

# LẬP TRÌNH GIAO DIỆN

Nguyễn Thị Mai Trang

Nguyễn Thị Mai Trang

1

1

## Chương 2

# Ngôn ngữ lập trình C#

2

## Mục tiêu

- Phân biệt và so sánh được các đặc điểm của ngôn ngữ C# và ngôn ngữ lập trình đã học (C++)
- Thao tác thành thạo trong môi trường Visual Studio.NET để xây dựng được ứng dụng bằng ngôn ngữ C#
- Sử dụng được cú pháp và ngôn ngữ C# trong lập trình
- Vận dụng được kỹ thuật xử lý ngoại lệ để phát hiện và xử lý lỗi chương trình

Nguyễn Thị Mai Trang

3

3

## NỘI DUNG

1. Giới thiệu ngôn ngữ lập trình C#
2. Các đặc điểm của ngôn ngữ
3. Các bước xây dựng một ứng dụng bằng C#
4. Từ khóa trong C#
5. Các kiểu dữ liệu cơ bản
6. Biến, hằng
7. Toán tử
8. Cấu trúc lựa chọn
9. Cấu trúc lặp
10. Xử lý ngoại lệ

Nguyễn Thị Mai Trang

4

4

## 2.1 Giới thiệu ngôn ngữ lập trình C#

- Được Microsoft công bố năm 2000
- Là một ngôn ngữ mạnh mẽ nhưng đơn giản dành cho các nhà phát triển → tạo ra các ứng dụng bằng cách sử dụng Microsoft.NET Framework.
- Được phát triển dựa trên nền tảng từ C ++,
  - Loại bỏ bớt những cú pháp không còn phù hợp
  - Bổ sung nhiều tính năng mới.

Nguyễn Thị Mai Trang

5

5

## Giới thiệu ngôn ngữ lập trình C# (tt)

- Ngôn ngữ C#:
  - Có nguồn gốc từ C, C++ → cú pháp gần giống như C++, loại bỏ macro, template, đa kế thừa.
  - Là ngôn ngữ hướng đối tượng hoàn toàn, hỗ trợ các đặc trưng của ngôn ngữ lập trình hướng đối tượng:
    - tính đóng gói (encapsulation)
    - tính đa hình (polymorphism)
    - tính kế thừa (inheritance).
  - Hỗ trợ mạnh mẽ về các cơ chế xử lý ngoại lệ, thu hồi bộ nhớ tự động, bảo mật mã nguồn...
  - Dùng để xây dựng nhiều loại ứng dụng như web, dịch vụ web, xử lý văn bản, đồ họa, bảng tính,...

Nguyễn Thị Mai Trang

6

6

## 2.2 Các đặc điểm của ngôn ngữ C#

- Đối với Lập trình trực quan: thao tác trực quan để tạo ra giao diện dựa vào các đối tượng như hộp hội thoại, button,... với nhiều thuộc tính định dạng.
- Đối với Lập trình sự kiện:
  - Cung cấp những đối tượng cho phép xây dựng chương trình theo hướng sự kiện (Event-Driven Programming).
  - Các đối tượng thiết kế giao diện đều được hỗ trợ các hàm xử lý sự kiện.
- Đối với Lập trình hướng đối tượng: C# là một ngôn ngữ hướng đối tượng hoàn toàn

Nguyễn Thị Mai Trang

7

7

## 2.3 Các bước xây dựng một ứng dụng bằng C#

- Nội dung:
  - Tạo project trong VS.Net, chọn ngôn ngữ C#
  - Cấu trúc một chương trình C#
  - Thiết kế giao diện
  - Viết code
  - Biên dịch, thực thi

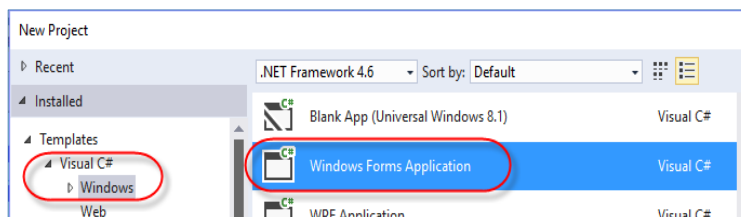
Nguyễn Thị Mai Trang

8

8

## Các bước xây dựng một ứng dụng bằng C# (tt)

- Tạo project ứng dụng Windows Form:



Nguyễn Thị Mai Trang

9

9

## Các bước xây dựng một ứng dụng bằng C# (tt)

- Cấu trúc một chương trình C#
  - Một chương trình C# có duy nhất một hàm Main, nằm trong file program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FirstProgram
{
    0 references
    static class Program
    {
        [STAThread]
        0 references
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

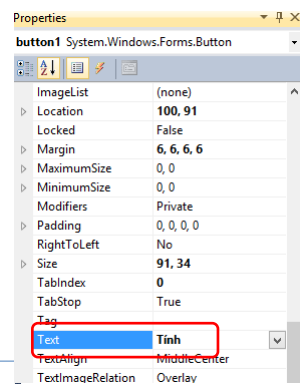
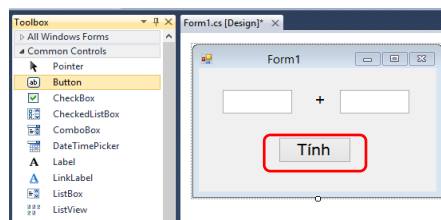
Nguyễn Thị Mai Trang

10

10

## Các bước xây dựng một ứng dụng bằng C# (tt)

- Thiết kế giao diện:
  - Mỗi project mới được tạo có một Form duy nhất
  - Drag một đối tượng trên thanh ToolBox kéo thả vào Form
  - Thay đổi thuộc tính đối tượng từ bảng Properties



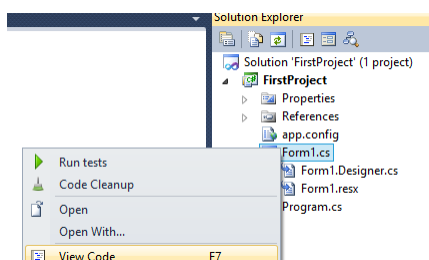
Nguyễn Thị Mai Trang

11

11

## Các bước xây dựng một ứng dụng bằng C# (tt)

- Viết code:
  - Chuyển sang phần code-behind (file.cs) bằng một trong các cách sau:
    - Click chuột phải trên tên Form trong cửa sổ Solution Explorer, chọn View Code
    - Nhấp double chuột trên Form
    - Nhấp double chuột trên một đối tượng trên Form



Nguyễn Thị Mai Trang

12

12

## Các bước xây dựng một ứng dụng bằng C# (tt)

```
using System.Windows.Forms;
```

Vùng khai báo các không gian tên (namespace)

```
namespace MyApplication
```

```
{
    public partial class Form1 : Form
```

```
{
    public Form1()
    {
        InitializeComponent();
    }
}
```

Hàm xử lý sự kiện click chuột trên Button có tên là button1

```
private void button1_Click(object sender, EventArgs e)
{
}
```

```
private void MyFunction()
{
}
```

Hàm người dùng tự định nghĩa

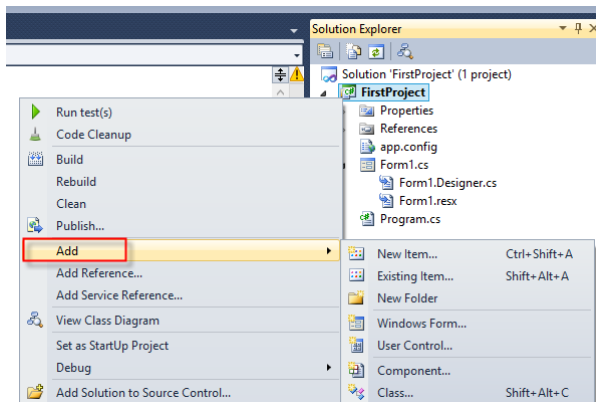
Nguyễn Thị Mai Trang

13

13

## Các bước xây dựng một ứng dụng bằng C# (tt)

- Thêm các thành phần vào project
  - Click phải trên tên project, chọn Add



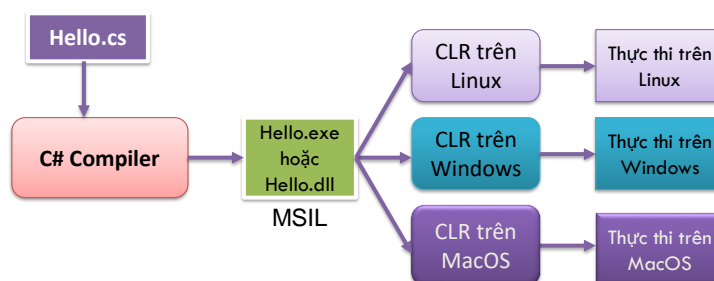
Nguyễn Thị Mai Trang

14

14

## Các bước xây dựng một ứng dụng bằng C# (tt)

- Biên dịch, thực thi
  - Mã nguồn C# (tập tin \*.cs) được biên dịch qua MSIL
    - MSIL: tập tin .exe hoặc .dll
  - MSIL được CLR thông dịch qua mã máy



Nguyễn Thị Mai Trang

15

15

## 2.4 Từ khóa trong C#

### Các từ khóa dành riêng (Reserved Keyword)

abstract	as	base	bool	break	byte	case
catch	char	checked	class	const	continue	decimal
default	delegate	do	double	else	enum	event
explicit	extern	false	finally	fixed	float	for
foreach	goto	if	implicit	in	int	interface
internal	is	lock	long	namespace	new	null
object	operator	out	override	params	private	protected
public	readonly	ref	return	sbyte	sealed	short
sizeof	stackalloc	static	string	struct	switch	this
throw	true	try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	void	volatile	while

### Các từ khóa dùng trong ngữ cảnh (Contextual Keyword)

add	alias	ascending	descending	dynamic	from	get
global	group	into	join	let	orderby	partial
remove	select	set				

Nguyễn Thị Mai Trang

16

16



## Từ khóa trong C# (tt)

- Các từ khóa đặc biệt:
  - namespace: là một gói những thực thể (các class) → tránh sự trùng lặp tên giữa các lớp, cho phép tổ chức mã nguồn khoa học và hợp lý.
  - using:
    - khai báo các namespace, làm cho chương trình gọn hơn, không cần phải viết từng namespace cho từng đối tượng.
    - Dùng trong code để tạo và hủy các đối tượng một cách tự động (phần cuối chương).

## Từ khóa trong C# (tt)

- Các từ khóa đặc biệt:
  - namespace: là một gói những thực thể (các class) → tránh sự trùng lặp tên giữa các lớp, cho phép tổ chức mã nguồn khoa học và hợp lý.
  - using:
    - khai báo các namespace, làm cho chương trình gọn hơn, không cần phải viết từng namespace cho từng đối tượng.
    - Dùng trong code để tạo và hủy các đối tượng một cách tự động (phần cuối chương).

## Từ khóa trong C# (tt)

- Các từ khóa đặc biệt:
  - using: ví dụ, thay vì phải viết:
    - `System.Windows.Form.Button bt = new System.Windows.Form.Button();`
  - Có thể viết:
    - `using System.Windows.Form;`
    - `//...`
    - `Button bt = new Button();`
  - static: khai báo các thành phần tĩnh như hàm, biến của một lớp  
 → truy xuất các thành phần này mà không cần tạo một đối tượng của lớp.

## 2.5. Các kiểu dữ liệu trong C#

- C# hỗ trợ hai kiểu dữ liệu:
  - Kiểu dữ liệu giá trị (value): gồm các kiểu dữ liệu xây dựng sẵn (Predefined types) như kiểu số, ký tự, luận lý, kiểu dữ liệu liệt kê, kiểu cấu trúc (struct)...
  - Kiểu dữ liệu tham chiếu (reference): các kiểu dữ liệu do người dùng định nghĩa: class, interface, delegate, array

## Các kiểu dữ liệu trong C# (tt)

### • Predefined types

Type	Description	Examples
object	The ultimate base type of all other types	object o = new Stack();
string	String type; a string is a sequence of Unicode characters	string s = "Hello";
sbyte	8-bit signed integral type	sbyte val = 12;
short	16-bit signed integral type	short val = 12;
int	32-bit signed integral type	int val = 12;
long	64-bit signed integral type	long val1 = 12; long val2 = 34L;
byte	8-bit unsigned integral type	byte val1 = 12; byte val2 = 34U;
ushort	16-bit unsigned integral type	ushort val1 = 12; ushort val2 = 34U;

Nguyễn Thị Mai Trang

21

21

## Các kiểu dữ liệu trong C# (tt)

### • Predefined types

Type	Description	Examples
uint	32-bit unsigned integral type	uint val1 = 12; uint val2 = 34U;
ulong	64-bit unsigned integral type	ulong val1 = 12; ulong val2 = 34U; ulong val3 = 56L; ulong val4 = 78UL;
float	Single-precision floating point type	float value = 1.23F;
double	Double-precision floating point type	double val1 = 1.23 double val2 = 4.56D;
bool	Boolean type; a bool value is either true or false	bool value = true;
char	Character type; a char value is a Unicode character	char value = 'h';
decimal	Precise decimal type with 28 significant digits	decimal value = 1.23M;

Nguyễn Thị Mai Trang

22

22

## Các kiểu dữ liệu trong C# (tt)

- Các ký tự đặc biệt trong C#
  - Ký tự '\': đặt trước để hiển thị các ký tự đặc biệt

Cách sử dụng	Ý nghĩa	Cách sử dụng	Ý nghĩa
\\	Ký tự \	\f	Form feed
\'	Ký tự '	\n	Dòng mới
\"	Ký tự "	\r	Carriage return
\?	Ký tự ?	\t	Tab ngang
\a	Tiếng beep	\v	Tab dọc
\b	Backspace		

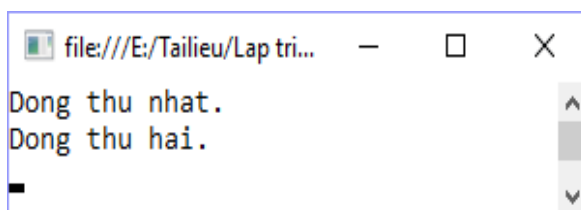
Nguyễn Thị Mai Trang

23

23

## Các kiểu dữ liệu trong C# (tt)

- Các ký tự đặc biệt trong C#
  - Ký tự '\': ví dụ
    - Console.WriteLine("Dong thu nhât.\nDong thu hai.");



```
file:///E:/Tailieu/Lap tri...
Dong thu nhât.
Dong thu hai.
```

Nguyễn Thị Mai Trang

24

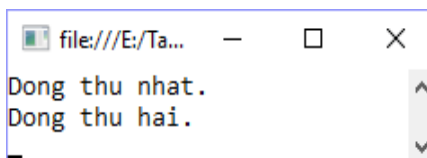
24

## Các kiểu dữ liệu trong C# (tt)

- Các ký tự đặc biệt trong C#

- Ký tự '@':

- Dùng đặt trước tên biến khi tên biến trùng với từ khóa trong C#. Ví dụ: `int @new = 10;`
- Dùng để thay thế ký tự '\' để cách viết tự nhiên hơn. Ví dụ:
- `string path = "D:\\Baitap";` → `string path = @"D:\Baitap";`
- `string literalTwo = "Dong thu nhât\nDong thu hai";`
- → `string literalTwo = @"Dong thu nhât  
Dong thu hai";`



Nguyễn Thị Mai Trang

25

25

## Kiểu dữ liệu liệt kê (Enumerations)

- Sử dụng một biến có thể lấy một giá trị trong một bộ giá trị cố định.
- Đặt tên chung cho một tập các giá trị nguyên (tương tự như tập các hằng), làm cho chương trình rõ ràng, dễ hiểu hơn.
- Ví dụ, để biểu diễn bốn hướng bắc, nam, đông, tây, ta có thể sử dụng bốn biến kiểu số nguyên như: `Bac=1`, `Nam=2`, `Dong=3`, `Tay=4`
- Các giá trị số trên không mang ý nghĩa liên quan đến các hướng và thường khó nhớ → kiểu dữ liệu enum.

Nguyễn Thị Mai Trang

26

26

## Kiểu dữ liệu liệt kê (tt)

- Cú pháp:

```
enum <TenKieuDulieu>
{
    <Giatri1> [= hằng],
    <Giatri2> [= hằng],
    <Giatri3> [= hằng],
    ...
    <GiatriN> [= hằng]
}
```

- Sử dụng::

```
– <TenKieuDulieu> <TenBien>;
  <TenBien> = <TenKieuDulieu>.<Giatri>;
```

Nguyễn Thị Mai Trang

27

27

## Kiểu dữ liệu liệt kê (tt)

```
enum Huong
```

```
{
```

```
    Bac = 1,
```

```
    Nam = 2,
```

```
    Dong = 3,
```

```
    Tay = 4
```

```
}
```

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

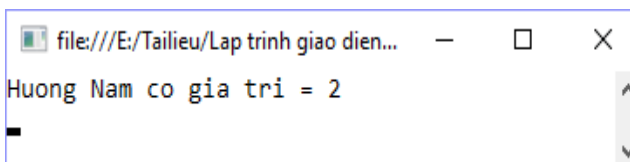
```
{
```

```
    Huong huong = Huong.Nam;
```

```
    Console.WriteLine("Huong {0} co gia tri = {1}",  
                      huong,(int)huong);
```

```
    Console.ReadLine();
```

```
}
```



Nguyễn Thị Mai Trang

28

28

## Kiểu dữ liệu cấu trúc (struct)

- Khai báo với từ khóa struct → tạo ra một kiểu dữ liệu có thể lưu trữ nhiều giá trị.
- struct là kiểu dữ liệu giá trị, được lưu trữ trên stack trong bộ nhớ, được dùng trong trường hợp lưu trữ các dữ liệu tương đối nhỏ.
- Cú pháp khai báo:
  - struct <TenCautruc>
  - {
  - <Khai báo các thành phần>
  - }

Nguyễn Thị Mai Trang

29

29

## Kiểu dữ liệu cấu trúc (tt)

- struct có thể có các trường, thuộc tính, phương thức, phương thức khởi tạo
- Một số đặc điểm của struct:
  - Không thể khai báo một phương thức khởi tạo không tham số cho một cấu trúc.
  - Các thuộc tính của cấu trúc đều phải được khởi tạo giá trị.
  - Khi khai báo các trường của cấu trúc, không thể khởi gán giá trị.
  - Trường hợp gọi phương thức khởi tạo của cấu trúc với từ khóa new, các thuộc tính chưa khởi trị sẽ được tự động gán trị 0 hoặc null.

Nguyễn Thị Mai Trang

30

30

## Kiểu dữ liệu cấu trúc (tt)

struct Thoigian //tên cấu trúc

```
{
    int gio, phut, giay;
    public Thoigian(int g, int p, int gi)
    {
        gio = g;
        phut = p;
        giay = gi;
    }
    public int Gio
    {
        get { return gio; }
        set { gio = value; }
    }
    public int Phut
    {
        get { return phut; }
        set { phut = value; }
    }
    public int Giay
    {
        get { return giay; }
        set { giay = value; }
    }
}
```

Nguyễn Thị Mai Trang

31

31

## Kiểu dữ liệu cấu trúc (tt)

```
public void Chuanhoa() {
    int t = giay / 60;
    giay = giay >= 60 ? giay % 60 : giay;
    phut = phut + t;
    t = phut / 60;
    phut = phut >= 60 ? phut % 60 : phut;
    gio = gio + t;
    gio = gio >= 24 ? gio % 24 : gio ;
}
public string HienthiThoigian() {
    Chuanhoa();
    return String.Format("{0:0} : {1:00} : {2:00}", gio, phut, giay);
}
}
```

Nguyễn Thị Mai Trang

32

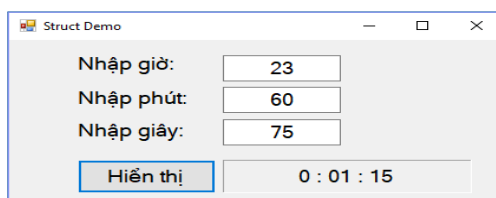
32



## Kiểu dữ liệu cấu trúc (tt)

- Sử dụng kiểu cấu trúc trong Form:

```
private void btHienthi_Click(object sender, EventArgs e)
{
    Thoigian t = new Thoigian (int.Parse(txtGio.Text),
                               int.Parse(txtPhut.Text),
                               int.Parse(txtGiay.Text));
    lbThoigian.Text = t.HienthiThoigian();
}
```



Nguyễn Thị Mai Trang

33

33

## 2.6 Biến, Hằng

- Biến trong C# được khai báo và sử dụng tương tự như C++, nhưng phải được khởi tạo trước khi sử dụng.
- Cú pháp khai báo biến:
  - [MucTruyCap] KieuDulieu TenBien [= Giatri] ;
  - MucTruyCap: từ khóa public, private, protected,... chỉ định mức truy cập của biến, mặc định là private.
  - KieuDulieu: kiểu dữ liệu.
  - TenBien: tên biến.
  - Giatri: giá trị gán cho biến.

```
int number;
private int sum = 0;
double radius = 5.0;
protected char myChar = 'A';
```

Nguyễn Thị Mai Trang

34

34

## Biến, Hằng (tt)

- Tầm vực của biến:
  - Biến khai báo bên trong phương thức thì có phạm vi trong phương thức đó, gọi là biến cục bộ, không thể truy xuất giữa các phương thức.
  - Biến khai báo bên trong thân của một lớp thì có phạm vi là lớp đó.
  - Trong một phạm vi hoạt động (scope), không thể có hai biến cùng tên

Nguyễn Thị Mai Trang

35

35

## Biến, Hằng (tt)

- Ví dụ về tầm vực của biến:

```
class Program
{
    static void Main(string[] args)
    {
        int j = 0;
        for (int i = 1; i <= 5; i++)
        {
            int j = i * 2; //lỗi
        }
    }
}
```

```
class Program
{
    int j = 0; //biến j thuộc lớp
    static void Main(string[] args)
    {
        for (int i = 1; i <= 5; i++)
        {
            int j = i * 2; //biến j là biến cục bộ
        }
    }
}
```

Nguyễn Thị Mai Trang

36

36

## Biến, Hằng (tt)

- Hằng tương tự biến nhưng giá trị của hằng không thay đổi khi chương trình thực thi.
- Hằng phải được khởi tạo khi khai báo và chỉ khai báo duy nhất một lần trong chương trình và không được thay đổi giá trị.
- Cú pháp khai báo hằng:
  - `<const> <type> <CONSTNAME> = <value>;`
- Ví dụ:
  - `const double PI = 3.14158;`
  - `public const int MAX = 100;`

Nguyễn Thị Mai Trang

37

37

## 2.7. Toán tử trong C#

- Toán tử gán: `=`
- Toán tử số học: `+`, `-`, `*`, `/`, `%` (chia lấy phần dư)
- Toán tử tăng, giảm: `++`, `--`, `+=`, `-=`, `*=`, `/=`, `%=`
- Toán tử quan hệ: `==`, `!=`, `>`, `>=`, `<`, `<=`
  - Lưu ý: phân biệt `=` và `==`
- Toán tử logic: `&&` (and), `||` (or), `!` (not)

Nguyễn Thị Mai Trang

38

38

## 2.8 Cấu trúc lựa chọn

- Cấu trúc if đơn

- Cú pháp:

- if (dieukien)

- Khoi\_lenh

- Lệnh trong Khoi\_lenh chỉ thực hiện khi dieukien có giá trị true

- Ví dụ:

```
if (diem >= 60)
    Console.WriteLine("Đậu");
```

Nguyễn Thị Mai Trang

39

39

## Cấu trúc lựa chọn (tt)

- Cấu trúc if..else

- Cú pháp:

- if (dieukien)

- Khoi\_lenh\_1

- else

- Khoi\_lenh\_2

- dieukien = true → lệnh trong Khoi\_lenh\_1

- Ngược lại, → lệnh trong Khoi\_lenh\_2.

- Ví dụ:

```
if (diem >= 60 )
    Console.WriteLine("Đậu");
else
    Console.WriteLine("Rớt");
```

Nguyễn Thị Mai Trang

40

40

## Cấu trúc lựa chọn (tt)

- Sử dụng biểu thức điều kiện
  - Sử dụng toán tử `?`, `:` thay thế cho `if ..else` trong những trường hợp đơn giản
  - Cú pháp:
 

```
dieukien ? giatri1 : giatri2;
```

    - `dieukien = true` → `giatri1`
    - `dieukien = false` → `giatri2`

```
Console.WriteLine( diem >= 5 ? "Đậu" : "Rớt" );
```

Nguyễn Thị Mai Trang

41

41

## Cấu trúc lựa chọn (tt)

- `if .. else` lồng nhau

```
if (diem >= 90)
    Console.WriteLine ("A");
else
    if (diem >= 80)
        Console.WriteLine ("B");
    else
        if (diem >= 70)
            Console.WriteLine ("C");
        else
            if (diem >= 60)
                Console.WriteLine ("D");
            else
                Console.WriteLine ("F");
```

```
if (diem >= 90)
    Console.WriteLine ("A");
else if (diem >= 80)
    Console.WriteLine ("B");
else if (diem >= 70)
    Console.WriteLine ("C");
else if (diem >= 60)
    Console.WriteLine ("D");
else
    Console.WriteLine ("F");
```

Nguyễn Thị Mai Trang

42

42

## Cấu trúc lựa chọn (tt)

- Lưu ý khi sử dụng if .. else lồng nhau:
  - Một else luôn kết hợp với if gần nó nhất
  - Nên sử dụng các cặp ngoặc { } trong trường hợp sử dụng if .. else lồng nhau

```
if ( x > 5 )
  if ( y > 5 )
    Console.WriteLine( "x và y đều > 5" );
  else
    Console.WriteLine( "x <= 5" );
```



```
if ( x > 5 )
{
  if ( y > 5 )
    Console.WriteLine( "x và y đều > 5" );
}
else
  Console.WriteLine( "x <= 5" );
```

Nguyễn Thị Mai Trang

43

## Cấu trúc lựa chọn (tt)

- Lưu ý (tt):
  - Trong một khối lệnh if hoặc else, nếu chứa từ hai câu lệnh, phải đặt trong cặp ngoặc { }

```
if ( diem >= 60 )
  Console.WriteLine( "Đậu" );
else
{
  Console.WriteLine( "Rớt" );
  Console.WriteLine( "Bạn phải học lại." )
}
```

Nguyễn Thị Mai Trang

44

44

## Cấu trúc lựa chọn (tt)

- Cấu trúc switch

```
switch (Bien_kiemtra)
{
    case <giatri_1>:
        //code thực hiện nếu Bien_kiemtra = giatri_1
        break;
    case <giatri_2>:
        //code thực hiện nếu Bien_kiemtra = giatri_2
        break;
    ...
    case <giatri_n>:
        //code thực hiện nếu Bien_kiemtra = giatri_n
        break;
    [default]:
        //code thực hiện nếu Bien_kiemtra khác các giá trị trên
        break;
}
```

Nguyễn Thị Mai Trang

45

45

## Cấu trúc lựa chọn (tt)

- Cấu trúc switch, ví dụ:

```
public double Tinhtoan (double a, double b, char pheptoa)
{
    double ketqua = 0;
    switch(pheptoa)
    {
        case '+':
            ketqua = a + b; break;
        case '-':
            ketqua = a - b; break;
        case '*':
            ketqua = a * b; break;
        case '/':
            if(b!=0) ketqua = a / b;
            break;
    }
    return ketqua;
}
```

Nguyễn Thị Mai Trang

46

## 2.9 Cấu trúc lặp

- Cấu trúc lặp for
  - Cú pháp:  

```
for (bien_khoi_tao; dieukien ; buoc_lap)
  Khoi_lenh
```
  - Ví dụ, in ra màn hình các số từ 1 đến 5:  

```
for (int i = 1; i <= 5; i++)
  Console.WriteLine("{0}", i);
```

Nguyễn Thị Mai Trang

47

47

## Cấu trúc lặp (tt)

- Cấu trúc lặp while:
  - Cú pháp:  

```
while (dieukien)
{
  Khoi_lenh
}
```
  - dieukien được kiểm tra trước
    - true: thực hiện lệnh trong Khoi\_lenh
    - false: thoát khỏi vòng lặp

Nguyễn Thị Mai Trang

48

48



## Cấu trúc lặp (tt)

- Cấu trúc lặp while:
  - Ví dụ, tính tổng các số từ 1 đến 5

```
int sum = 0;
int i = 1;
while (i <= 5)
{
    sum += i;
    i++;
}
```

Nguyễn Thị Mai Trang

49

49

## Cấu trúc lặp (tt)

- Cấu trúc lặp do .. while
  - Cú pháp:
 

```
do
{
    Khai_lenh
}while (dieukien);
```

    - Thực hiện lệnh trong Khai\_lenh
    - Kiểm tra dieukien
      - true: thực hiện bước lặp tiếp theo
      - false: thoát khỏi vòng lặp

Nguyễn Thị Mai Trang

50

50

## Cấu trúc lặp (tt)

- Cấu trúc lặp do .. while

– Ví dụ, in ra màn hình các số từ 1 đến 10

```
int i = 1;
do
{
    Console.Write("{0}", i);
    i++;
} while (i <= 10);
```

Nguyễn Thị Mai Trang

51

51

## Cấu trúc lặp (tt)

- Cấu trúc lặp foreach:

- Dùng để duyệt các phần tử của một mảng, tập hợp.
- Không có biến đếm mà sử dụng một biến để đại diện cho giá trị từng phần tử.
- Cú pháp:
 

```
foreach (kieu_dulieu bien_daidien in ten_taphop)
{
    Khoi_lenh
}
```

  - kieu\_dulieu: kiểu dữ liệu.
  - bien\_daidien: biến đại diện cho mỗi phần tử trong mảng
  - in: từ khóa.
  - ten\_taphop: tên tập hợp / mảng.

Nguyễn Thị Mai Trang

52

52

## Cấu trúc lặp (tt)

- Cấu trúc lặp foreach:

– Ví dụ, tính tổng các phần tử trong mảng.

```
int [ ] arrInt = {2 , 4 , 5 , 7 , 8};
int sum = 0;
foreach (int number in arrInt)
    sum += number;
```

Nguyễn Thị Mai Trang

53

53

## Cấu trúc lặp (tt)

- Lệnh break, continue, return

– break: được sử dụng trong hai trường hợp:

- Trong cấu trúc switch: thoát khỏi cấu trúc switch.
- Trong vòng lặp: thoát khỏi vòng lặp trực tiếp chứa nó.

– continue: ngừng thực hiện các công việc còn lại của vòng lặp hiện thời và quay về đầu vòng lặp để thực hiện bước lặp tiếp theo

– return: thoát khỏi một hàm, trả quyền điều khiển về phía triệu gọi hàm (caller)

Nguyễn Thị Mai Trang

54

54

## 2.10 Xử lý ngoại lệ

- Ngoại lệ là một lỗi phát sinh không mong muốn trong quá trình thực thi một chương trình, còn gọi là lỗi.
- Lỗi xảy ra trong code hoặc một lời gọi hàm từ code.
- Có nhiều nguyên nhân sinh ra lỗi chương trình như thiếu bộ nhớ, thiếu tài nguyên, thao tác của người sử dụng,...

Ví dụ:

- lỗi chia cho 0
- lỗi sai định dạng
- lỗi truy xuất một tập tin không tồn tại
- ...

Nguyễn Thị Mai Trang

55

55

## Xử lý ngoại lệ (tt)

- CLR hỗ trợ cơ chế xử lý lỗi một cách tự động.
- Khi có lỗi phát sinh: chương trình hiển thị một thông báo từ trình biên dịch và được yêu cầu đóng.
- → gây khó chịu cho người sử dụng, không đảm bảo tính tin cậy của chương trình phần mềm.
- Có thể ngăn chặn hoặc xử lý lỗi bằng cách nắm bắt và xử lý các ngoại lệ.
- Thư viện .NET framework cung cấp các lớp, đối tượng xử lý ngoại lệ.

Nguyễn Thị Mai Trang

56

56

## Xử lý ngoại lệ (tt)

- Xử lý ngoại lệ với try...catch...finally

– Cú pháp:

```
try{
    // các lệnh thực hiện
}
catch ([Exception]){
    // các lệnh xử lý lỗi
}
finally{
    //các lệnh kết thúc xử lý
}
```

Nguyễn Thị Mai Trang

57

57

## Xử lý ngoại lệ (tt)

- try..catch..finally (tt):

- Các lệnh dễ sinh ra lỗi đặt trong khối try { }.
- Các lệnh xử lý lỗi đặt trong khối catch { }, có thể sử dụng nhiều khối catch để bắt nhiều ngoại lệ.
- Các lệnh trong khối finally { } luôn được thực hiện.
- Khi lỗi phát sinh, các lệnh tiếp theo trong khối try không được thực hiện mà chuyển sang các lệnh xử lý trong khối catch.
- Để bắt giữ ngoại lệ, trong khối catch ta sử dụng các đối tượng Exception.

Nguyễn Thị Mai Trang

58

58

## Xử lý ngoại lệ (tt)

- try..catch..finally, ví dụ:

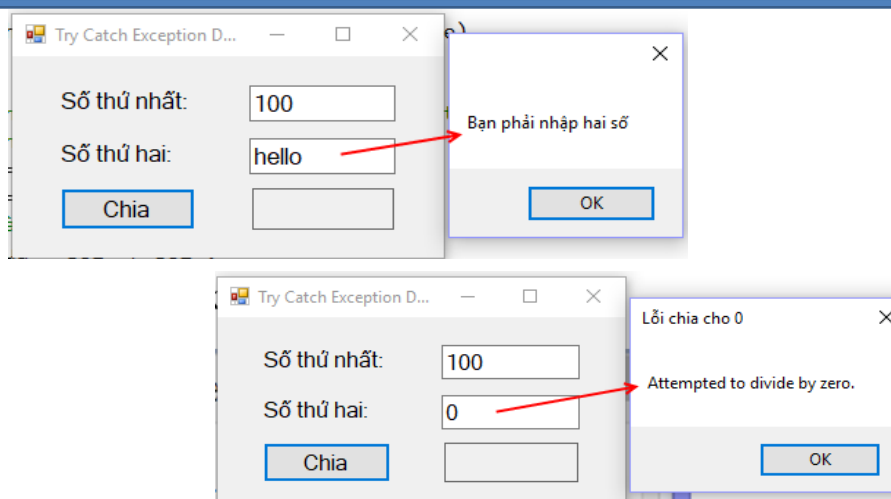
```
try { /*hai lệnh bên dưới sẽ sinh ra lỗi nếu giá trị
nhập trên hai textbox không phải là số*/
    int so1 = int.Parse(txtSo1.Text);
    int so2 = int.Parse(txtSo2.Text);
    int ketqua = so1 / so2 ; //có thể sinh ra lỗi chia cho 0
    lbKetqua.Text = ketqua.ToString();
}
catch (FormatException) {
    MessageBox.Show("Bạn phải nhập hai số");
}
catch (DivideByZeroException ex) {
    MessageBox.Show(ex.Message,"Lỗi chia cho 0");
}
```

Nguyễn Thị Mai Trang

59

59

## Xử lý ngoại lệ (tt)



Nguyễn Thị Mai Trang

60

60

## Xử lý ngoại lệ (tt)

- Sử dụng lệnh throw:
  - Lệnh throw được dùng để ném ra một tín hiệu về sự bất bình thường làm phát sinh một ngoại lệ.
  - Khi ngoại lệ phát sinh, việc thực thi trong khi CLR sẽ được dừng và tìm kiếm một trình xử lý ngoại lệ
  - Nếu không tìm được một trình xử lý ngoại lệ nào trong chương trình thì chương trình sẽ kết thúc

Nguyễn Thị Mai Trang

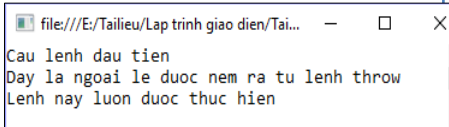
61

61

## Xử lý ngoại lệ (tt)

- throw, ví dụ:

```
static void Main(string[] args){
    try {
        Console.WriteLine("Cau lenh dau tien");
        throw new Exception("Day la ngoai le duoc nem ra tu lenh throw");
        Console.WriteLine("Lenh nay khong bao gio thuc hien");
    }
    catch (Exception ex) {
        Console.WriteLine(ex.Message);
    }
    finally {
        Console.WriteLine("Lenh nay luon duoc thuc hien");
    }
    Console.ReadLine();
}
```



```
file:///E:/Tai lieu/Lap trinh giao dien/Tai...  -  □  X
Cau lenh dau tien
Day la ngoai le duoc nem ra tu lenh throw
Lenh nay luon duoc thuc hien
```

Nguyễn Thị Mai Trang

62

62

## Xử lý ngoại lệ (tt)

- Sử dụng checked và unchecked
  - Xử lý lỗi "tràn số".
  - Lệnh checked: kiểm tra và làm phát sinh ngoại lệ `OverflowException` khi có lỗi tràn số.

```
try {
    int a = 999999;
    int b = 999999;
    int tích = checked (a * b);
    Console.WriteLine("{0} * {1} = {2}", a, b, tích);
}
catch (OverflowException ex) {
    Console.WriteLine(ex.Message);
}
```

Nguyễn Thị Mai Trang

63

63

## Xử lý ngoại lệ (tt)

- Sử dụng checked và unchecked
  - Lệnh unchecked: hủy bỏ kiểm tra, không làm phát sinh ngoại lệ `OverflowException` khi có lỗi tràn số

```
int number = int.MaxValue;
unchecked
{
    number++;
    Console.WriteLine("Chúng tôi sẽ nghiên cứu lỗi này!");
}
```

Nguyễn Thị Mai Trang

64

64



## Xử lý ngoại lệ (tt)

- Phát biểu using: được sử dụng trong code nhằm tạo các đối tượng tự hủy một cách an toàn.

Ví dụ:

```
using (StreamReader reader = new StreamReader("info.txt"))
{
    string row;
    while ((row = reader.ReadLine()) != null)
        Console.WriteLine(row);
}
```

Nguyễn Thị Mai Trang

65

65

## Xử lý ngoại lệ (tt)

- Các lớp ngoại lệ thường dùng

Tên lớp ngoại lệ	Ý nghĩa
MethodAccessException	Lỗi truy cập đến các thành phần (phương thức,...) không được phép truy cập
ArrayTypeMismatchException	Kiểu mảng không phù hợp
ArithmeticException	Lỗi liên quan đến các phép toán
DivideByZeroException	Lỗi chia cho 0
FormatException	Lỗi sai định dạng một kiểu dữ liệu
IndexOutOfRangeException	Lỗi truy xuất ngoài chỉ số của mảng
InvalidCastException	Phép gán không hợp lệ
NullReferenceException	Tham chiếu đến một đối tượng null
OutOfMemoryException	Tràn bộ nhớ
OverflowException	Tràn phép toán

Nguyễn Thị Mai Trang

66

66