá trị cho các phần tử mảng
        A[0] = "Le Thi Thanh Huong";
        A[1] = "Lop CTH53";
        A[2] = 1991;
        //in mảng
        for (int i = 0; i < A.GetLength(0); i++)
        {
            Console.WriteLine("{0}    ", A[i]);
        }
        Console.ReadLine();
    }
}
```

3. Ngôn ngữ lập trình C#

➤ **Lớp Array:** Cung cấp các phương thức dùng để tạo, thao tác, tìm kiếm và sắp xếp mảng cho các loại mảng.

```
class Program
{
    //in mảng
    public static void In_Mang(int []A)
    {
        foreach (int i in A)
        {
            Console.Write("{0} ", i);
        }
        Console.WriteLine();
    }
    static void Main(string[] args)
    {
        int[] A = new int [5] {123,234,12,453,654};
        Array.Sort(A);
        In_Mang (A);
        Console.ReadLine();
    }
}
```

Nội dung

Tổng quan về .Net, Visual Studio

Các loại ứng dụng dùng .Net Framework

Ngôn ngữ lập trình C#

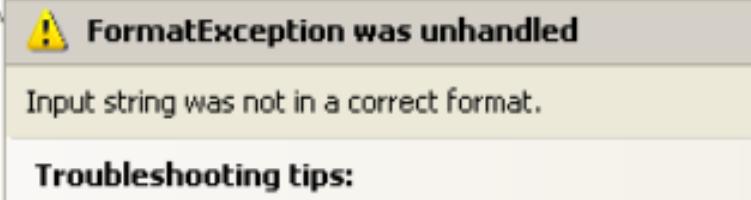
Xử lý ngoại lệ

4. Xử lý ngoại lệ

➤ Tình huống phát sinh ngoại lệ:

```
static void Main()
{
    int i; Console.Write("Nhập giá trị cho i: ");
    i = int.Parse(Console.ReadLine());
}
```

```
i = int.Parse(Console.ReadLine());
Console.WriteLine("giá trị i" + i)
Console.ReadLine();
```

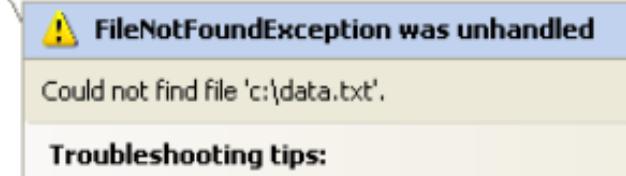


4. Xử lý ngoại lệ

➤ Tình huống phát sinh ngoại lệ:

```
static void Main()
{
    StreamReader sr = new StreamReader("c:\\data.txt");
    int a = int.Parse(sr.ReadLine());
    int b = int.Parse(sr.ReadLine());
    Console.WriteLine("Giá trị a+b= " +(a+b));
}
```

```
StreamReader sr = new StreamReader("c:\\data.txt");
int a = int.Parse(sr.ReadLine());
int b = int.Parse(sr.ReadLine());
Console.ReadLine();
```



4. Xử lý ngoại lệ

➤ Các loại lỗi:

- Syntax error: biên dịch chương trình
- Runtime Error: chạy chương trình (ngoại lệ)
- Logical Error: thuật giải sai, tính toán sai.

➤ C# sử dụng kỹ thuật bắt ngoại lệ (Handling Exception) để bắt và xử lý lỗi (error) phát sinh trong quá trình thực thi chương trình.

➤ Hai cơ chế xử lý ngoại lệ:

- Giải quyết tức thì: C# sử dụng cấu trúc **try...catch...finally** để kiểm tra, bắt và xử lý ngoại lệ.
- Ném ngoại lệ ra ngoài **throw...**

4. Xử lý ngoại lệ

➤ Cấu trúc try ... catch ... finally

```
try
{
    // các lệnh có thể phát sinh ngoại lệ
}
catch [(Exception e)]
{
    // các lệnh xử lý ngoại lệ
}
finally
{
    // các lệnh thực hiện bất kể có xuất hiện ngoại lệ hay không
}
```

4. Xử lý ngoại lệ

➤ Ví dụ

```
static void Main()
{
    int i=0;
    Console.WriteLine("nhap i:");
    try
    {
        i = int.Parse(Console.ReadLine());
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        // Console.WriteLine("gia tri nhap khong hop le!");
    }
    finally
    { Console.WriteLine("gia tri i" + i); }
}
```

4. Xử lý ngoại lệ

➤ **throw ...**

```
throw đối_tượng_lớp_Exception;
```

- Khi một ngoại lệ được tung ra, chương trình ngay lập tức sẽ dừng lại và CLR sẽ tìm , kiểm tra chương trình bắt ngoại lệ, nếu không tìm thấy nó sẽ kết thúc chương trình

4. Xử lý ngoại lệ

➤ Ví dụ

```
static void Nhap()
{
    int i = 0;
    Console.WriteLine("nhap i:");
    try
    {
        i = int.Parse(Console.ReadLine());
    }
    catch (Exception ex) { throw ex; }
    Console.WriteLine(i);
}
static void Main(string[] args)
{
    try { Nhap(); }
    catch { Console.WriteLine("gia tri kg hop le!"); }
}
```

4. Xử lý ngoại lệ

- Có thể có nhiều đoạn lệnh **catch** trong một câu lệnh **try...catch** tương ứng với nhiều ngoại lệ khác nhau. Xử lý **catch** sẽ theo thứ tự từ class con đến class cha.
- Đoạn lệnh **try...catch** có thể đặt trong phương thức có thể phát sinh ngoại lệ hoặc đặt ở cấp cao hơn, phương thức triệu gọi đoạn mã có thể phát sinh ngoại lệ

4. Xử lý ngoại lệ

```
static void Cong()
{ int a,b = 0;
  Console.WriteLine("nhap a,b:");
  try
  { a = int.Parse(Console.ReadLine());
    b = int.Parse(Console.ReadLine());
    double kq = a / b;
  } catch (DivideByZeroException)
  { Console.WriteLine("loi chia o!"); }
  catch (FormatException)
  { Console.WriteLine("gia tri nhap kg hop le!"); }
  catch (Exception ex)
  { Console.WriteLine(ex.Message) ; }
}
```

4. Xử lý ngoại lệ

➤ Một số lớp ngoại lệ của .Net

MethodAccessException	Lỗi truy cập, do truy cập đến thành viên hay phương thức không được truy cập
ArgumentException	Lỗi tham số đối mục
ArgumentNullException	Đối mục Null, phương thức được truyền đối mục Null không được chấp nhận
ArithmeticException	Lỗi liên quan đến các phép toán
ArrayTypeMismatchException	Kiểu mảng không hợp, khi cố lưu trữ kiểu không thích hợp vào mảng
DivideByZeroException	Lỗi chia cho zero
FomatException	Định dạng không chính xác một đối mục nào đó

4. Xử lý ngoại lệ

➤ Một số lớp ngoại lệ của .Net

IndexOutOfRangeException	Chỉ số truy cập mang không hợp lệ, dùng nhỏ hơn chỉ số nhỏ nhất, lớn hơn chỉ số lớn nhất của mảng.
invalidCastException	Phép gán không hợp lệ
MulticastNotSupportedException	MultiCast không được hỗ trợ, do việc kết hợp hai delegate không đúng
NotFiniteNumberException	Không phải số hữu hạn, số không hợp lệ
NotSupportedException	Phương thức không hỗ trợ, khi gọi một phương thức không tồn tại bên trong lớp

Kết thúc chủ đề 2

