

LẬP TRÌNH GIAO DIỆN

Nguyễn Thị Phương Trang

Chương 2

Ngôn ngữ lập trình C#

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

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

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.

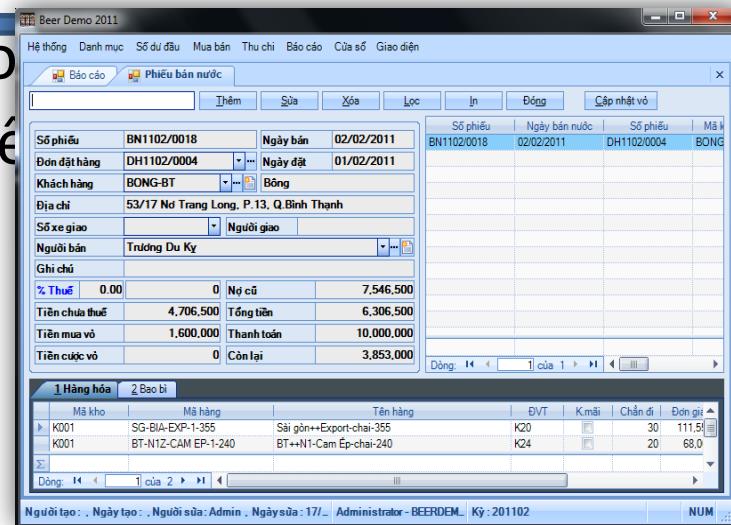
Giới thiệu ngôn ngữ lập trình C# (tt)

- 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,...

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

- Gói phần mềm ứng dụng phục vụ cho cơ quan, doanh nghiệp: phần mềm kế toán, quản lý nhân sự, quản lý mua, bán hàng hóa....



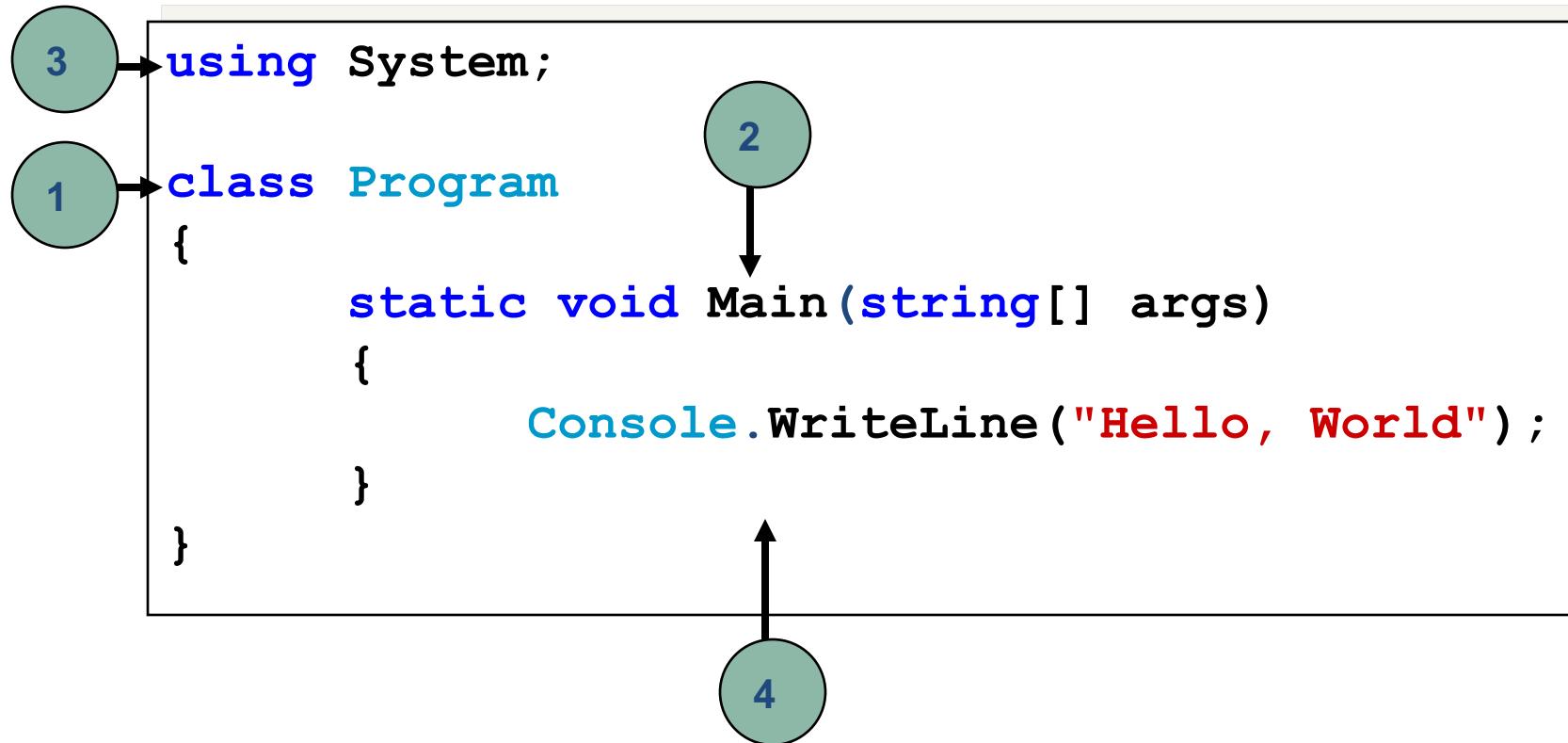
- Trò chơi, web



- Ứng dụng cho thiết bị di động



Chương trình C# đầu tiên



Lớp

- Một ứng dụng C# gồm tập các class và struct
- Một lớp gồm tập dữ liệu và phương thức
- Cú pháp

```
class ClassName
{
    ...
}
```

Phương thức Main

- Phương thức **Main** được định nghĩa trong lớp
- Chú ý khi viết hàm **Main**
 - Ký tự **M** phải viết HOA, “**Main**”
 - Phải có một hàm **Main** là entry point của chương trình
 - Khai báo **Main**: **static void Main**
- Khi hàm **Main** kết thúc hay gặp lệnh return thì ứng dụng kết thúc

Dùng Directive và System namespace

- .NET Framework cung cấp nhiều lớp tiện ích
 - Các lớp được tổ chức thành các namespace
- **System** là namespace được dùng thông dụng nhất
- Khi sử dụng lớp phải chỉ rõ lớp đó thuộc namespace nào

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

■ Dùng directive

```
using System;  
...  
Console.WriteLine("Hello, World");
```

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

Xuất dữ liệu

– Cú pháp 2:

```
void Console.WriteLine(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

Ví dụ:

```
Console.WriteLine("Tổng của {0}+{1} ={2} ", a, b, tong);  
Console.WriteLine("{0}\n{1}", "Welcome to ", "C# Programming!");  
Console.WriteLine(String.Format("Prime numbers less than 10: {0},  
{1}, {2}, {3}", 2, 3, 5, 7));
```

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

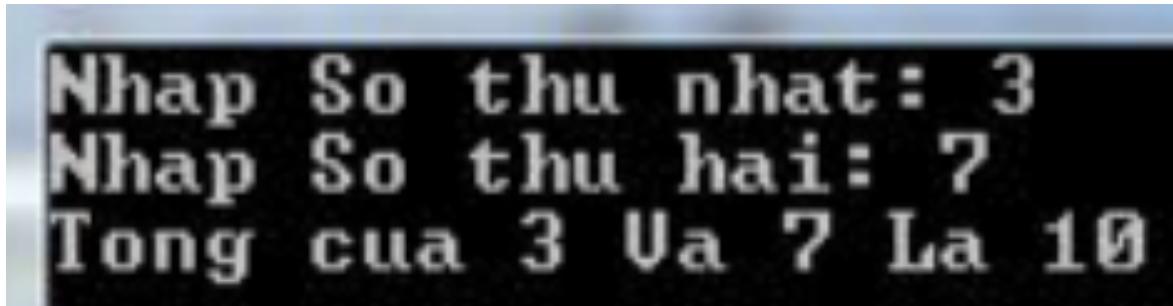
```
Kieu.Parse("chuoi");
```

Ví dụ:

```
string s = "123";
int data = int.Parse(s);
int nValue=System.Int32.Parse("113");
double dValue = System.Double.Parse("3 . 5");
```

Bài tập

1. Viết chương trình xuất ra yêu cầu nhập vào hai số từ bàn phím xuất ra kết quả tổng của hai số vừa nhập.



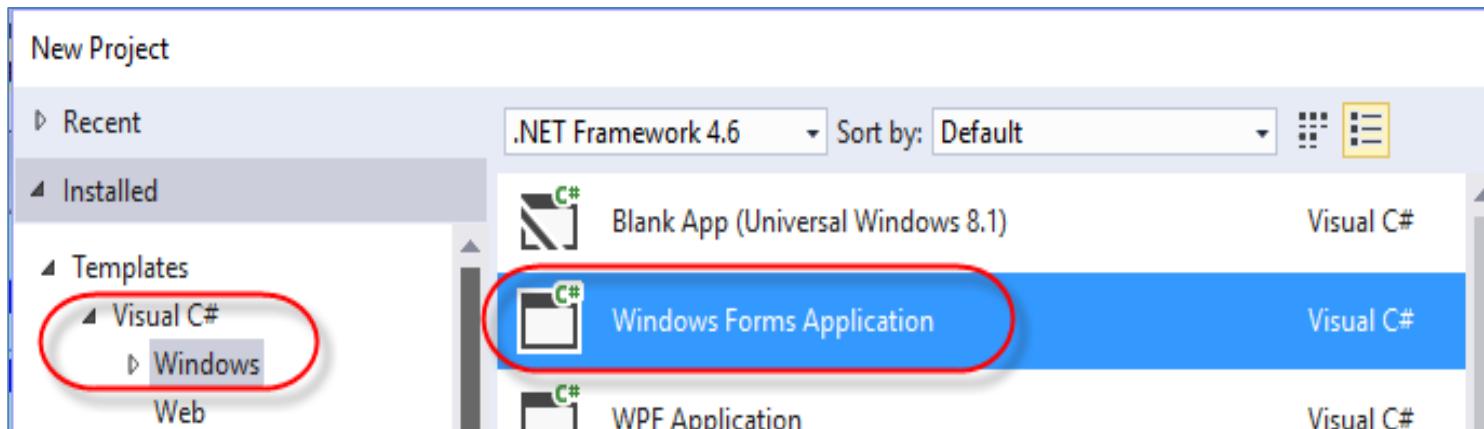
```
Nhap So thu nhat: 3
Nhap So thu hai: 7
Tong cua 3 Va 7 La 10
```

2.3 Các bước xây dựng một ứng dụng winform 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

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

- Tạo project ứng dụng Windows Form:



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

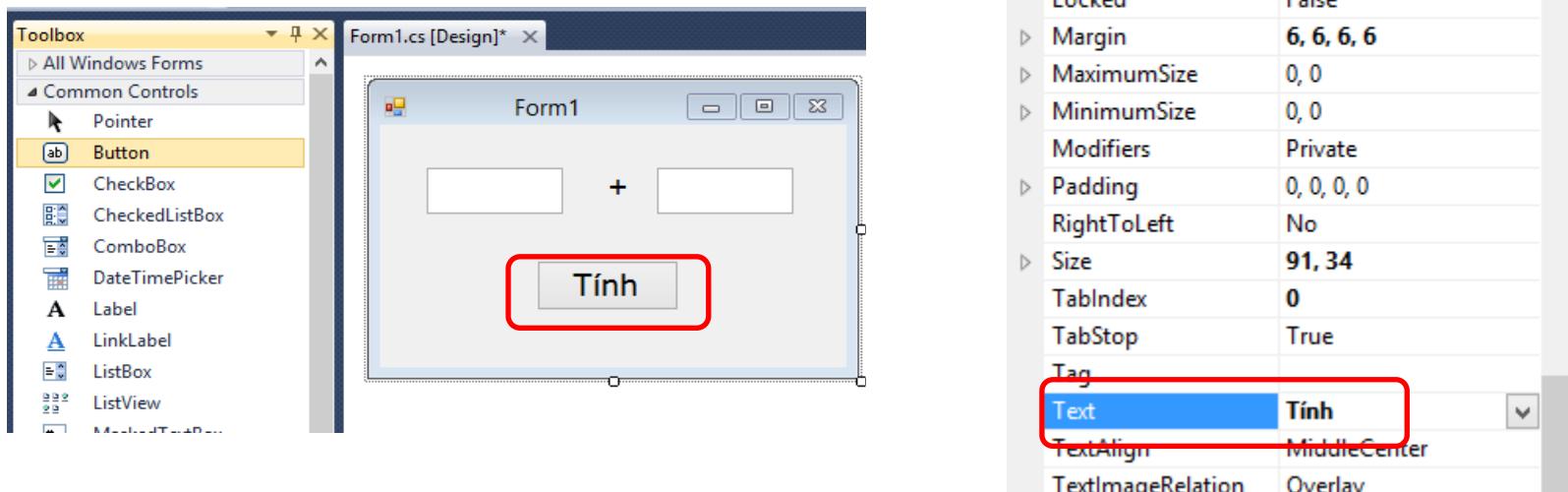
- Cấu trúc một chương trình C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace FirstProgram
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

The code block shows the structure of a simple C# Windows application. It includes basic using statements for System, System.Collections.Generic, System.Linq, System.Threading.Tasks, and System.Windows.Forms. The application is named 'FirstProgram' and contains a single static class 'Program'. The 'Main' method is highlighted with a red rounded rectangle, indicating it is the entry point of the application. Inside 'Main', three lines of code initialize the application's visual style, set text rendering defaults, and run a new instance of 'Form1'.

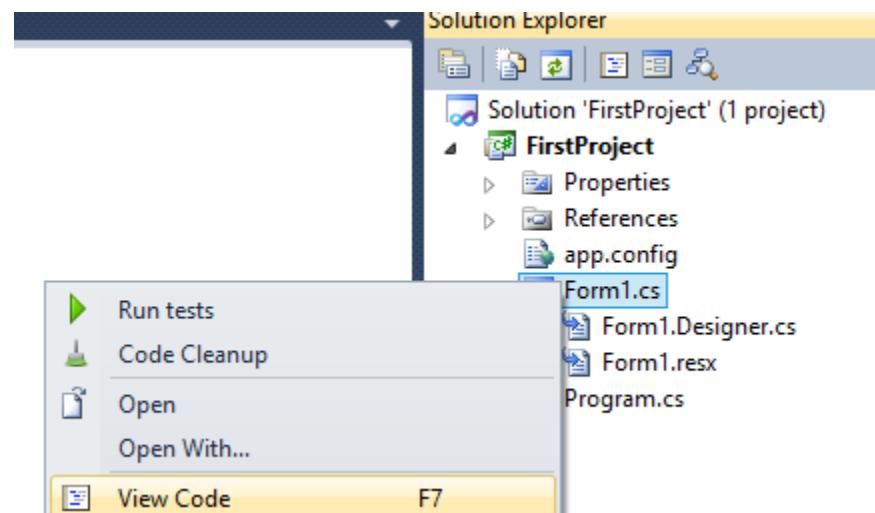
Các bước xây dựng một ứng dụng bằng C# (tt)

- 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



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



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

```
using System.Windows.Forms;  
  
namespace MyApplication  
{  
    public partial class Form1 : Form  
    {  
        public Form1()  
        {  
            InitializeComponent();  
        }  
  
        private void button1_Click(object sender, EventArgs e)  
        {  
        }  
  
        private void MyFunction()  
        {  
        }  
    }  
}
```

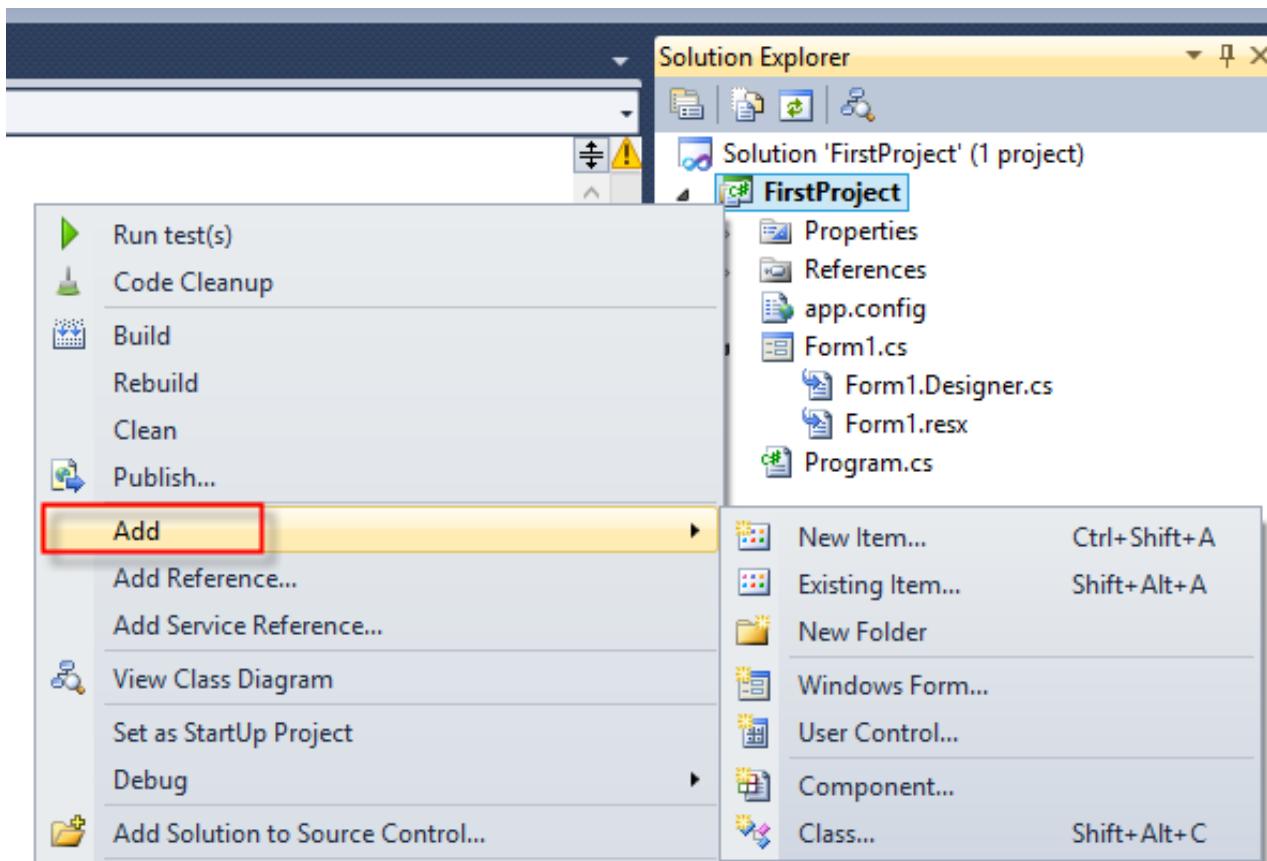
Vùng khai báo các không gian tên (namespace)

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

Hàm người dùng tự định nghĩa

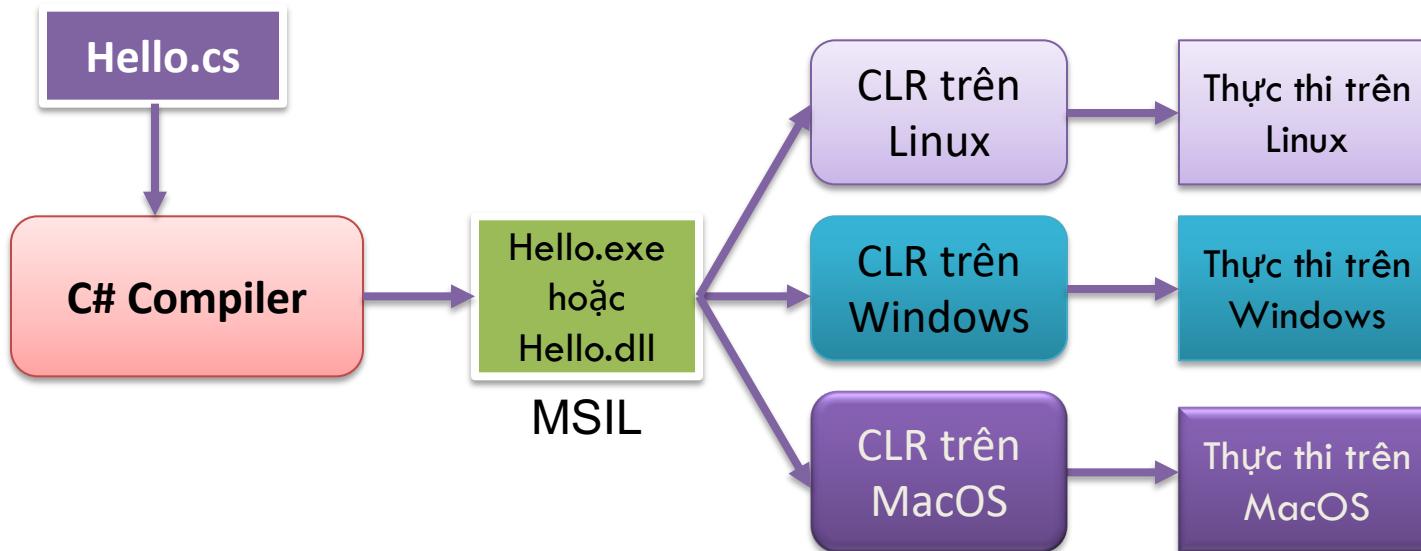
Các bước xây dựng một ứng dụng bằng C# (tt)

- Thêm các thành phần vào project



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

- Biên dịch, thực thi



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				

Từ khóa trong C# (tt)

- Các từ khóa đặc biệt:
 - namespace
 - using
 - static

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;

Các kiểu dữ liệu trong C# (tt)

- Predefined types

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

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		

Các kiểu dữ liệu trong C# (tt)

- Các ký tự đặc biệt trong C#
 - Ký tự ‘\’
 - Ký tự ‘@’:

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.

Enum

- Định nghĩa kiểu enum
 - Cú pháp

```
enum Name { value1, value2, ... }
```

- Ví dụ

```
enum Color { Red, Green, Blue }
```

- Mặc nhiên, phần tử đầu tiên trong enum được gán giá trị 0 và các phần tử sau có giá trị tăng dần lên 1 (Red=0, Green=1, Blue=2)

Enum

- Giá trị mặc nhiên có thể thay đổi (như ví dụ sau)

```
public enum WeekDays
{
    Monday=1,
    Tuesday=2,
    Wednesday=3,
    Thursday=4,
    Friday=5
}
```

Enum

- Dùng kiểu enum
 - Ví dụ

```
Color color;  
color = Color.Red;
```

- hay

```
color = (Color) 0;
```

- Hiện giá trị enum
 - Ví dụ

```
Console.WriteLine("{0}", color);
```

Enum

- Lặp qua các giá trị của enum

```
foreach (Color color in Enum.GetValues(typeof(Color)))
{
    ...
}
```

Kiểu dữ liệu cấu trúc (struct)

- Struct: Struct là một cấu trúc dữ liệu có thể chứa các thành viên dữ liệu và các phương thức. Struct được coi như là một phiên bản lightweight của class
- Đặc điểm của struct
 - Struct là một kiểu giá trị
 - Không thể thừa kế
 - Constructor nếu khai báo rõ ràng thì constructor đó phải có ít nhất 1 tham số
 - Các field không được khởi tạo khi khai báo (khởi tạo trong constructor hay code sử dụng struct)

Struct

- Cú pháp

```
[attributes] [modifiers] struct <StructName>[:interfaces]
{
    [struct-body]
} ;
```

Struct – Ví dụ

```
struct RGB
{
    public int Red;
    public int Green;
    public int Blue;
}
```

```
RGB rgb;
rgb.Red = 0xFF;
rgb.Green = 0xFF;
rgb.Blue = 0xFF;
```

Struct

```
struct Point
{
    public int x, y;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}

class Test
{
    static void Main()
    {
        Point[] points = new Point[100];
        for (int i = 0; i < 100; i++)
            points[i] = new Point(i, i);
    }
}
```

- Một cấu trúc có thể có các thành phần sau:
 - Trường dữ liệu;
 - Phương thức;
 - Phương thức khởi tạo;

```
struct Time
{
    int hour, minute, second; ← - - - - - Trường dữ liệu
    public Time(int h, int m, int s)
    {
        hour = h;
        minute = m;
        second = s;
    }
    public int Hour
    {
        get { return hour; }
        set { hour = value; }
    }
    public int Minute
    {
        get { return minute; }
        set { minute = value; }
    }
    public int Second
    {
        get { return second; }
        set { second = value; }
    }
    public string Showtime()
    {
        return String.Format("{0:00}:{1:00}:{2:00}",hour,minute,second);
    }
    public void Normalize()
    {
        int t = second / 60;
        second = second >= 60 ? second % 60 : second;
        t = (minute + t) / 60;
        minute = (minute + t) >= 60 ? (minute + t) % 60 : (minute + t);
        hour = (hour + t) > 23 ? (hour + t) % 23 : (hour + t);
    }
}
```

Trường dữ liệu

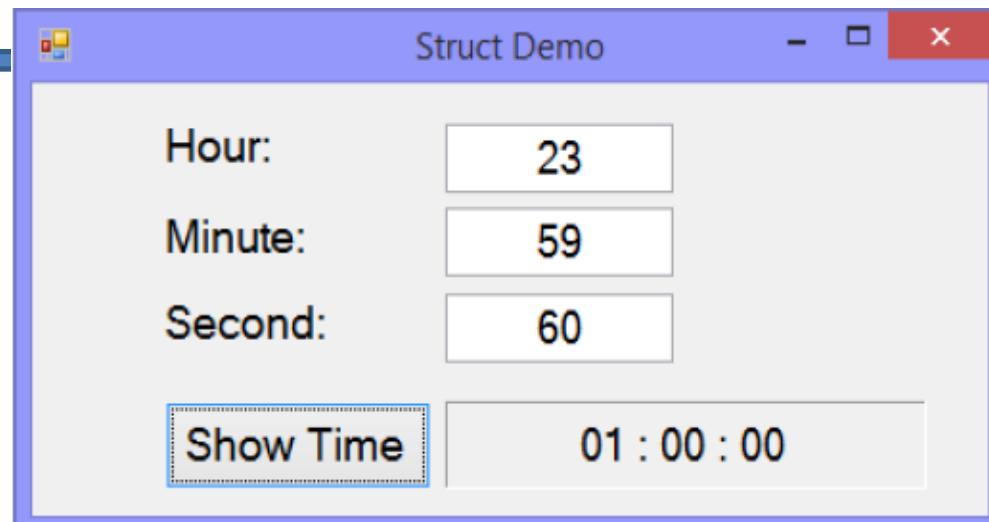
Phương thức khởi tạo

Các thuộc tính

Các phương thức
Thành viên

Struct

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    private void btShowTime_Click(object sender, EventArgs e)
    {
        Time t = new Time(int.Parse(txtHour.Text),
                           int.Parse(txtMinute.Text),
                           int.Parse(txtSecond.Text));
        t.Normalize();
        lbTime.Text = t.Showtime();
    }
}
```



Struct

- Chú ý
 - Phương thức khởi tạo của Struct phải có tham số.
 - Các thuộc tính của Struct đều phải được khởi tạo.

2.6 Biến, Hằng

- 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.

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

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

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 MAX = 100;`

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)

2.8 Cấu trúc lựa chọn

- Cấu trúc if đơn
 - Cú pháp:
if (dieukien)
 Khoi_lenh

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

Cấu trúc lựa chọn (tt)

- Cấu trúc if..else

- Cú pháp:

```
if (dieukien)
    Khoi_lenh_1
else
    Khoi_lenh_2
```

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

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;

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

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");
```

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
 - Trong một khối lệnh if hoặc else, nếu chứa từ hai câu lệnh, phải đặt trong cặp ngoặc { }

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;  
}
```

2.9 Cấu trúc lặp

- Cấu trúc lặp for
 - Cú pháp:
 - for (bien_khoi_tao; dieukien ; buoc_lap)
Khoi_lenh

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

Cấu trúc lặp (tt)

- Cấu trúc lặp do .. while

- Cú pháp:

- do

- {

- Khoi_lenh

- }while (dieukien);

- Thực hiện lệnh trong Khoi_lenh

- Kiểm tra dieukien

- true: thực hiện bước lặp tiếp theo

- false: thoát khỏi vòng lặp

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.

Lệnh break, continue, return

➤break:

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

➤continue:

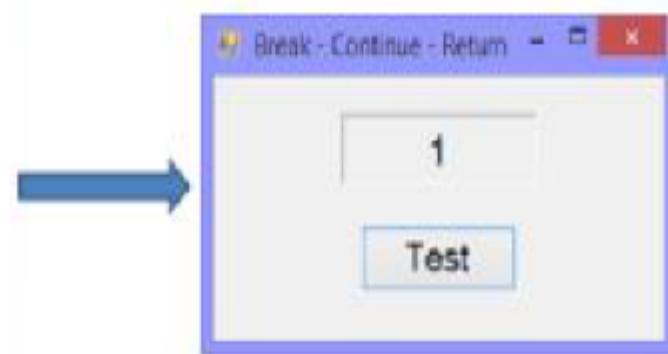
➤Không thực hiện tiếp các câu lệnh mà quay về đầu vòng lặp để thực hiện tiếp.

➤return:

➤Thoát khỏi hàm

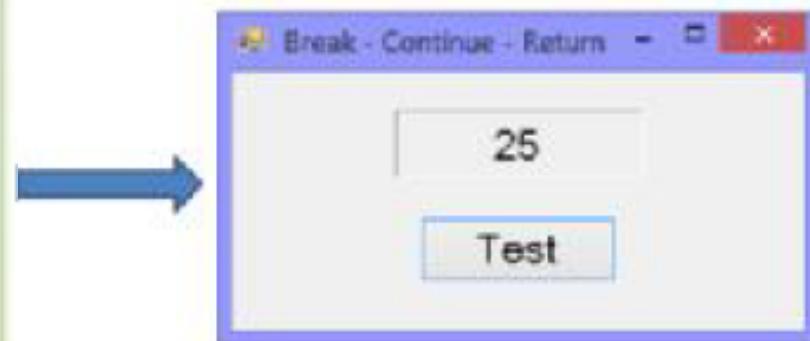
Lệnh break

```
public void Tinh tong1()
{
    int sum = 0;
    for (int i = 1; i <= 10; i++)
    {
        if (i % 2 == 0) break;
        sum += i;
    }
    label1.Text = sum.ToString();
}
```



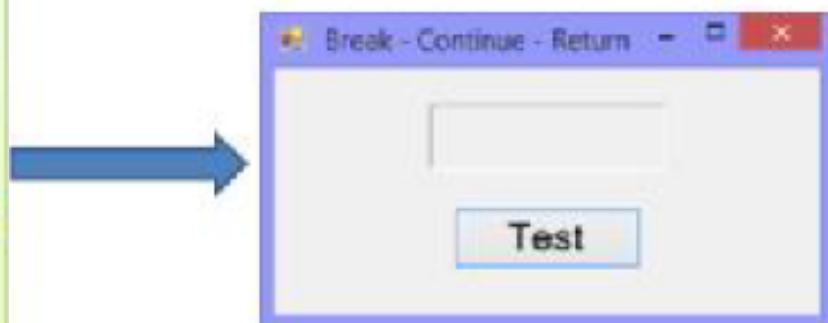
Lệnh continue

```
public void Tinh tong2()
{
    int sum = 0;
    for (int i = 1; i <= 10; i++)
    {
        if (i % 2 == 0) continue;
        sum += i;
    }
    label1.Text = sum.ToString();
}
```



Lệnh return

```
public void TinhTong3()
{
    int sum = 0;
    for (int i = 1; i <= 10; i++)
    {
        if (i % 2 == 0) return;
        sum += i;
    }
    label1.Text = sum.ToString();
}
```



Xử lý ngoại lệ

- Ngoại lệ (exception) là một “phát sinh” không bình thường trong quá trình thực thi một chương trình, còn gọi là lỗi.
- Chương trình dù đã không còn bug hay error vẫn có thể cho ra các exception (truy cập , bộ nhớ, thao tác người dùng)
- Có thể dùng các đối tượng exception có sẵn, tự tạo exception, hay bắt exception trong exception (trong trường hợp sửa lỗi)

Xử lý ngoại lệ - Try .. Catch.. finally

- Cú pháp:

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

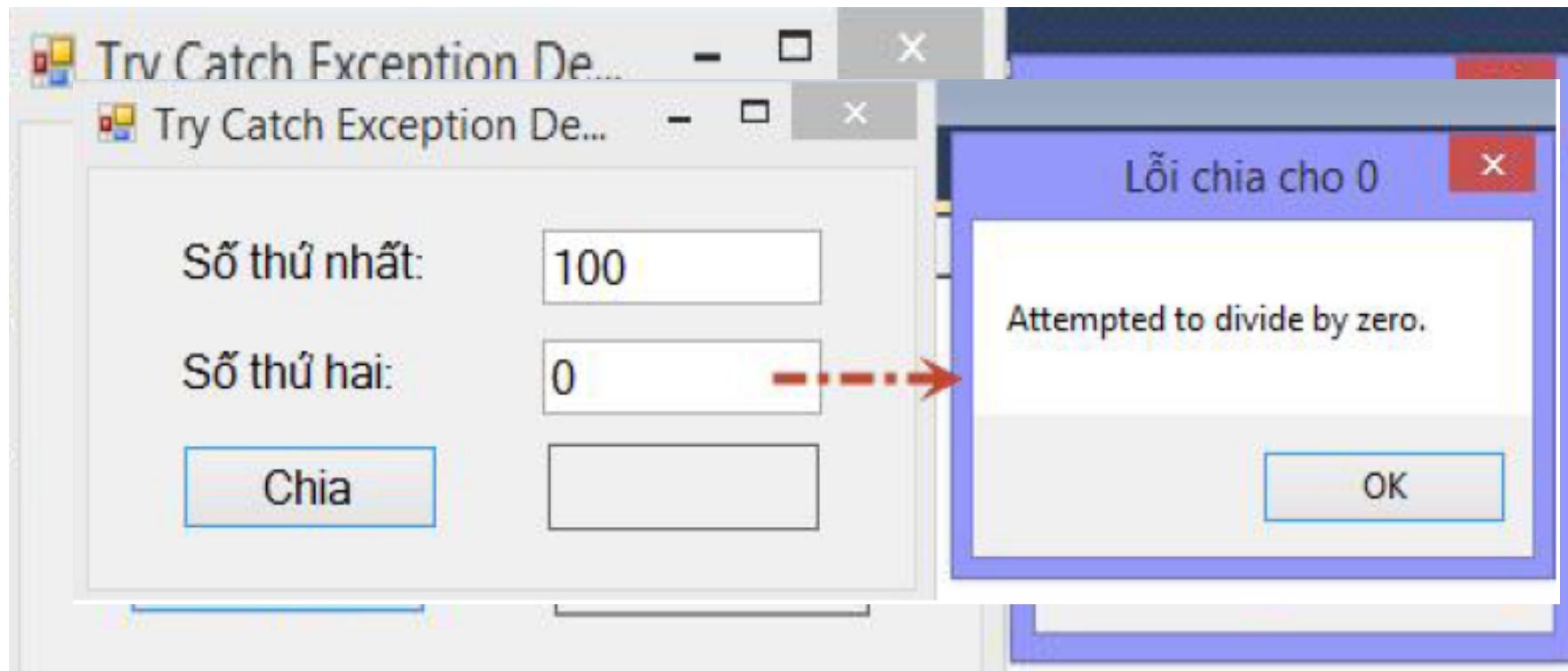
Xử lý ngoại lệ - Try .. Catch.. finaly

- Các lệnh dễ sinh lỗi đặt trong khối try{}.
- Các lệnh xử lý lỗi đặt trong khối catch{}
- Các lệnh trong khối finally{} luôn luôn được thực hiện trong cả 2 trường hợp có hay không có phát sinh lỗi.
- Khi có lỗi phát sinh, các lệnh tiếp theo trong khối try không được thực hiện, mà chuyển sang các lệnh trong khối catch.
- Để bắt ngoại lệ, sử dụng đối tượng exception

Xử lý ngoại lệ - Try .. Catch.. finally

```
try
{
    int so1, so2;
    double kq;
    so1 = int.Parse(txtSo1.Text);
    so2 = int.Parse(txtSo2.Text); ← Lỗi nếu nhập vào khác
    kq = so1 / so2; ← giá trị số
    lbKetQua.Text = kq.ToString(); ← Lỗi nếu số 2 bằng 0
}
catch (FormatException)
{
    MessageBox.Show("Bạn phải nhập hai số");
}
catch (DivideByZeroException ex)
{
    MessageBox.Show(ex.Message, "Lỗi chia cho 0");
}
```

Xử lý ngoại lệ - Try .. Catch.. finally



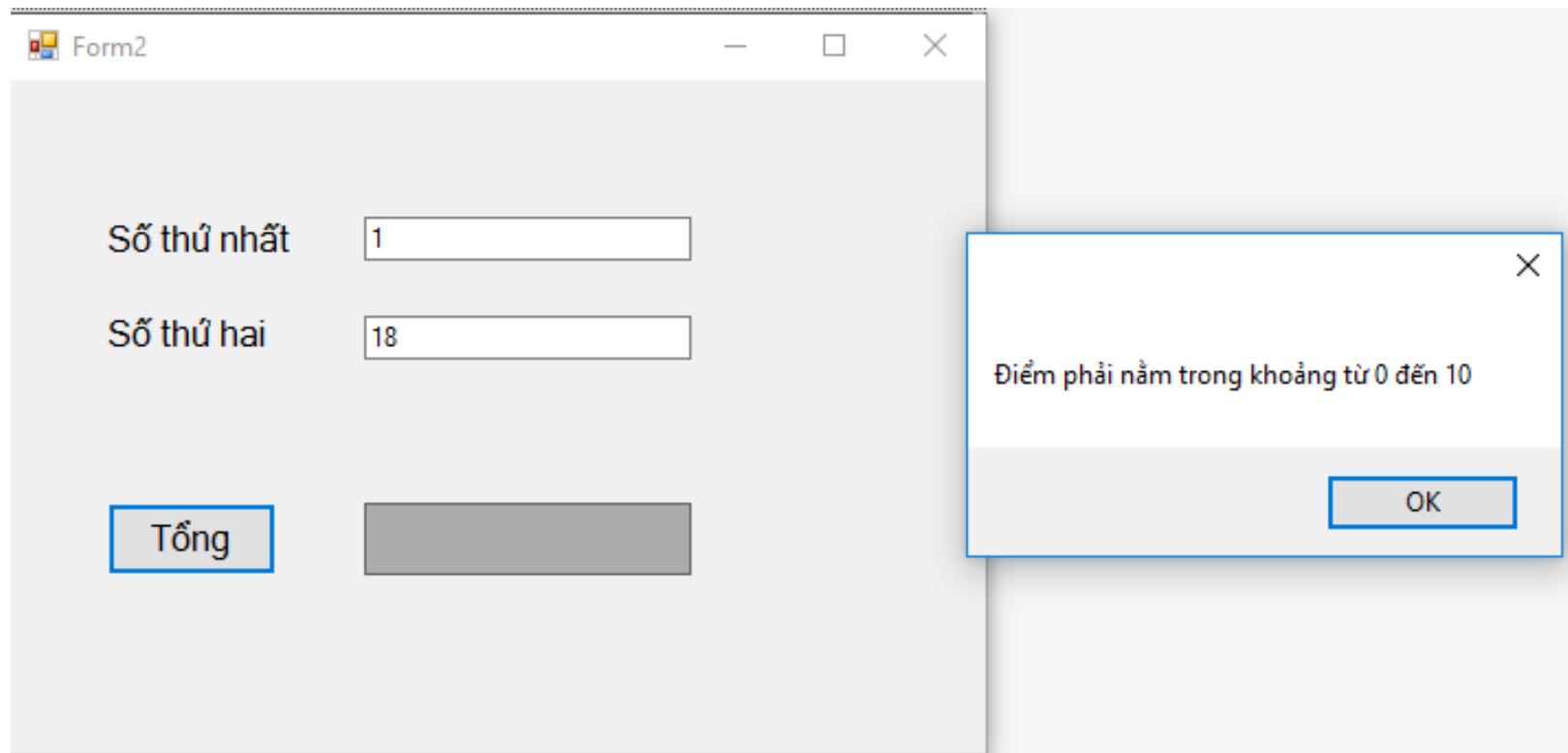
Xử lý ngoại lệ - Lệnh Throw

- Được dùng để phát sinh một tín hiệu bất thường làm phát sinh ngoại lệ.
- Khi phát sinh ngoại lệ, việc thực thi trong CLR sẽ dừng để tìm trình xử lý ngoại lệ, nếu không tìm thấy, thì chương trình sẽ kết thúc.

Xử lý ngoại lệ - Lệnh Throw

```
try
{
    int so1, so2, kq;
    so1 = int.Parse(txtSo1.Text);
    so2 = int.Parse(txtSo2.Text);
    if (int.Parse(txtSo1.Text) > 10 || int.Parse(txtSo1.Text) < 0
        || int.Parse(txtSo2.Text) > 10 || int.Parse(txtSo2.Text) < 0)
        throw new Exception("Điểm phải nằm trong khoảng từ 0 đến 10 ");
    kq = so1 + so2;
    lbKetQua.Text = kq.ToString();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

Xử lý ngoại lệ - Lệnh Throw



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.
 - Lệnh unchecked: hủy bỏ kiểm tra, không làm phát sinh ngoại lệ OverflowException khi có lỗi tràn số
 - Ví dụ: xem tài liệu học tập Lập trình giao diện

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);
}
```

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