CHỦ ĐỀ 2

MICROSOFT .NET và C#

GV: Nguyễn Đình Hưng

Email: hungnd@ntu.edu.vn

1

1

Nội dung

- Tổng quan về .Net, Visual Studio
- Các loại ứng dụng dung .Net Framework
- Ngôn ngữ lập trình C#
- Xử lý ngoại lệ

2

2

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:

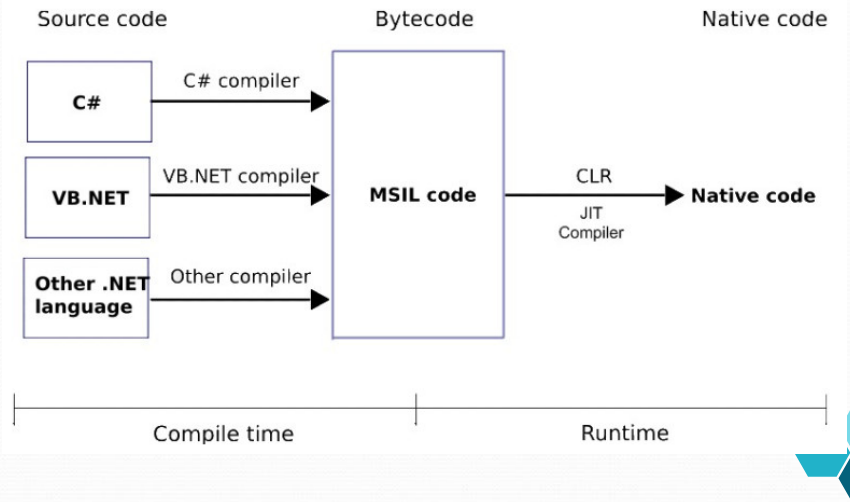
- **Runtime** (môi trường hoạt động)
- **Libraries** (thư viện)
- **Toolings** (công cụ phát triển)

3

3

1. Tổng quan về .Net, Visual Studio

➢ **Môi trường hoạt động**



4

4

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**

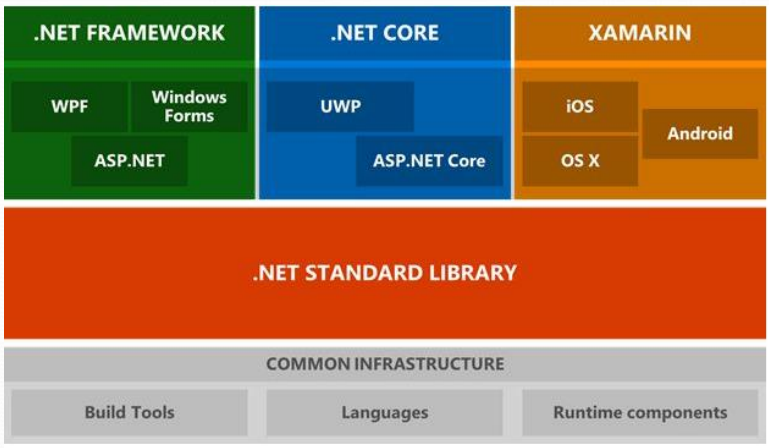
- Các công cụ của .NET bao gồm compiler và Visual Studio .NET.
- Đối với nền tảng .NET core mới thì có thêm công cụ dòng lệnh (dotnet cli)

5

5

1. Tổng quan về .Net, Visual Studio

➢ **3 phiên bản của .Net**



6

6

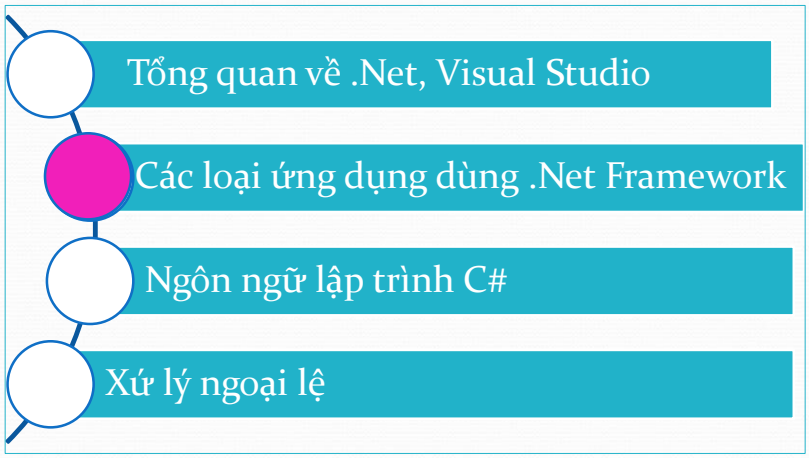
1. Tổng quan về .Net, Visual Studio

- **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.
  - ...

7

7

Nội dung



8

8

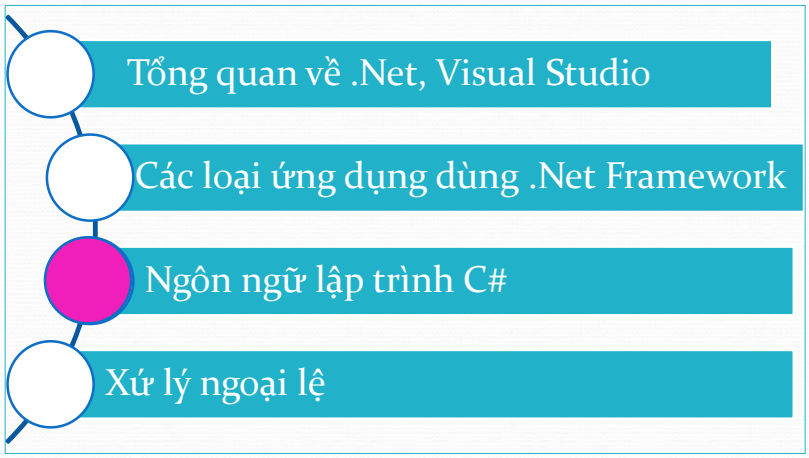
2. Các loại ứng dụng dùng .Net Framework

- 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 Froms)
  - Dịch vụ XML Web
  - ...

9

9

Nội dung



10

10

3. Ngôn ngữ lập trình C#

➢ Cấu trúc chương trình C#

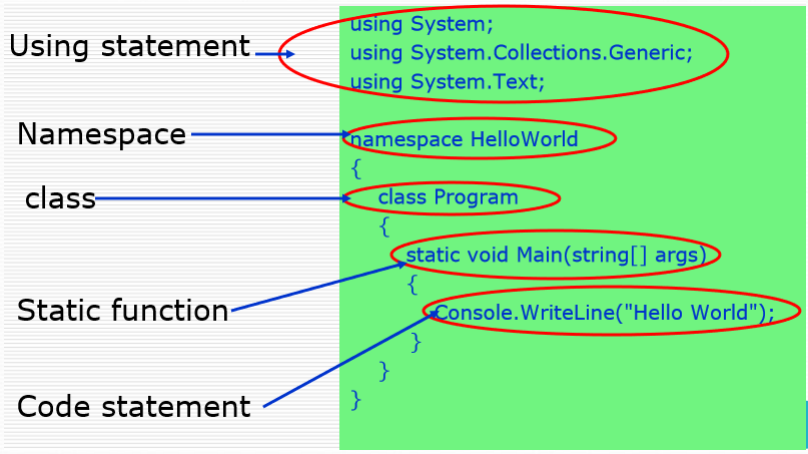
```
using <namespace sử dụng>
...
namespace <Tên Namespace>
{
        [Khóa truy xuất] class <Tên lớp>
        {
                public static void Main()
                {
                        ...
                }
                // thành viên khác ...
        }
        // lớp khác ...
```

11

11

3. Ngôn ngữ lập trình C#

➢ Ví dụ chương trình HelloWorld



12

12

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.**

13

3. Ngôn ngữ lập trình C#

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

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

14

3. Ngôn ngữ lập trình C#

➢ Định dạng xuất

- Xuất giá trị biến theo thứ tự: Dùng cặp dấu {} và số thự 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);
```

15

15

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)

16

16

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	Sytem. Int32	32	$-2^{31}..2^{31}-1$
long	Sytem. 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

17

3. Ngôn ngữ lập trình C#

➢ **Kiểu dữ liệu:**

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

Ví dụ: class SinhVien

18

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)

19

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;
```

20

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";
```

21

21

3. Ngôn ngữ lập trình C#

- **Các toán tử (Operators):** xem trong slide
- **Chuyển đổi kiểu dữ liệu:**
  - Chuyển đổi ngầm (implicity)
  - Chuyển đổi tường minh (explicity)
  - Dùng lớp Convert: Convert.ToDataType(Sourcevalue)
  - Dùng phương thức ToString()
  - Dùng phương thức Parse()

    <DataType>.Parse(<Sourcevalue>);
  - Dùng phương thức TryParse()

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

22

22

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.ToDataType(Sourcevalue)

```
Ví dụ: string  s1 = "50.5";
    double x = Convert.ToDouble(s1);
    int y = Convert.ToInt32(s1);
```

23

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);
```

24

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)
        …
```

25

25

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]
}
```

26

26

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.

27

27

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}
```

28

28

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 }
```

29

29

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; }

30

30

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

31

Nội dung

- Tổng quan về .Net, Visual Studio
- Các loại ứng dụng dùng .Net Framework
- Ngôn ngữ lập trình C#
- Xử lý ngoại lệ

32

4. Xử lý ngoại lệ

- Tình huống phát sinh ngoại lệ:

```
static void Main()
   {
      int i; Console.Write("Nhap gia tri cho i: ");
      i = int.Parse(Console.ReadLine());
   }
```

```
i = int.Parse(Console.ReadLine());
Console.WriteLine("gia tri i" + i);
Console.ReadLine ();
```

FormatException was unhandled

Input string was not in a correct format.

Troubleshooting tips:

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 tri 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 ();
```

FileNotFoundException was unhandled

Could not find file 'c:\data.txt'.

Troubleshooting tips:

4. Xử lý ngoại lệ

- **Các loại lỗi:**
  - Syntax error: biên dịch chương tình
  - 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 (errror) 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...

35

35

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 thi bất kể có xuất hiện ngoại lệ kg
}
```

36

36

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); }
    }
```

37

37

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

38

38

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!"); }
    }
```

39

39

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

40

40

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 0!"); }
    catch (FormatException)
    {  Console.WriteLine("gia tri nhap kg hop le!"); }
    catch (Exception ex)
    {  Console.WriteLine(ex.Message) ; }
}
```

41

41

41

Kết thúc chủ đề 2

42

42