



TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa CNTT – Bộ môn CNPM

LẬP TRÌNH WINDOWS



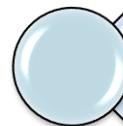
Giang viên: Nguyễn Thị Phương Dung

Email: dungntp@tlu.edu.vn

SĐT: 0946 079 903



Giới thiệu chung về môn học



Đề cương môn học



Cách thức đánh giá môn học



Phương pháp dạy và học



Giáo trình, tài liệu học tập



Chuẩn đầu ra cần đạt được





Đề cương môn học

- **Tên tiếng anh:** Windows Programming
- **Mã số:** CSE283
- **Số tín chỉ:** 3 tín chỉ
- **Số tiết:** 30 tiết lý thuyết, 15 tiết thực hành
(chi tiết được phân bổ trong bảng sau)





Chi tiết đề cương môn học

TT	Tên chương	Số tiết 45 tiết ~ 3 tín chỉ		
		Tổng số	Lý thuyết	Thảo luận, BT
1	Giới thiệu C#, Kiểu dữ liệu, biến, vào/ra dữ liệu, phép toán, cấu trúc điều khiển, rẽ nhánh	6	3	3
2	OOP trong C#	6	3	3
3	Lập trình winform cơ bản, sự kiện và ủy quyền	9	6	3
4	Lập trình winform nâng cao	9	6	3
5	Lập trình winform có kết nối cơ sở dữ liệu	9	6	3
6	Xây dựng ứng dụng sử dụng C#	6	6	
	Cộng:	45	30	15





Cách đánh giá

- Điểm quá trình: 50%
 - Chuyên cần (đi học đầy đủ)
 - 2 bài kiểm tra
 - 1 bài tập lớn
- Điểm kiểm tra cuối kỳ: 50%
 - Hình thức thi: thực hành 60 đến 90 phút





Phương pháp dạy và học

- Giảng viên:
 - Thuyết trình, có minh họa.
 - Nêu vấn đề, thảo luận tại lớp.
 - Chia nhóm lớp
- Sinh viên
 - Tự nghiên cứu tài liệu
 - Làm bài tập độc lập và theo nhóm





Tài liệu học tập

- Bài giảng của giáo viên
- Tài liệu tham khảo:
 - Giáo trình lập trình nâng cao (ĐHTL)
 - Kỹ thuật lập trình ứng dụng C#.NET toàn tập (NXB Lao động – Xã hội)
 - Các giải pháp lập trình C#
 - Windows Forms Programming with C#
 - Programming C# 4.0, sixth editions





Chuẩn đầu ra cần đạt được

Thành thạo các kỹ năng trong lập trình C#:

- Lập trình hướng đối tượng: Lớp, kế thừa, đa hình, hàm ảo, lớp trừu tượng, nạp chồng toán tử và nạp chồng hàm.
- Xử lý ngoại lệ try...catch
- Xây dựng phần mềm winform từ cơ bản đến nâng cao có kết nối cơ sở dữ liệu.





TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa CNTT – Bộ môn CNPM

LẬP TRÌNH WINDOWS



Giang viên: Nguyễn Thị Phương Dung

Email: dungntp@tlu.edu.vn

SĐT: 0946 079 903



Tổng quan về .Net Framework và ngôn ngữ lập trình C#



Tổng quan về .Net Framework

- Được phát triển bởi Microsoft
- Là một nền tảng lập trình và thực thi ứng dụng chủ yếu trên hệ điều hành Microsoft Windows
- Bao gồm môi trường Common Language Runtime (CLR) và tập các thư viện hỗ trợ lập trình .Net Framework Class Library





Common Language Runtime (CLR)

CLR là một máy ảo, cung cấp các dịch vụ:

- An ninh phần mềm (*security*)
- Quản lý bộ nhớ (*memory management*)
- Xử lý lỗi ngoại lệ (*exception handling*).





.Net Framework Class Library

Là những thư viện hỗ trợ việc xây dựng các chương trình phần mềm như:

- Lập trình giao diện
- Truy cập, kết nối cơ sở dữ liệu
- Ứng dụng web
- Các giải thuật, cấu trúc dữ liệu
- Giao tiếp mạng
- ...





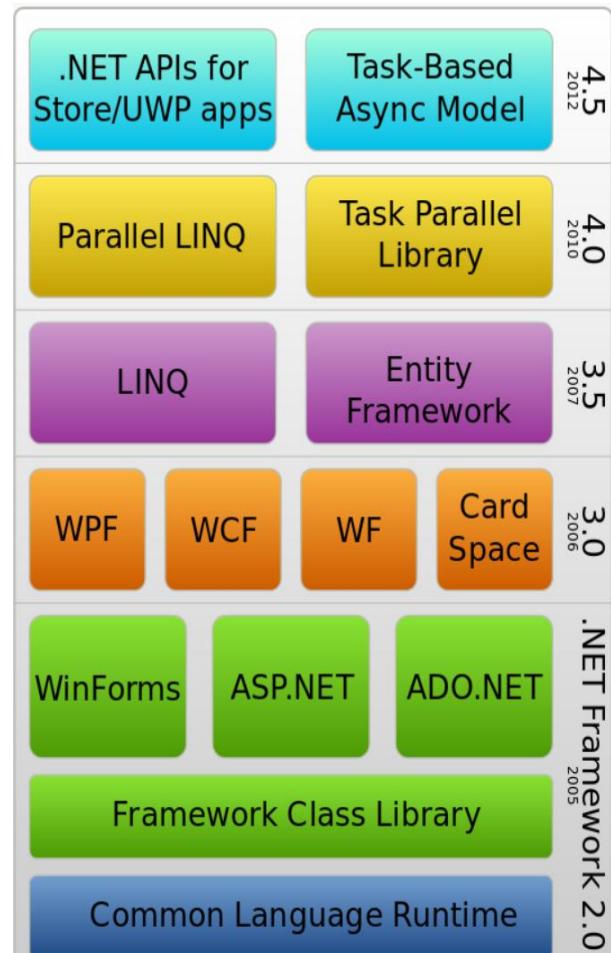
Một số thư viện nền tảng

Namespace	Description
System	Chứa các lớp cơ bản
System.IO	Chứa các lớp cho thao tác Input và Output
System.Net	Chứa các lớp liên quan đến network protocol
System.Collections	Chứa các lớp liên quan đến xử lý tập hợp
System.Data	Chứa các lớp của ADO.NET
System.Drawing	Chứa các lớp thực thi chức năng GUI
System.Threading	Chứa các lớp lập trình MultiThread
System.Web	Chứa các lớp liên quan đến HTTP protocol
System.Xml	Chứa các lớp liên quan XML



.Net Framework

- Qua nhiều giai đoạn phát triển, đến nay .Net Framework đã tích hợp rất nhiều thành phần thiết kế sẵn giúp cho việc lập trình được nhanh hơn, đơn giản hơn.
- Hỗ trợ đa ngôn ngữ: C++.Net, VB.Net, Jscript.Net, F#, C#.





Giới thiệu về ngôn ngữ lập trình C#

- C# là ngôn ngữ lập trình đơn giản:
 - C# khá giống C / C++ về diện mạo, cú pháp, biểu thức, toán tử.
 - Các chức năng của C# được lấy trực tiếp từ ngôn ngữ C / C++ nhưng được cải tiến để làm cho ngôn ngữ đơn giản hơn.





Giới thiệu về ngôn ngữ lập trình C#

- C# là ngôn ngữ hiện đại, có những tính năng:
 - Xử lý ngoại lệ
 - Thu gom bộ nhớ tự động
 - Có những kiểu dữ liệu mở rộng
 - Bảo mật mã nguồn





Giới thiệu về ngôn ngữ lập trình C#

- C# là ngôn ngữ hướng đối tượng với những đặc tính:
 - Sự đóng gói (encapsulation)
 - Sự kế thừa (inheritance)
 - Tính đa hình (polymorphism)





Giới thiệu về ngôn ngữ lập trình C#

- C# là ngôn ngữ mạnh mẽ và mềm dẻo khi được dùng để tạo ra các ứng dụng:
 - Xử lý văn bản
 - Xử lý đồ họa
 - Xử lý bảng tính
 - Thậm chí tạo ra những trình biên dịch cho các ngôn ngữ khác.





Giới thiệu về ngôn ngữ lập trình C#

- C# là một ngôn ngữ lập trình trực quan
- Là một trong những ngôn ngữ được phát triển ở nền tảng .Net
- Để sử dụng được C# cần cài đặt .Net Framework
- Để sử dụng được các tính năng lập trình trực quan cần cài đặt Microsoft Visual Studio





Một số dạng ứng dụng của C#

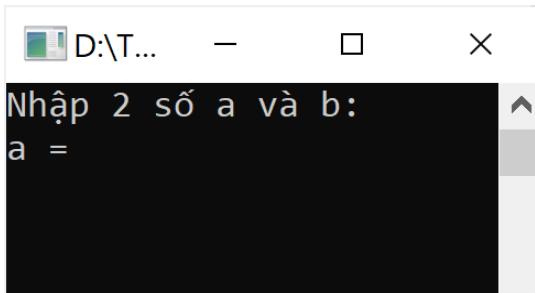
- Chương trình Console
 - Giao tiếp với người dùng bằng bàn phím
 - Chỉ sử dụng cửa sổ dòng lệnh, không có giao diện đồ họa
- Chương trình winform
 - Giao tiếp với người dùng bằng bàn phím và chuột
 - Có giao diện đồ họa và xử lý sự kiện
- Chương trình webform
 - Kết hợp với ASP.NET, C# đóng vai trò xử lý ngầm
 - Có giao diện đồ họa và xử lý sự kiện



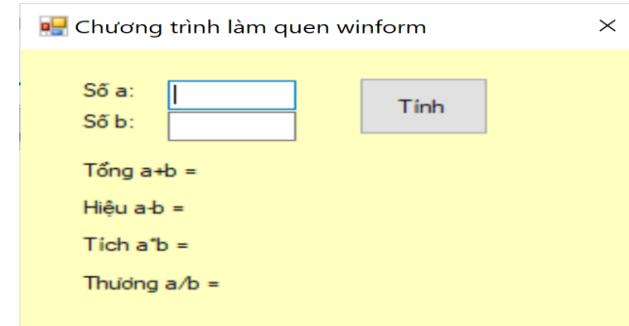


Giao diện các loại ứng dụng trong C#

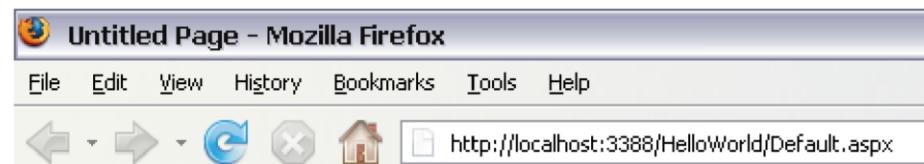
- Chương trình Console:



- Chương trình winform:



- Chương trình webform:



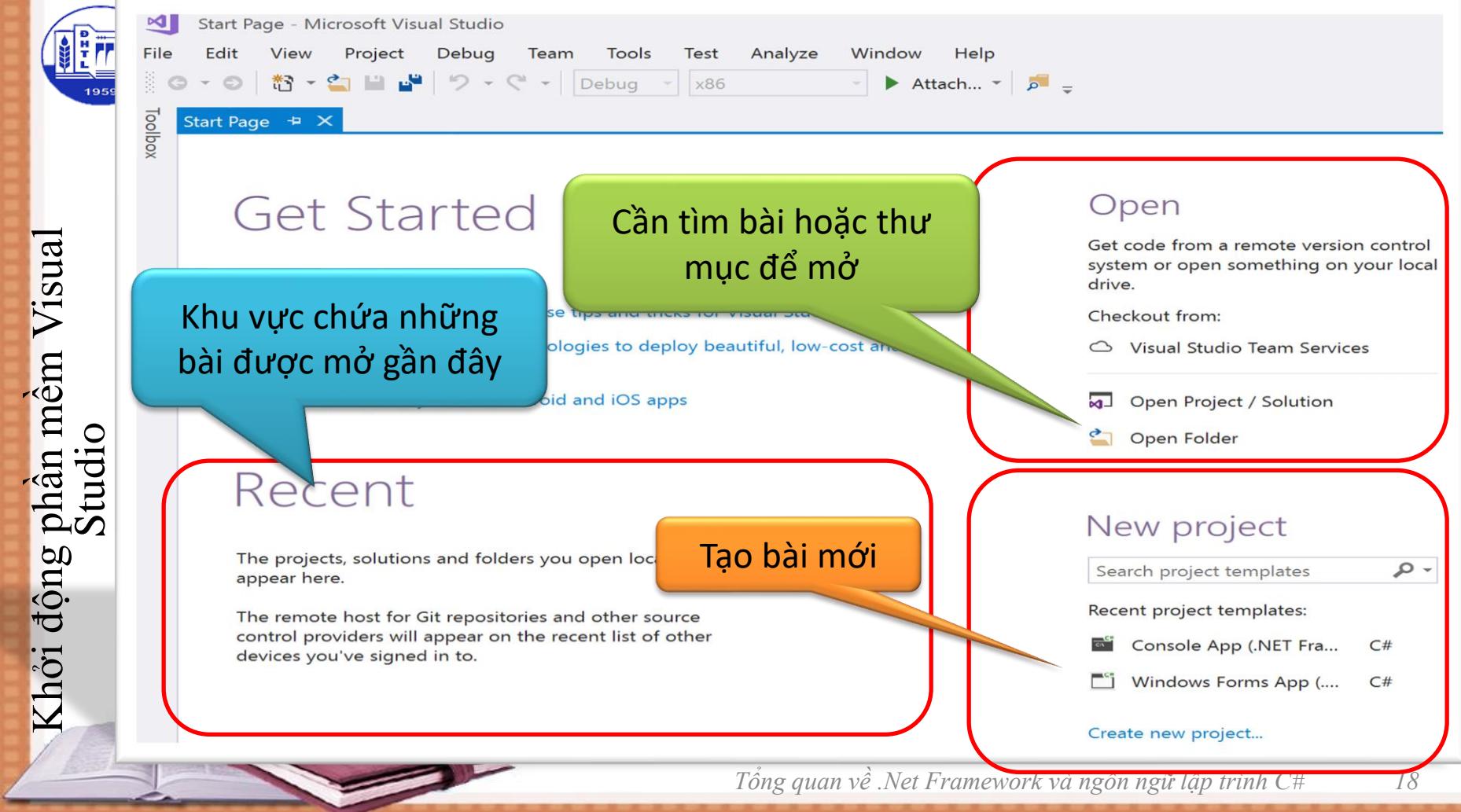


Cách tạo một chương trình Console trong C#



Khởi động phần mềm Visual Studio





Tạo mới một chương trình

New Project

Recent

Installed

Visual C#

Get Started
Windows Desktop
.NET Standard

Lựa chọn Visual C# để đảm bảo chương trình được viết bằng C#

Not finding what you are looking for?

Open Visual Studio Installer

Name:

ConsoleApp1

Location:

D:\

Solution name:

ConsoleApp1

Framework:

.NET Framework 4.6.1

Sort by: Default



WPF App (.NET Framework)

Visual C#



Windows Forms App (.NET Framework)

Visual C#



Console App (.NET Framework)

Visual C#



Class Library (.NET Standard)

Visual C#



Class Library (.NET Framework)

Visual C#



Shared Project

Visual C#



Class Library (Legacy Portable)

Visual C#

Đặt lại tên cho project của mình (1 project là 1 bài toán nhỏ)

Check vào ô này để hệ thống tự tạo thư mục mới để lưu trữ dự án

Lựa chọn Console App để tạo một chương trình chạy bằng dòng lệnh

Nhớ bấm nút Browse để lựa chọn nơi lưu trữ dự án

Đặt lại tên cho solution (1 solution là 1 chương trình lớn gồm 1 hoặc nhiều bài toán nhỏ)

Cuối cùng bấm nút OK để hoàn tất

Browse...

Create directory for solution

OK

Cancel



Cấu trúc thư mục solution

(D:) > MySolution

Tên thư mục chứa solution

Name

Mỗi project được lưu trong một thư mục con

MyApp

NhapXuat

MySolution.sln

Tên file solution

Mỗi thư mục project con sẽ chứa những file mã chương trình có phần mở rộng là .cs

ne
bin
obj
Properties
NhapXuat.csproj
Program.cs



Làm việc với một chương trình Console trong C#

The screenshot shows the Microsoft Visual Studio interface. In the center, the code editor displays the following C# code:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Nhaphuat
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13         }
14     }
15 }
```

Three callout boxes provide explanatory text:

- A green callout box points to the Solution Explorer window, which lists a solution named 'BTChuongl' containing a project named 'Nhaphuat'.

Một Solution có thể có nhiều Project.
- A blue callout box points to the code editor, highlighting the namespace declaration.

Mỗi Project có thể có nhiều Class
- An orange callout box points to the Solution Explorer window again, specifically highlighting the project node.

Cửa sổ Solution Explorer cho biết thông tin về Solution

The Solution Explorer window is highlighted with a red border.



Làm việc với một chương trình Console trong C#

Sau khi viết xong chương trình, ấn vào nút Start để thực hiện dịch và chạy chương trình

Mỗi Project được tạo một không gian tên riêng.

Mỗi Project có một hàm Main() riêng. Các lệnh sẽ được viết vào trong hàm Main()

```
5  using System.Threading;
6
7  namespace Nhaphxuat
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13         }
14     }
15 }
```



Làm việc với một chương trình Console trong C#

The screenshot shows the Microsoft Visual Studio interface with the following details:

- Title Bar:** BTChuongl - Microsoft Visual Studio
- Menu Bar:** File, Edit, View, Project, Build, Debug, Team, Tools, Test, Analyze, Window, Help
- Toolbox:** Standard icons for file operations.
- Solution Explorer:** Shows a single project named 'Nhapxuat' with files: Properties, References, App.config, and Program.cs.
- Properties:** Standard icons for project settings.
- Task List:** Standard icons for code navigation.
- Code Editor:** Displays the 'Program.cs' file content:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;

6

7  namespace Nhapxuat
8  {
9      class Program
10     {
11         static void Main(string[])
12         {
13         }
14     }
15 }
```
- Annotations:** A red box highlights the first five 'using' statements in the code editor. A blue callout bubble points to the 'Properties' tab in the Solution Explorer with the text: "Mỗi Project đều được mặc định thêm các chỉ thị này nhằm thuận tiện cho việc viết lệnh".
- Status Bar:** 161 %, Ln 12, Col 10, Ch 10, INS, Ready, Error List, Add to Source Control, 23

Mỗi Project đều được mặc định
thêm các chỉ thị này nhằm thuận
tiện cho việc viết lệnh



Lệnh nhập/xuất trong C#

- Việc nhập, xuất dữ liệu ra màn hình Console trong C# sử dụng lệnh **ReadLine()**, **WriteLine()**
- 2 lệnh này thuộc lớp **Console** trong namespace **System**
- Do đó ở đầu chương trình sử dụng chỉ thị **using System** thì trong chương trình chỉ cần viết **Console.WriteLine()** mà không cần viết **System.Console.WriteLine();**





Lệnh nhập/xuất trong C#

- VD:

```
namespace Nhaphxuat
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, what your name?");
            string name = Console.ReadLine();
            Console.WriteLine("Hello " + name);
            Console.ReadLine();
        }
    }
}
```





Nhập dữ liệu vào một biến

- Chú ý:
 - Lệnh **ReadLine()** dùng để nhập một dòng dữ liệu
 - Muốn nhập dữ liệu kiểu số **cần chuyển đổi kiểu** dữ liệu cho dòng dữ liệu nhập vào
 - Lệnh chuyển kiểu: là các lệnh thuộc lớp **Convert** trong namespace **System**
 - VD: **int a;**
a = Convert.ToInt32(Console.ReadLine());





Các kiến thức đã học trong bài

- Tổng quan về .Net Framework
- Tổng quan về ngôn ngữ lập trình C#
- Cách tạo chương trình Console trong C#





TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa CNTT – Bộ môn CNPM

LẬP TRÌNH WINDOWS



Giảng viên: Nguyễn Thị Phương Dung
Email: dungntp@tlu.edu.vn



Các khái niệm cơ bản trong C#

- Các kiểu dữ liệu
- Các từ khóa
- Các toán tử
- Các cấu trúc điều khiển





Các kiểu dữ liệu trong C#

Kiểu C#	Số byte	Kiểu .NET	Mô tả
byte	1	Byte	Số nguyên dương không dấu từ 0 đến 255
char	2	Char	Ký tự Unicode
bool	1	Boolean	Giá trị logic true / false
sbyte	1	Sbyte	Số nguyên có dấu từ -128 đến 127
short	2	Int16	Số nguyên có dấu từ -32768 đến 32767
ushort	2	UInt16	Số nguyên dương không dấu từ 0 đến 65535
int	4	Int32	Số nguyên có dấu từ -2.147.483.647 đến 2.147.483.647
uint	4	UInt32	Số nguyên không dấu từ 0 đến 4.294.967.295
float	4	Single	Kiểu dấu chấm động, giá trị xấp xỉ từ -3.4E-38 đến 3.4E+38, với 7 chữ số có nghĩa
double	8	Double	Kiểu dấu chấm động có độ chính xác gấp đôi, giá trị xấp xỉ từ -1.7E-308 đến 1.7E+308, với 15, 16 chữ số có nghĩa
decimal	8	Decimal	Có độ chính xác đến 28 con số và giá trị thập phân, được dùng trong tính toán tài chính, kiểu này đòi hỏi phải có hậu tố “m” hay “M”
long	8	Int64	Kiểu số nguyên có dấu có giá trị trong khoảng -9.223.370.036.854.775.808 đến 9.223.372.036.854.775.807
ulong	8	UInt64	Số nguyên không dấu từ 0 đến 0xffffffffffffffff



Các từ khóa trong C#

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



Xuất các ký tự đặc biệt

Ký tự	Ý nghĩa
\'	Dấu nháy đơn
\"	Dấu nháy kép
\\"	Dấu chéo
\0	Ký tự null
\a	Alert
\b	Backspace
\f	Sang trang form feed
\n	Dòng mới
\r	Đầu dòng
\t	Tab ngang
\v	Tab dọc





Các toán tử

Toán tử một ngôi	Ý nghĩa	Ví dụ
++	Tăng 1	$++a \approx a = a+1$ // tiền tố, tăng trước khi sd a $a++ \approx a = a+1$ // hậu tố, tăng sau khi sd a
--	Giảm 1	$--a \approx a = a-1$ // tiền tố, giảm trước khi sd a $a-- \approx a = a-1$ // hậu tố, giảm sau khi sd a
-	Lấy đối	-a lấy số đối của số a





Các toán tử

Toán tử hai ngôi	Ý nghĩa
=	Toán tử gán
+	Phép cộng
-	Phép trừ
*	Phép nhân
/	Phép chia lấy phần nguyên
%	Phép chia lấy phần dư





Các toán tử

Toán tử tự gán (2 ngôi)	Ý nghĩa
$+=$	$a += b \Leftrightarrow a = a+b$
$-=$	$a -= b \Leftrightarrow a = a-b$
$*=$	$a *= b \Leftrightarrow a = a*b$
$/=$	$a /= b \Leftrightarrow a = a/b$
$%=$	$a %= b \Leftrightarrow a = a\%b$





Toán tử điều kiện (3 ngôi)

- Cú pháp:

Kết quả = (biểu thức kiểm tra) ? giá trị đúng : giá trị sai

- VD: $\max = (a > b) ? a : b;$

- Tương đương câu lệnh *if ... else* như sau:

if(a>b)

max = a;

else

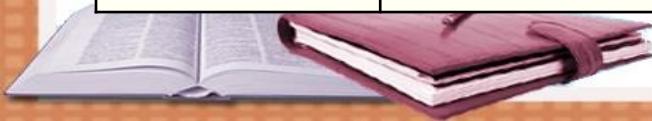
max = b;





Toán tử quan hệ

Toán tử	Mô tả	Ví dụ
<code>==</code>	So sánh bằng	$7 == 5$ // trả về false $(b=2) == 5$ // trả về false
<code>!=</code>	Khác	$(3 != 2)$ // trả về true
<code><</code>	Nhỏ hơn	$(5 < 5)$ // trả về false
<code>></code>	Lớn hơn	$(3 > 2)$ // trả về true
<code><=</code>	Nhỏ hơn hoặc bằng	$(6 <= 6)$ // trả về true
<code>>=</code>	Lớn hơn hoặc bằng	$(6 >= 4 + 2)$ // trả về true





Toán tử Logic

Toán tử	Chức năng
&&	Trả kết quả là True khi cả 2 toán hạng đều là True
 	Trả về kết quả là True khi chỉ một trong 2 toán hạng là True
!	Chuyển đổi giá trị từ True thành False và ngược lại





Chuyển đổi kiểu dữ liệu

- Chuyển đổi ngầm định
 - Được thực hiện ngầm định
 - Không bị mất thông tin
 - Ví dụ:
 - short x = 20;
 - int y = x; //=> chuyển đổi ngầm định và không mất thông tin, vì mọi giá trị kiểu short đều thuộc về int





Chuyển đổi dữ liệu

- Chuyển đổi tường minh
 - Gán ép một giá trị cho một biến thuộc kiểu dữ liệu khác: <tên biến> = (tên kiểu)<biến kiểu lớn hơn>;
 - Ví dụ:
 - short x; int y = 100; x = y;
 - //không thực hiện được vì kiểu của x < kiểu của y => việc chuyển ngầm định sẽ bị mất thông tin. Như vậy, muốn phép gán này không bị lỗi thì phải viết như sau: **x = (short)y;**//chuyển đổi tường minh/ép kiểu





Chuyển đổi dữ liệu

- Chuyển đổi tường minh
 - Sử dụng các lệnh chuyển kiểu trong lớp Convert thuộc namespace System
 - Ví dụ:
`int a;`
`a = Convert.ToInt32(Console.ReadLine());`





Bài tập

- Viết chương trình nhập tên, năm sinh. Xuất ra lời chào tên vừa nhập và thông báo số tuổi của người đó dựa vào năm sinh





Cách khai báo biến

- Giống C++ đã học

- Cú pháp:

<Kiểu_Dữ_Liệu> <tên_biến> [= <giá_trị>] ;

- Ví dụ:

- int a; //khai báo biến a kiểu số nguyên

- int x = 10; //khai báo biến x kiểu số nguyên và gán giá trị khởi tạo ban đầu cho x là 10





Các cấu trúc điều khiển

- Giống C++ đã học
- Cấu trúc rẽ nhánh:
 - if
 - if....else
 - switch....case
 - Toán tử điều kiện





Câu lệnh rẽ nhánh if

- Cú pháp:

*if (Biểu_thức_Boolean_1)
 Câu_lệnh_1;*

- Hoặc:

*if (Biểu_thức_Boolean_1)
{
 Câu_lệnh_1;
 Câu_lệnh_2;
}*



Các câu lệnh sẽ
được thực hiện
khi biểu thức
kiểm tra có kết
quả là đúng



Câu lệnh rẽ nhánh if...else

- Cú pháp

if (Biểu_thức_Boolean)

//Câu lệnh khi biểu thức Boolean đúng

else

//Câu lệnh khi biểu thức Boolean sai





Câu lệnh rẽ nhánh if...else

- Hoặc

```
if (Biểu_thức Boolean){  
    Câu_lệnh_khi_đúng_1;  
    Câu_lệnh_khi_đúng_2;  
}  
else{  
    Câu_lệnh_khi_sai_1;  
    Câu_lệnh_khi_sai_2;  
}
```





Câu lệnh rẽ nhánh switch...case

- Việc lựa chọn nhánh nào để thực thi được dựa trên **biểu thức điều khiển**
- Biểu thức điều khiển cho một câu lệnh switch phải trả về một giá trị bool hoặc một bộ liệt kê các **hằng số**, hoặc một giá trị kiểu số nguyên hoặc một kí tự

```
switch (Bieu_thuc_dieu_khien)
{
    case Hang_so_1:
        Dãy_câu_lệnh_1
        break;
    case Hang_so_2:
        Dãy_câu_lệnh_2
        break;
    .
    .
    case Hang_so_n:
        Dãy_câu_lệnh_n
        break;
    default:
        Dãy_câu_lệnh_mac_dinh
        break;
}
```



Câu lệnh rẽ nhánh switch...case

- Câu lệnh rẽ nhánh switch...case thường được sử dụng khi muốn thực hiện các khối lệnh khác nhau với mỗi lựa chọn khác nhau của một biểu thức hay giá trị của một biến.
- Sau các câu lệnh của mỗi lựa chọn case nên dùng lệnh break để bỏ qua các lựa chọn case khác
- Sau các lựa chọn case nên có lựa chọn default để thông báo rằng chưa có lựa chọn case nào được chọn.





Câu lệnh rẽ nhánh switch...case

- Có thể có trường hợp nhiều lựa chọn cùng thực hiện một công việc
- Ví dụ:

```
switch(thang)
{
    case 4:
    case 6:
    case 9:
    case 11:
        Console.WriteLine("Tháng {0} có 30 ngày", thang);
        break;
```





Các cấu trúc điều khiển

- Cấu trúc lặp
 - for
 - while
 - do... while
 - foreach





Cấu trúc vòng lặp for

- Được dùng khi biết trước số lần lặp
- Cú pháp:

for (khởi tạo biến chạy; kiểm tra biến chạy; thay đổi giá trị biến chạy)

{

các câu lệnh;

}





Cấu trúc lặp for

- Ví dụ: tính tổng các số nguyên dương nhỏ hơn 10
Thực hiện gán biến $i = 0$;
Lặp 10 lần, mỗi lần cộng i vào tổng rồi tăng i lên 1 đơn vị

```
int tong = 0;
for (int i = 1; i < 10; i++)
    tong = tong + i;
```





Hãy cho biết kết quả của vòng lặp sau?

```
int i;  
for(i=0; i<10; i-=2)  
    cout<<i<<" ";  
cout<<endl;
```





Hãy cho biết kết quả của vòng lặp sau?

```
int i=0;  
for(;i<10; i++)  
    cout<<i<<" ";  
cout<<endl;
```





Cấu trúc lặp while

- Được dùng khi không biết trước số lần lặp
- Cú pháp: while(*biểu thức điều kiện*)
 câu lệnh;
• Hoặc: while(*biểu thức điều kiện*)
 {
 câu lệnh 1;
 câu lệnh 2;

 }





Vòng lặp while

- Chú ý:
 - Vòng lặp dừng khi biểu thức kiểm tra điều kiện cho ra giá trị sai
 - Do đó khối lệnh trong vòng lặp while phải có lệnh làm thay đổi giá trị biến chạy sao cho biểu thức kiểm tra điều kiện trở thành sai.
 - Nếu không thì vòng lặp sẽ bị lặp vô hạn





Vòng lặp while

- Ví dụ:
 - Lệnh $t = t/10$ là lệnh làm thay đổi giá trị biến chạy
 - Sau mỗi lần thực hiện lệnh này, t sẽ giảm 10 lần, cho đến một lúc nào đó t sẽ bằng 0. Khi đó biểu thức kiểm tra $t > 0$ sẽ không còn đúng nữa và vòng lặp while sẽ dừng

```
while (t > 0)
{
    tong += t % 10;
    t = t / 10;
}
```





Cấu trúc lặp do...while

- Được dùng khi không biết trước số lần lặp nhưng cần thực hiện các câu lệnh ít nhất 1 lần trước khi kiểm tra điều kiện để lặp tiếp
- Cú pháp:
 - do câu lệnh;
while(biểu thức điều kiện);
 - Hoặc:
 - do {
các câu lệnh;
}
while(biểu thức điều kiện);





Cấu trúc lặp do...while

- Chú ý: giống như vòng lặp while
 - Vòng lặp do...while sẽ dừng khi biểu thức kiểm tra điều kiện cho ra giá trị sai
 - Do đó khôi lệnh trong vòng lặp do...while phải có lệnh làm thay đổi giá trị biến chạy sao cho biểu thức kiểm tra điều kiện dần bị sai.
 - Nếu không thì vòng lặp sẽ bị lặp vô hạn





Cấu trúc lặp do...while

- Ví dụ:

```
do
{
    Console.WriteLine("Nhập số n thỏa mãn 0<n<2000000: ");
    n = Convert.ToInt32(Console.ReadLine());
}
while (n <=0 || n >= 2000000);
```

Đây chính là lệnh làm
cho vòng lặp dừng





Hãy cho biết kết quả của vòng lặp sau?

```
int n;  
do{  
    cout<<"Nhập N = ";  
    cin>>n;  
}  
while(n<0 && n>1000000);
```





Cấu trúc vòng lặp foreach

- Dùng khi duyệt tất cả các phần tử trong mảng (mảng đã được khởi tạo)
- Cú pháp: với **a** là một mảng các phần tử có giá trị, **x** là một biến cùng kiểu với các phần tử của mảng **a**
foreach(x in a)

{

//công việc

}





Cấu trúc vòng lặp foreach

- Dùng khi duyệt tất cả các phần tử trong mảng
- Ví dụ:

```
//khai báo mảng a gồm 10 phần tử kiểu số nguyên  
int[] a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
//đọc từng phần tử của mảng a  
foreach (int x in a)  
    //viết các phần tử vừa duyệt lên cùng một dòng  
    //mỗi phần tử cách nhau một dấu ; và một dấu cách  
    Console.WriteLine(x + "; ");
```





Kiến thức đã học trong bài

- Các kiểu dữ liệu
- Các từ khóa
- Các toán tử
- Các cấu trúc điều khiển





TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa CNTT – Bộ môn CNPM

LẬP TRÌNH WINDOWS



Giảng viên: Nguyễn Thị Phương Dung
Email: dungntp@tlu.edu.vn
SĐT: 0946 079 903



XỬ LÝ NGOẠI LỆ



Ngoại lệ là gì?

- Một ngoại lệ (Exception) là một vấn đề xuất hiện trong khi thực thi một chương trình.
- Ví dụ:
 - Lỗi chuyển đổi sai kiểu (chuyển ký tự thành số)
 - Lỗi chia cho số 0.
 - Lỗi truy cập tới phần tử của mảng với chỉ số không đúng
 - Lỗi truy cập vào đối tượng null
 -





Xử lý ngoại lệ

- Xử lý ngoại lệ là cách đưa ra những thông báo khi chương trình gặp phải ngoại lệ trong quá trình chạy, giúp chương trình vẫn có thể chạy đến khi kết thúc một cách mượt mà theo chủ ý, không bị dừng đột ngột.





Thực hiện bắt ngoại lệ

- Sử dụng khối lệnh try...catch()
- Cú pháp:

```
try
{
    //các lệnh có thể gây ra ngoại lệ
}
catch(/*khai báo biến bắt ngoại lệ*/)
{
    //các lệnh xử lý ngoại lệ
}
```





Ví dụ

Lệnh có khả năng bị lỗi

```
int a;  
Console.WriteLine("a = ");  
try  
{  
    a = Convert.ToInt32(Console.ReadLine());  
}  
catch(FormatException e)  
{  
    Console.WriteLine("Lỗi " + e.ToString());  
}
```

Lỗi bắt được từ lệnh
trên





Bắt nhiều lỗi cùng lúc

```
try
{
    //đoạn mã có khả năng bị lỗi
}
catch(/*loại lỗi 1*/)
{
    //đoạn mã xử lý
}
catch /*loại lỗi 2*/
{
    //đoạn mã xử lý
}
```





Ví dụ

```
try
{
    Console.Write("a = ");
    a = Convert.ToInt32(Console.ReadLine());
    Console.Write("b = ");
    b = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Kết quả: {0}/{1}={2}", a, b, a / b);
}
catch(FormatException e)
{
    Console.WriteLine("Lỗi nhập sai dữ liệu");
}
catch(DivideByZeroException e)
{
    Console.WriteLine("Lỗi chia cho 0 ");
}
```



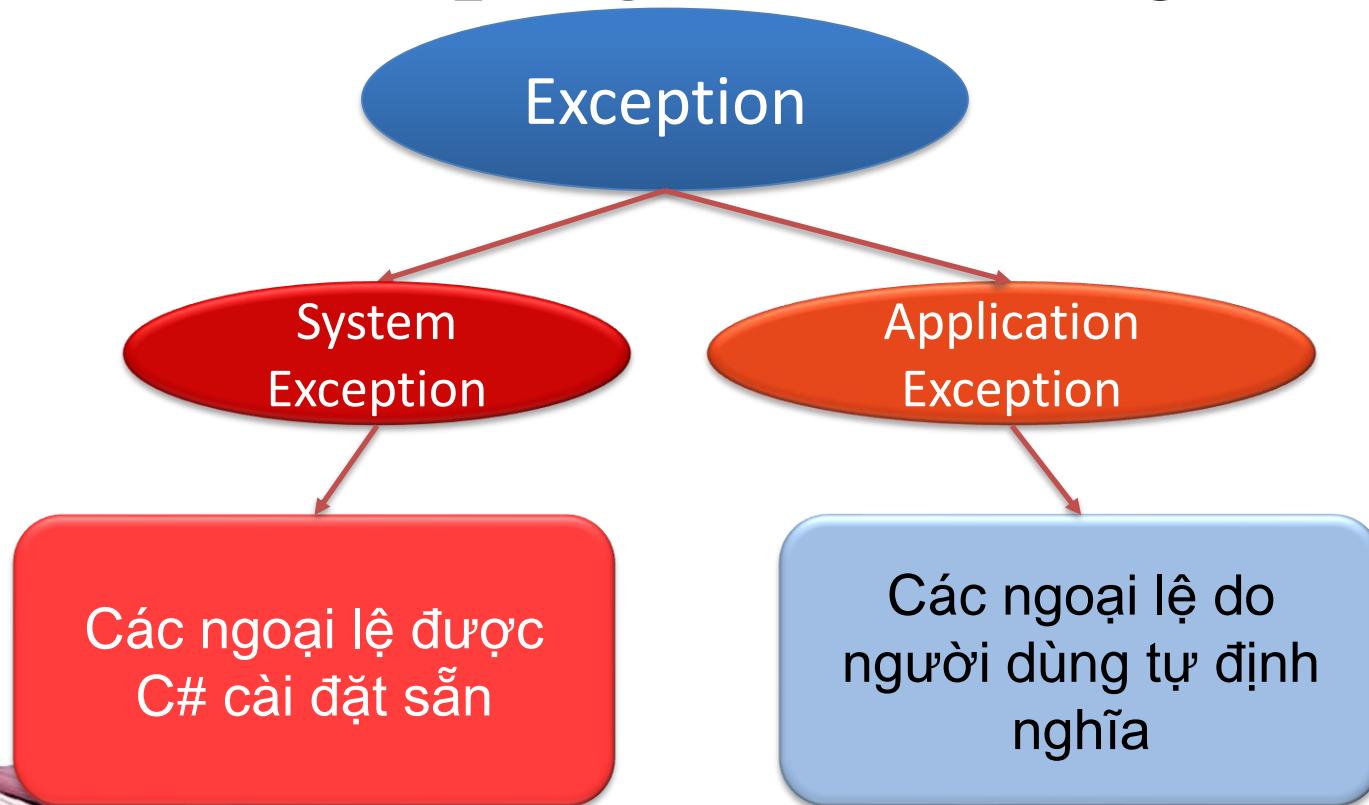


Bắt nhiều lỗi cùng lúc

```
try
{
    //đoạn mã có khả năng bị lỗi
}
catch(/*loại lỗi 1*/)
{
    //đoạn mã xử lý
}
catch /*loại lỗi 2*/
{
    //đoạn mã xử lý
}
....
finally
{
    //đoạn mã thực hiện mặc định cho dù có lỗi hoặc không lỗi xảy ra
}
```



Phân cấp ngoại lệ trong C#





Một số ngoại lệ thông dụng trong C#

Kiểu ngoại lệ	Ý nghĩa
DivideByZeroException	Lỗi chia cho số 0
InvalidCastException	Xử lý lỗi được tạo trong khi ép kiểu
OutOfMemoryException	Xử lý lỗi được tạo từ việc thiếu bộ nhớ rỗng
StackOverflowException	Xử lý lỗi được tạo từ việc tràn ngăn xếp (stack)
FormatException	Lỗi chuyển đổi định dạng





Một số ngoại lệ thông dụng trong C#

Kiểu ngoại lệ	Ý nghĩa
Exception	Lớp ngoại lệ cơ bản
SystemException	Lớp cơ bản của mọi ngoại lệ phát ra tại thời điểm chạy của chương trình.
IndexOutOfRangeException	Được ném ra tại thời điểm chạy khi truy cập vào một phần tử của mảng với chỉ số không đúng.
NullReferenceException	Ném ra tại thời điểm chạy khi một đối tượng null được tham chiếu.
AccessViolationException	Ném ra tại thời điểm chạy khi tham chiếu vào vùng bộ nhớ không hợp lệ.
InvalidOperationException	Ném ra bởi phương thức khi ở trạng thái không hợp lệ.



Một số ngoại lệ thông dụng trong C#

Kiểu ngoại lệ	Ý nghĩa
ArgumentNullException	Lớp này là con của ArgumentException, nó được ném ra bởi phương thức mà không cho phép thông số null truyền vào.
ArgumentOutOfRangeException	Lớp này là con của ArgumentException, nó được ném ra bởi phương thức khi một đối số không thuộc phạm vi cho phép truyền vào nó.
ExternalException	Lớp cơ bản cho các ngoại lệ xảy ra hoặc đến từ môi trường bên ngoài.
ArgumentException	Lớp cơ bản cho các ngoại lệ liên quan tới đối số (Argument).





Ví dụ

```
static float phepchia(int a, int b)
{
    float ketqua = 0;
    try
    {
        ketqua = a / b;
    }
    catch(DivideByZeroException)
    {
        Console.WriteLine("Mau so khong hop le");
    }
    return ketqua;
}
```





Bài tập

- Thiết kế một giao diện cho phép nhập 2 số a và b. Thực hiện đưa ra kết quả của các phép tính:
 - $a + b$
 - $a - b$
 - $a * b$
 - a / b
- Yêu cầu:
 - Đưa ra thông báo lỗi khi nhập sai dữ liệu
 - Đưa ra thông báo lỗi khi chia cho số 0





Bài tập 2

- Viết chương trình cho phép nhập vào một số có 7 chữ số
- Thông báo lỗi nếu nhập không đúng





Ngoại lệ tự định nghĩa

- Ké thừa lớp **ApplicationException**
- Cú pháp khai báo lớp ngoại lệ:

```
class <tên lớp>: ApplicationException  
{  
    public <tên lớp> (string msg) : base(msg){ }  
}
```





Ngoại lệ tự định nghĩa

- Cú pháp gọi lớp ngoại lệ thực hiện:
`throw new <tên lớp ngoại lệ>(<lời thông báo lỗi>);`
- Sử dụng: giống như khi sử dụng các lớp ngoại lệ cơ bản





Ví dụ ngoại lệ tự định nghĩa

```
class myexception:ApplicationException  
{  
    public myexception (string s):base(s) { }  
}
```





Ví dụ ngoại lệ tự định nghĩa

```
class canbo
{
    int tuoi;
    public void nhap()
    {
        Console.WriteLine("Nhập tuổi: ");
        try
        {
            tuoi = Convert.ToInt32(Console.ReadLine());
        }
        catch(FormatException ex)
        {
            Console.WriteLine(ex.ToString());
        }
        if (tuoi < 10 || tuoi > 18)
            throw new myexception("Tuổi phải nằm trong khoảng từ 10 đến 18");
    }
}
```



Ví dụ ngoại lệ tự định nghĩa

```
class Program
{
    static void Main(string[] args)
    {
        canbo cb = new canbo();
        try
        {
            cb.nhap();
        }
        catch (myexception ex)
        {
            Console.WriteLine(ex.ToString());
        }
    }
}
```



Bài tập

- Định nghĩa lớp ngoại lệ của riêng mình
- Thực hiện tạo một lớp tam giác có:
 - Dữ liệu: 3 cạnh
 - Phương thức: nhập, tính chu vi
 - Yêu cầu đưa ra thông báo lỗi khi nhập sai dữ liệu về cạnh tam giác
- Viết chương trình thực hiện:
 - Khai báo một mảng gồm 3 tam giác.
 - Nhập dữ liệu cho 3 tam giác đó
 - Đưa ra các thông báo lỗi có thể gặp khi chạy chương trình





Bài tập

- Định nghĩa lớp ngoại lệ của riêng mình để thực hiện bắt các ngoại lệ sau:
 - Lỗi nhập sai dữ liệu
 - Lỗi truy cập ngoài chỉ số mảng
 - Kiểm tra tuổi tuyển dụng lao động
 - Nếu tuổi <18 thì thông báo là quá trẻ
 - Nếu tuổi >40 thì thông báo là quá già
 - Nếu tuổi từ 18 đến 40 thì thông báo là đạt yêu cầu





TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa CNTT – Bộ môn CNPM

LẬP TRÌNH WINDOWS



Giang viên: Nguyễn Thị Phương Dung
Email: dungntp@tlu.edu.vn
SĐT: 0946 079 903



OPP TRONG C#



Nội dung

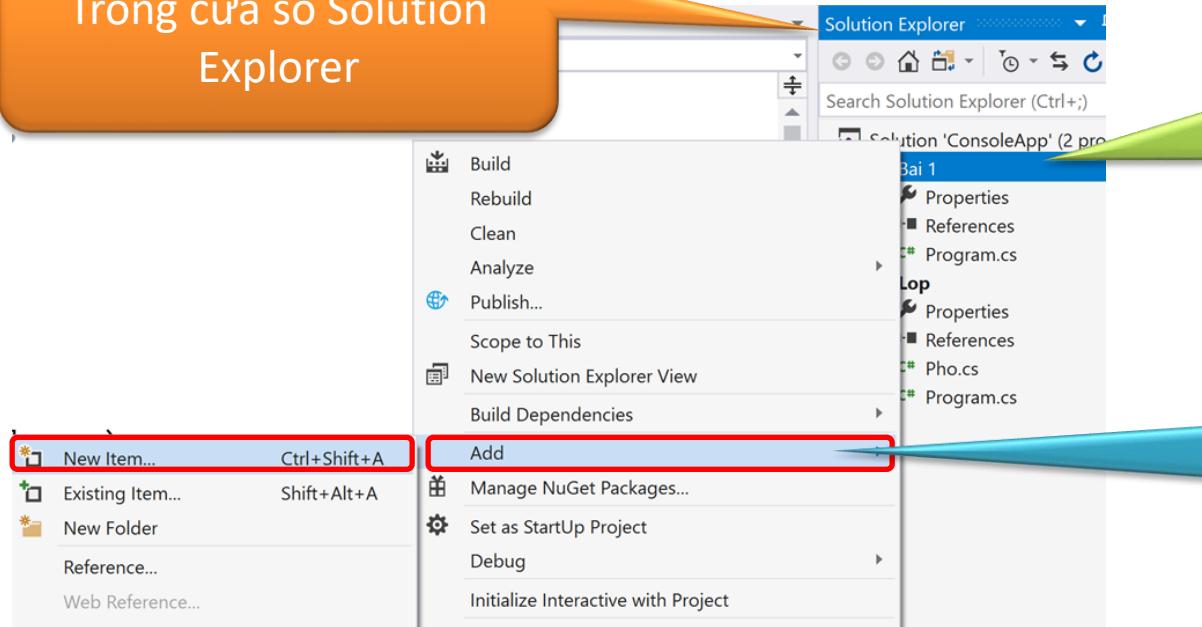
- Khai báo lớp
- Định nghĩa lớp
- Sử dụng lớp





Khai báo lớp

Trong cửa sổ Solution Explorer



Bấm chuột phải vào tên project

Chọn Add/New Item





Khai báo lớp

Add New Item - Bai 1

Installed

Visual C# Items

Sort by: Default

Search (Ctrl+E)

Type: Visual C# Items

An empty class is created.

Class

Interface

Windows Form

User Control

Component Class

User Control (WPF)

About Box

ADO.NET Entity Data Model

Application Configuration File

Application Manifest File

Assembly Information File

Name: Class1.cs

Add Cancel

Chọn Visual C# Items

Đặt tên lớp

Ấn nút Add



- Tạo một lớp Phanso thuộc project Bai 1
=> xuất hiện file Phanso.cs trong project Bai 1

```
Phanso.cs  X Program.cs
C# Bai 1  Bai_1.Phanso
1   using System;
2   using System.Collections;
3   using System.Linq;
4   using System.Text;
5
6   namespace Bai_1
7   {
8       class Phanso
9       {
10    }
11 }
12
```

Solution Explorer

Search Solution Explorer (Ctrl+Shift+F)

Solution 'ConsoleApp' (1 project)

- Bai 1
 - Properties
 - References
 - Phanso.cs
 - Program.cs



Định nghĩa lớp

- Cú pháp:

[quyền truy cập] class <tên lớp> [:lớp cơ sở]

{

[quyền truy cập] <kiểu dữ liệu> <tên thành phần>;

[quyền truy cập] <kiểu trả về> <tên phương thức> (danh sách các tham số)

{

//định nghĩa phương thức

}

}





Định nghĩa lớp

- Trong đó:
 - **class**: là từ khóa để khai báo lớp
 - **Kiểu dữ liệu**: là những kiểu cơ bản hoặc những kiểu đã được định nghĩa
 - **Kiểu trả về**: là những kiểu cơ bản hoặc những kiểu đã được định nghĩa hoặc **void** (phương thức không trả về dữ liệu)





Định nghĩa lớp

- Trong đó:
 - **Quyền truy cập:** là các quyền được liệt kê trong bảng sau





Định nghĩa lớp

Từ khóa	Giới hạn truy cập
public	Không hạn chế. (Những thành viên được đánh dấu là public có thể được dùng bởi bất kì các phương thức nào của lớp và có thể được dùng trong những lớp khác.)
private	Che dấu. Những thành viên được đánh dấu là private thì chỉ được sử dụng trong các phương thức của lớp.
protected	Thành phần nào được đánh dấu là protected trong lớp X thì chỉ được dùng trong lớp X và các lớp dẫn xuất từ X
internal	Được dùng trong các lớp có cùng namespace
partial	Được dùng khi lớp được khai báo và định nghĩa trên nhiều file





Ví dụ

Không khai báo quyền
truy cập thì mặc định là
private

```
namespace Bai_1
{
    class Phanso
    {
        int tuso, mauso;
        public void nhap()
        {
            Console.Write("Tu so = ");
            tuso = Convert.ToInt32(Console.ReadLine());
            Console.Write("Mau so = ");
            mauso = Convert.ToInt32(Console.ReadLine());
        }
    }
}
```





Chú ý

- Phương thức (method) là các hàm (function)
- Tên phương thức thường được đặt theo tên của hành động
- Tham số của phương thức nằm trong cặp ngoặc tròn ngay sau tên phương thức
- Muốn truyền tham chiếu thì nhớ thêm từ khóa **ref** hoặc **out**





Sử dụng lớp

- Khai báo đối tượng
 - <ten lớp> <tên đối tượng>;
- Ví dụ:

Phanso ps1, ps2; //Khai báo 2 đối tượng ps1 và ps2
thuộc kiểu Phanso





Sử dụng lớp

- Khởi tạo đối tượng
<tên đối tượng> = **new** <hàm khởi tạo của lớp>;
- Ví dụ:
ps1 = **new** Phanso(); //khởi tạo ps1 bằng hàm khởi tạo mặc định
ps2 = **new** Phanso(1,2); //khởi tạo ps2 bằng hàm khởi tạo có tham số)





Sử dụng lớp

- Truy cập tới dữ liệu và hàm thành phần của đối tượng:
 - Phụ thuộc vào quyền được truy cập vào dữ liệu đó
 - Sử dụng toán tử `(.)` để truy cập
 - Ví dụ: `ps1.nhap();` //gọi hàm nhập của lớp Phanso
`ps1.xuat();` //gọi hàm xuất của lớp Phanso





Ví dụ sử dụng lớp

```
class Program
{
    static void Main(string[] args)
    {
        Phanso A = new Phanso(); //khai báo và khởi tạo đối tượng
        Phanso B = new Phanso();
        Phanso C = new Phanso();
        Console.WriteLine("Nhập phân số A: ");
        A.nhap();
        Console.WriteLine("Nhập phân số B: ");
        B.nhap();
        C = A + B;
        Console.WriteLine("\nTổng 2 phân số là: ");
        A.xuat(); Console.Write(" + "); B.xuat(); Console.Write(" = "); C.xuat();
        Console.ReadLine();
    }
}
```





Nạp chồng toán tử

- Cú pháp:

public static <kiểu trả về> operator <toán tử>(danh sách tham số)

- Trong đó:

- Kiểu trả về là kiểu kết quả của phép tính
- Danh sách tham số bao gồm kiểu và tên tham số





Ví dụ

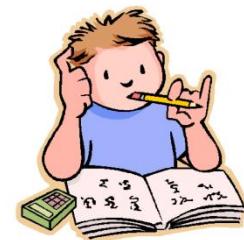
```
public static Phanso operator +(Phanso a, Phanso b)
{
    Phanso kq = new Phanso();
    kq.tuso = a.tuso * b.mauso + b.tuso * a.mauso;
    kq.mauso = a.mauso * b.mauso;
    return kq;
}
```





Bài tập

- Xây dựng một lớp phân số (đặt tên là PhanSo) bao gồm:
 - Các thành phần dữ liệu tử số và mẫu số
 - Phương thức nhập và hiển thị dữ liệu cho phân số
 - Xây dựng phương thức nạp chồng toán tử +, - , *, /
- Trong chương trình chính, khai báo và nhập dữ liệu cho 2 phân số ps1, ps2. Tính toán và hiển thị các kết quả sau:
 - $ps3 = ps1 + ps2$ (VD: $1/2 + 2/3 = 7/6$)
 - $ps3 = ps1 * ps2$
 - $ps3 = ps1 - ps2$
 - $ps3 = ps1 / ps2$





Bài tập

- Cải tiến bài toán trên:
 - Thêm phương thức rút gọn phân số
 - Hiển thị những phân số kết quả đã tính dưới dạng rút gọn





LẬP TRÌNH NÂNG CAO OPP TRONG C# (Tiếp)



Giang viên: Nguyễn Thị Phương Dung
Email: dungntp@tlu.edu.vn
SĐT: 0946 079 903



Nội dung

- Kế thừa
- Đa hình
- Hàm ảo
- Lớp trừu tượng
- Nạp chồng hàm





Kế thừa

- Là tính năng dùng lại trong lập trình hướng đối tượng.
- Khai báo một lớp dựa trên lớp đã tồn tại
- Lớp đã tồn tại gọi là **lớp cơ sở** hoặc **lớp cha (Base class)**
- Lớp kế thừa gọi là **lớp dẫn xuất** hoặc **lớp con (Derived class)**





Kế thừa

- Cú pháp khai báo
class <ten lớp dẫn xuất> : <ten lớp cơ sở>
{
 //định nghĩa lớp dẫn xuất
}
- Chú ý: Một số thành phần không được kế thừa
 - Các hàm tạo
 - Các hàm hủy





Kế thừa

- Trong lớp con gọi phương thức của lớp cha:
 - Sử dụng từ khóa **base**
 - Ví dụ:
 - Lớp CONNGUOI có các thông tin: họ tên, tuổi, quê quán, giới tính, ... và hàm khởi tạo có truyền vào các tham số
 - Lớp CANBO kế thừa từ lớp CONNGUOI và có thêm thông tin: hệ số lương, thậm niên công tác, ... => hàm khởi tạo truyền đầy đủ các tham số sẽ cần gọi lại hàm khởi tạo của lớp cha





Kế thừa

Lớp cha

```
class CONNGUOI
{
    protected string hoten, gioitinh, quequan;
    protected int tuoi;
    public CONNGUOI(string ht, string gt, string qq, int t)
    {
        hoten = ht;
        gioitinh = gt;
        quequan = qq;
        tuoi = t;
    }
}
```

Lớp con

```
class CANBO : CONNGUOI
{
    private double hesoluong;
    private int thamnien;
    public CANBO(string ht, string gt, string qq, int t, double hsl, int tn) : base(ht, gt, qq, t)
    {
        hesoluong = hsl;
        thamnien = tn;
    }
}
```

Hàm khởi tạo
có tham số
của lớp cha

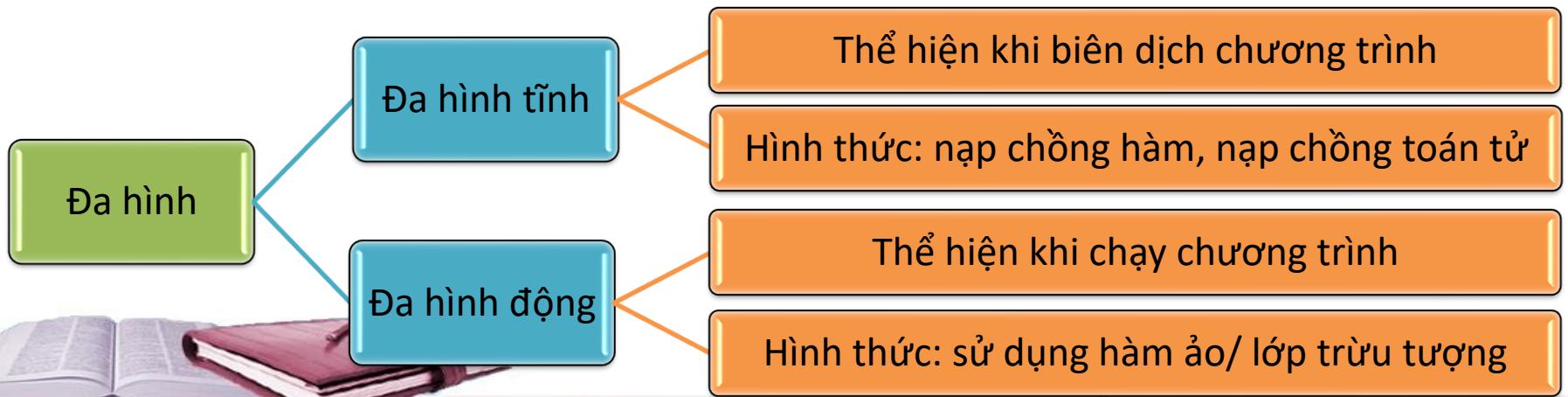
Hàm khởi tạo
có tham số
của lớp con

Gọi hàm khởi tạo
của lớp cha



Đa hình

- Là các hình thái thực hiện khác nhau của một phương thức
- Được phân làm 2 loại:





Đa hình tinh

- Sử dụng kỹ thuật nạp chồng hàm
- Là cách tạo ra những hàm:
 - Giống nhau về tên hoặc
 - Giống nhau về cả tên và kiểu trả về
 - Nhưng phải khác nhau về kiểu tham số hoặc
 - Khác nhau về số các tham số





Đa hình tĩnh

- Ví dụ:

2 hàm tạo có cùng tên, khác nhau số tham số truyền vào

```
class PHANSO
{
    int tuso, mauso;
    public PHANSO()
    {
        public PHANSO(int tu, int mau)
        {
            tuso = tu;
            mauso = mau;
        }
    }
}
```





Đa hình tĩnh

- Sử dụng kỹ thuật nạp chồng toán tử (đã học)
- Là cách sử dụng các toán tử +, - , *, /, % cho những kiểu dữ liệu người dùng tự định nghĩa
- Cú pháp:

public static <kiểu trả về> operator <toán tử>
(danh sách tham số)





Ví dụ

```
public static Phanso operator +(Phanso a, Phanso b)
{
    Phanso kq = new Phanso();
    kq.tuso = a.tuso * b.mauso + b.tuso * a.mauso;
    kq.mauso = a.mauso * b.mauso;
    return kq;
}
```





Đa hình động

- Là các hình thức thực hiện một phương thức trên các đối tượng khác nhau.
- Ví dụ:

```
CONNGUOI[] CN = new CONNGUOI[3];
CN[0] = new CONNGUOI("Hoa", "Nu", "HN", 15);
CN[1] = new CANBO("Hong", "Nu", "HP", 10, 3.0, 10);
CN[2] = new SINHVIEN("Hoang", "Nam", "TN", 20, 12345, 9.0);
CN[0].xuat();
CN[1].xuat();
CN[2].xuat();
```





1959
class CONNGUOI

```
{  
    các khai báo  
    public void xuat()  
    {  
        Console.WriteLine("Xuất các thông tin của CONNGUOI");  
    }  
}  
  
class CANBO : CONNGUOI  
{  
    các khai báo  
    public void xuat()  
    {  
        base.xuat(); //xuất các thông tin kế thừa từ lớp CONNGUOI  
        Console.WriteLine("Xuất thêm thông tin riêng của CANBO");  
    }  
}
```

Xuất các thông tin của CONNGUOI
Xuất các thông tin của CONNGUOI
Xuất các thông tin của CONNGUOI





Đa hình động

- Để thể hiện được tính đa hình động cần:
 - Khai báo **hàm ảo** ở lớp cha (**virtual**)
 - Ghi đè hàm đó ở lớp con (**override**)





Ví dụ về đa hình động

Khai báo hàm ảo

```
class CONNGUOI
{
    các khai báo
    public virtual void xuat()
    {
        Console.WriteLine("Xuất các thông tin của CONNGUOI");
    }
}

class CANBO : CONNGUOI
{
    các khai báo
    public override void xuat()
    {
        base.xuat(); //xuất các thông tin kế thừa từ lớp CONNGUOI
        Console.WriteLine("Xuất thêm thông tin riêng của CANBO");
    }
}
```

Ghi đè hàm ảo



Lớp trừu tượng

- Là lớp cơ sở cung cấp một phương thức giống nhau cho nhiều lớp dẫn xuất
- Phương thức chung này phải được khai báo là một phương thức trừu tượng
- Cần phải định nghĩa rõ các phương thức trừu tượng ở lớp dẫn xuất





Lớp trừu tượng

- Cú pháp khai báo lớp trừu tượng:
[quyền truy cập] **abstract class** <tên lớp>
- Cú pháp khai báo hàm trừu tượng:
[quyền truy cập] **abstract** <kiểu trả về> <tên phương thức> (ds tham số);





Lớp trừu tượng

- Ví dụ:
 - Lớp động vật có phương thức di chuyển, ăn
 - Lớp mèo có phương thức di chuyển bằng 4 chân
 - Lớp chim có phương thức di chuyển bằng cánh
 - Lớp mèo ăn thịt cá
 - Lớp chim ăn hoa quả
- => phương thức di chuyển và phương thức ăn của lớp động vật là một phương thức trừu tượng, chưa rõ ràng





Lớp trừu tượng

Khai báo lớp trừu tượng

Trong lớp trừu tượng
phải có ít nhất một
hàm trừu tượng

Trong các lớp dẫn xuất
cần phải ghi đè hàm trừu
tượng của lớp cơ sở

```
abstract class Dongvat
{
    // các khai báo khác
    public abstract void dichuyen();
}

class meo:Dongvat
{
    // các khai báo khác
    public override void dichuyen()
    {
        Console.WriteLine("Meo di chuyen bang 4 chan");
    }
}

class chim : Dongvat
{
    public override void dichuyen()
    {
        Console.WriteLine("Chim di chuyen bang canh");
    }
}
```

Hàm trừu tượng
không định nghĩa,
chỉ khai báo nên
phải có dấu ; ở cuối





Bài tập

- Tạo lớp phương tiện giao thông và các lớp dẫn xuất: ô tô, xe máy, tàu thủy, máy bay,...
- Định nghĩa phương thức di chuyển phù hợp với từng loại.
- Trong chương trình chính, tạo một phương tiện giao thông. Khi người dùng lựa chọn loại phương tiện nào thì gọi phương thức di chuyển của phương tiện đó





TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa CNTT – Bộ môn CNPM

LẬP TRÌNH WINDOWS



Giảng viên: Nguyễn Thị Phương Dung
Email: dungntp@tlu.edu.vn
SDT: 0946 079 903



Delegate và xử lý sự kiện



DELEGATE là gì?

- Delegate là kiểu dữ liệu đặc biệt, là biến kiểu tham chiếu, có khả năng lưu trữ một tham chiếu tới phương thức
- Delegates là một cơ chế hỗ trợ chung cho việc gọi phương thức gián tiếp trong khi chạy => Delegate được hiểu là Ủy quyền





Khai báo Delegate

- Khai báo Delegate trong C# quyết định các phương thức mà có thể được tham chiếu bởi Delegate đó.
- Một Delegate có thể tham chiếu tới một phương thức, mà có cùng dấu hiệu như của Delegate đó.





Ví dụ

- C# đã tạo ra một kiểu delegate có dạng như sau

```
public delegate void EventHandler(object sender, EventArgs e);
```

- Kiểu này được dùng để tham chiếu tới các phương thức có 2 tham số là object và EventArgs
- Sự kiện click của một button là một thể hiện của kiểu delegate đó (delegate có tên là EventHandler)

```
public event EventHandler Click;
```





Ví dụ

- Do vậy, khi tạo một sự kiện click cho một nút (Vd nút btSo1) ta sẽ thấy sự kiện này được tham chiếu tới một hàm có 2 tham số object và EventArgs

```
this.btSo1.Click += new System.EventHandler(this.btSo1_Click);  
  
private void btSo1_Click(object sender, EventArgs e)  
{  
}  
}
```





Khởi tạo Deletage

- Khi một kiểu delegate được khai báo, một đối tượng delegate phải được tạo với từ khóa **new** và được liên kết với một phương thức cụ thể.
- Khi tạo một delegate, tham số được truyền tới biểu thức **new** được viết tương tự như một lời gọi phương thức, nhưng không có tham số tới phương thức đó.





Ứng dụng của Delegate

- Trong lập trình C# delegate được sử dụng chính vào thực thi sự kiện (event) và các phương thức gọi sau (call-back methods).
- Để thực thi delegate trong ứng dụng cần:
 - Khai báo delegates (khai báo kiểu, khai báo biến)
 - Tạo thể hiện delegates (cho biến delegate tham chiếu tới phương thức)
 - Sử dụng delegates.





KHAI BÁO DELEGATE

**Khai báo kiểu delegate
cú pháp:**

`delegate <kiểu_trả_về> <tên_delegate> (<danh_sách_tham_số>)`

Ví dụ:

`delegate void Thongbao(string str); // tương tự khai báo phương thức
// sử dụng từ khoá delegate`

Khai báo biến delegate

`Thongbao thongbao1;`





TẠO THẺ HIỆN CHO DELEGATE

Cú pháp: new *DelegateType* (*obj.Method*)

- Biến delegate chứa phương thức và đối tượng nhận, nhưng không chứa tham số
new Thongbao(myObj.SayHello);

- Đối tượng có thể là *this* (và có thể bỏ qua)
new Thongbao(SayHello);





TẠO THẾ HIỆN CHO DELEGATE

- Phương thức có thể là *static*. Trong trường hợp này, tên của class phải được thay thế cho đối tượng.
new Thongbao (MyClass.StaticSayHello);





Tạo một thể hiện của delegate

- Dấu hiệu của phương thức phải trùng với dấu hiệu của *DelegateType*
 - số lượng tham số
 - kiểu dữ liệu của tham số (bao gồm cả kiểu trả về)
 - kiểu truyền tham số (ref, out, value)





Tạo thẻ hiện cho delegate

Tạo phương thức sẽ gán cho biến delegate

void SayHello(string str) //phương thức này phải có cùng kiểu trả về
và cùng tham số với delegate sẽ dùng nó

```
{  
    Console.WriteLine("Hello from " + str);  
}
```

Tạo thẻ hiện cho biến delegate

thongbao1 = new Thongbao(SayHello);





Sử dụng delegate

- Sử dụng delegate bằng cách đưa ra tên của delegate và truyền các tham số (nếu có).
- Sử dụng delegates tương tự như gọi một phương thức.

Ví dụ:

Lời gọi biến delegate

thongbao1("John");

// viễn dẫn phương thức SayHello("John")

//=> "Hello from John"





GÁN PHƯƠNG THỨC KHÁC NHAU

- Tất cả các phương thức phù hợp với delegate đều có thể được gán với biến delegate đó

```
void SayGoodBye(string str) {  
    Console.WriteLine("Good bye from " + sender);  
}
```

thongbao1 = new Thongbao(SayGoodBye);

thongbao1("John");// SayGoodBye("John") => "Good
bye from John"





GÁN PHƯƠNG THỨC KHÁC NHAU

Chú ý

- Biến delegate có thể được gán giá trị *null* (không có phương thức nào được gán cho nó).
- Nếu biến delegate bằng *null* thì sẽ không được gọi





Các kiểu delegate

- Delegates có hai kiểu và phụ thuộc vào yêu cầu của ứng dụng mà các kiểu của delegate được lựa chọn.
 - ◆ Single-cast delegate
 - ◆ Multicast delegate





Single-cast delegate

- Một single-cast delegate dẫn xuất từ lớp System.Delegate.
- Nó chứa tham chiếu tới chỉ một phương thức tại một thời điểm.





Multicast delegate

- Một multicast delegate dẫn xuất từ lớp System.MulticastDelegate.
- Nó chứa một lời gọi của danh sách phương thức.
- Kiểu trả về của tất cả delegates này phải là giống nhau.
- Khi một multicast delegate được gọi, nó sẽ xử lý tất cả phương thức theo thứ tự mà nó đã gán.





Multicast delegate

- Thêm phương thức vào multicast delegate sử dụng toán tử +
- Loại bỏ phương thức khỏi multicast delegate sử dụng toán tử -





Multicast delegate

- **Biến multicast delegate có thể chứa nhiều giá trị cùng một thời điểm**

Thongbao thongbao1;

thongbao1 = new Thongbao(SayHello);

thongbao1 += new Thongbao(SayGoodBye);

thongbao1("John");

// "Hello from John" "Good bye from John"





multicast delegate

- **Biến multicast delegate có thể chứa nhiều giá trị cùng một thời điểm**

Thongbao thongbao1;

thongbao1 = new Thongbao(SayHello);

thongbao1("John"); // "Hello from John"

thongbao1 += new Thongbao(SayGoodBye);

thongbao1 -= new Thongbao(SayHello);

thongbao1("John"); // "Good bye from John"





multicast delegate

- **Chú ý**

- Nếu multicast delegate là một hàm, thì sẽ trả về giá trị của hàm được tham chiếu cuối cùng.
- Vì một biến multicast delegate là 1 biến tham chiếu đến nhiều hàm. Khi gọi biến multicast delegate thực hiện thì nó sẽ lần lượt thực thi các hàm mà nó đã tham chiếu đến => kết quả sẽ là kết quả của hàm cuối cùng.





multicast delegate

- **Chú ý**

- Nếu multicast delegate có tham số ref thì tham số đó sẽ được truyền qua tất cả các phương thức.





multicast delegate

- **Chú ý**
 - Nếu multicast delegate có tham số out, tham số của lời gọi cuối cùng sẽ được trả về.





Bài tập

- Tạo 4 hàm cộng trừ nhân chia:
 - Tham số truyền vào là 2 số nguyên
 - Trả ra kết quả tương ứng với 2 số tham số truyền vào
- Tạo delegate
- Thực hiện tham chiếu đến 4 hàm trên
- Gọi và xem kết quả





bài tập 2

- Tạo một lớp Nguoi có các thông tin:
 - Tên, tuổi, chiều cao, cân nặng,
 - Phương thức so sánh tuổi, chiều cao, cân nặng giữa 2 người
- Thực hiện nhập thông tin 2 người
- Sử dụng delegate thực hiện gọi so sánh sẽ đưa ra kết quả của cả 3 phương thức so sánh tuổi, chiều cao, cân nặng của 2 người vừa nhập.





Bài tập 3

- Tạo một lớp SINHVIEN có các thông tin:
 - Tên, tuổi, lớp, điểm NNLT, điểm TinĐC,
 - Phương thức so sánh tuổi, điểm NNLT, điểm TinĐC giữa 2 SINHVIEN
- Thực hiện nhập thông tin 2 SINHVIEN
- Sử dụng delegate thực hiện gọi so sánh sẽ đưa ra kết quả của cả 3 phương thức so sánh tuổi, điểm NNLT, điểm TinĐC của 2 SINHVIEN vừa nhập.





bài tập 3

- Tạo các lớp với các thành phần sau:
 - Lớp người: tên, tuổi, giới tính, quê quán, nhập, xuất
 - Lớp động vật: tên, cân nặng, giá thành, nguồn gốc, nhập, xuất
 - Lớp xe: tên, màu sắc, nguồn gốc, giá thành, nhập, xuất
- Trong chương trình chính, tạo một menu cho phép người dùng lựa chọn làm việc với một đối tượng thuộc một trong các lớp trên.
- Sử dụng delegate để thực hiện lời gọi nhập và xuất thông tin cho đối tượng mà người dùng vừa chọn. (thực hiện 1 lời gọi nhập, 1 lời gọi xuất nhưng phải gọi đúng hàm nhập hoặc xuất của đối tượng mà người dùng chọn)





Sự kiện (event)



Giảng viên: Nguyễn Thị Phương Dung

Email: dungntp@tlu.edu.vn

SDT: 0946 079 903



EVENT - BẮT SỰ KIỆN TRÊN WINFORM

- Sự kiện là các tác động lên đối tượng trên form.
- Với mỗi tác động người dùng mong muốn thực hiện một nhiệm vụ cụ thể nào đó
- C#.Net đã thiết kế sẵn cho mỗi sự kiện một kiểu delegate tương ứng
- Lập trình viên chỉ cần xử lý các nhiệm vụ bên trong hàm mà delegate của sự kiện đó gọi đến.





BẮT SỰ KIỆN - EVENT

- Các delegate dùng để bắt sự kiện thường có dạng như sau:
delegate void SomeEvent (object sender, **SomeEventArgs** e);
- Trong đó
 - Kiểu trả về: void
 - Tham số thứ nhất: Đối tượng gửi sự kiện (kiểu *object*)
 - Tham số thứ hai: sự kiện (một đối tượng của lớp *EventArgs* tương ứng)





BẮT SỰ KIỆN - EVENT

- Ví dụ: Bắt sự kiện click cho nút btThanhtien thì sẽ khai báo:

```
this.btThanhtien.Click += new System.EventHandler(this.btThanhtien_Click);
```

Và định nghĩa hàm:

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





bài tập 3

- Tạo 3 lớp: nhà, xe và người. Trong đó mỗi lớp được mô tả với ít nhất các thành phần như sau:
 - Lớp nhà: có thông tin về diện tích sàn, có phương thức so sánh 2 nhà và viết ra nhà có diện tích lớn hơn.
 - Lớp xe: có thông tin về giá xe, có phương thức so sánh 2 xe và viết ra xe có giá cao hơn
 - Lớp người có thông tin về chiều cao, có phương thức so sánh 2 người và viết ra người cao hơn
- Tạo một form có menu cho phép chọn đối tượng làm việc:
 - Nếu chọn nhà thì hiển thị giao diện nhập thông tin về 2 nhà.
 - Nếu chọn xe thì hiển thị giao diện nhập thông tin về 2 xe.
 - Nếu chọn người thì hiển thị giao diện nhập thông tin về 2 người.
- Trên giao diện có một nút so sánh. Thực hiện so sánh 2 đối tượng đang làm việc trên giao diện bằng cách dùng **delegate**





TRƯỜNG ĐẠI HỌC THỦY LỢI

Khoa CNTT – Bộ môn CNPM

LẬP TRÌNH WINDOWS



Giang viên: Nguyễn Thị Phương Dung
Email: dungntp@tlu.edu.vn
SĐT: 0946 079 903



Lập trình Winform trong C#

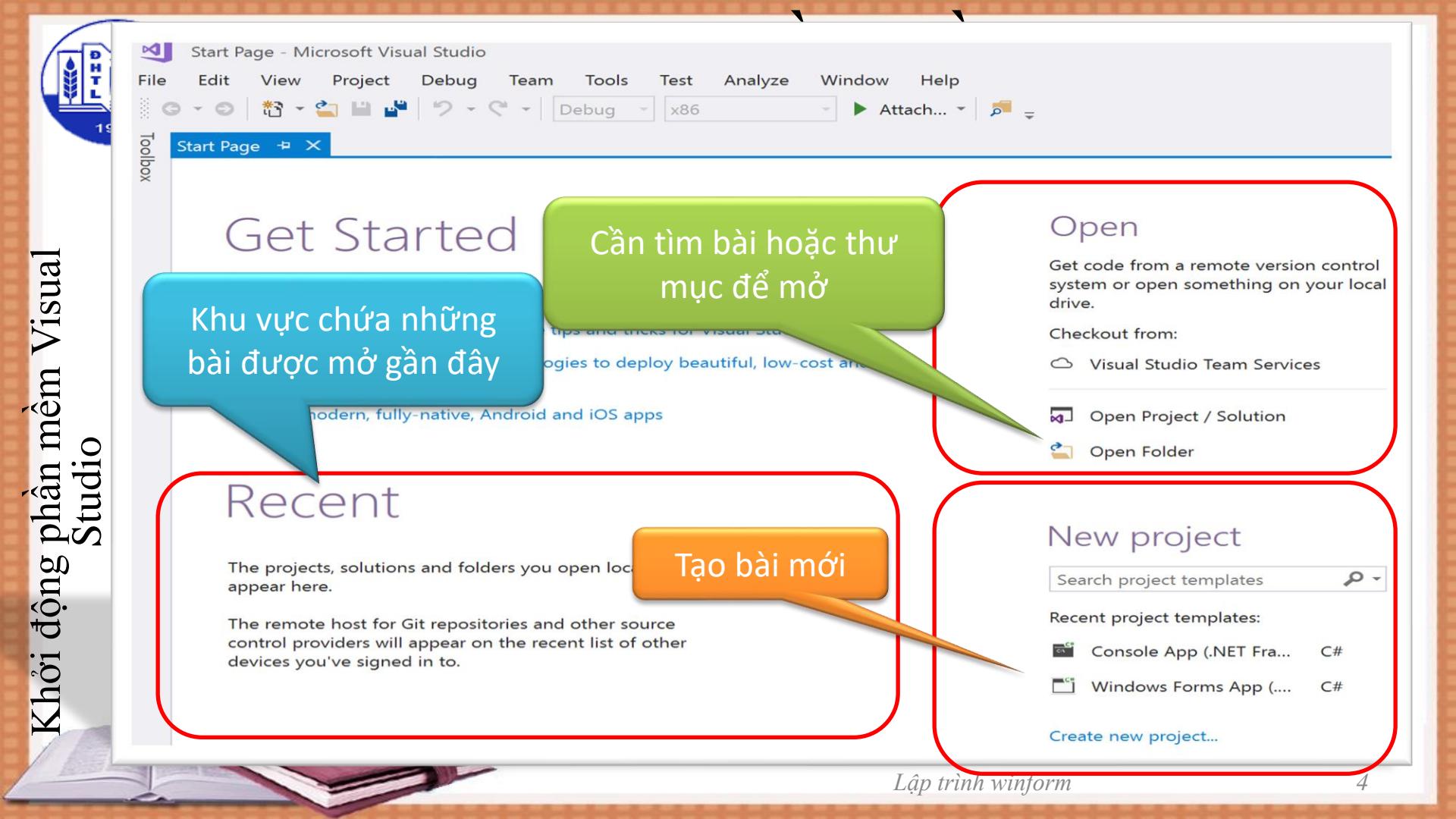


Làm quen với winform

- Tạo mới một solution/project
- Cấu trúc thư mục solution
- Giới thiệu giao diện làm việc visual studio
 - Khung thiết kế, viết code
 - Các cửa sổ phụ: solution, toolbox, properties
- Thêm các điều khiển vào form



Khởi động phần mềm Visual Studio



Tạo mới một chương trình winform



New Project

Recent

Installed

Visual C#

Console App (.NET Framework)
Windows Desktop
.NET Standard
T

Other

Other

Online

Lựa chọn Visual C# để đảm bảo chương trình được viết bằng C#

Sort by: Default



WPF App (.NET Framework)
Windows Forms App (.NET Framework)



Console App (.NET Framework)



Class Library (.NET Standard)



Class Library (.NET Framework)



Shared Project



Class Library (Legacy Portable)

Visual C#

Lựa chọn Windows Form App để tạo một chương trình chạy bằng giao diện windows

Not finding what you are looking for?

Open Visual Studio Installer

Name:

Bai 1

Location:

D:\TLU\NNLT nang cao\test\

Solution name:

Winform

Framework:

.NET Framework 4.6.1

Đặt lại tên cho project của mình (1 project là 1 bài toán nhỏ)

Check vào ô này để hệ thống tự tạo thư mục mới để lưu trữ dự án

Nhớ bấm nút Browse để lựa chọn nơi lưu trữ dự án

Đặt lại tên cho solution (1 solution là 1 chương trình lớn gồm 1 hoặc nhiều bài toán nhỏ)

Create directory for solution

Cuối cùng bấm nút OK để hoàn tất

OK

Cancel



Cấu trúc thư mục solution

> TLU > NNLT nâng cao > test > WinFormApp

Tên thư mục chứa solution

Name

Tên file solution

WinFormApp.sln

Nhapsos

Thư mục con chứa project

> test > WinFormApp > Nhapsos

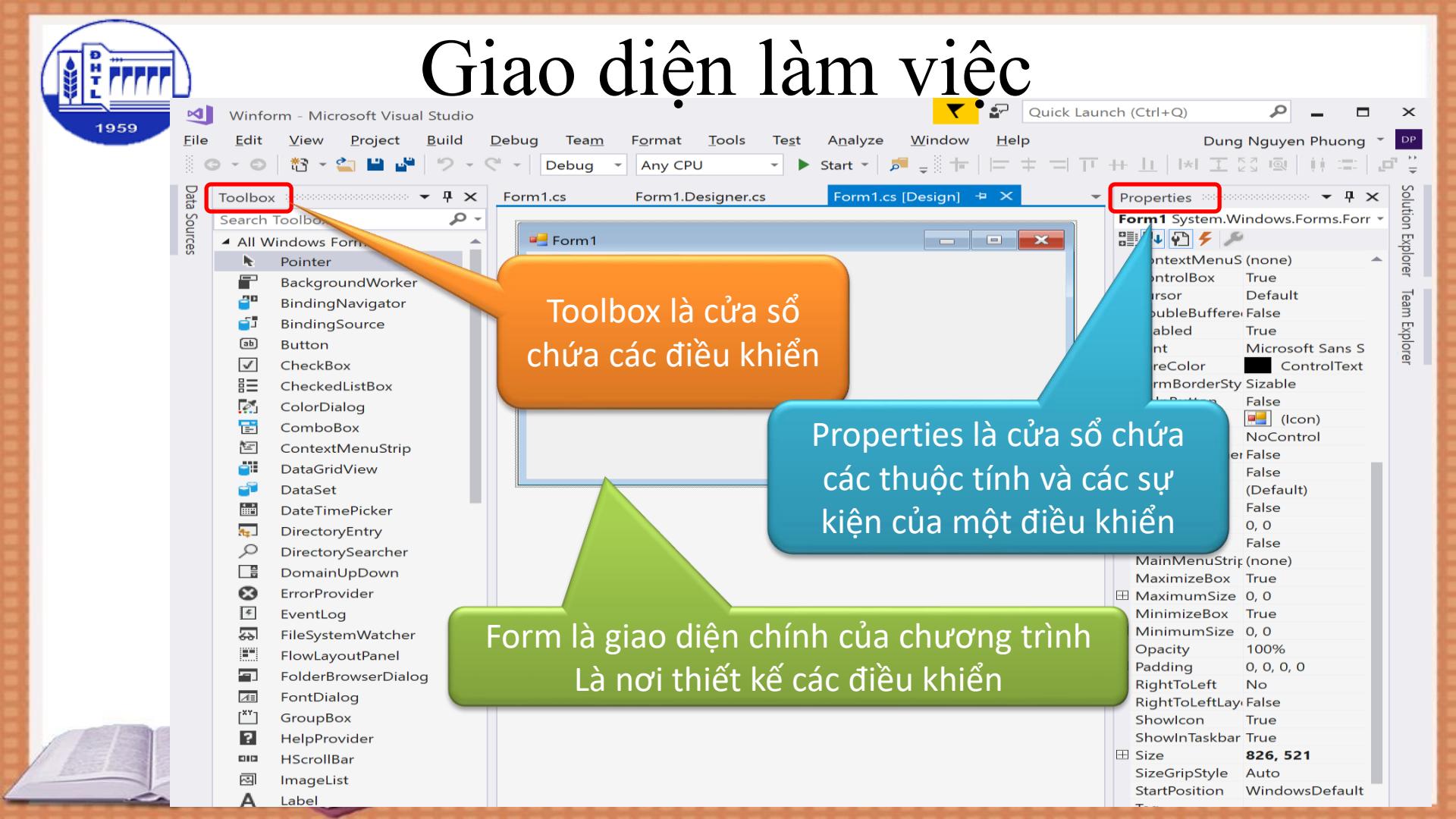
Name

- bin
- obj
- Properties
- Form1.cs
- Form1.Designer.cs
- Nhapsos.csproj
- Program.cs

Mỗi thư mục project con sẽ chứa những file mã chương trình có phần mở rộng là .cs và các thư mục



Giao diện làm việc



Toolbox

Properties

Toolbox là cửa sổ
chứa các điều khiển

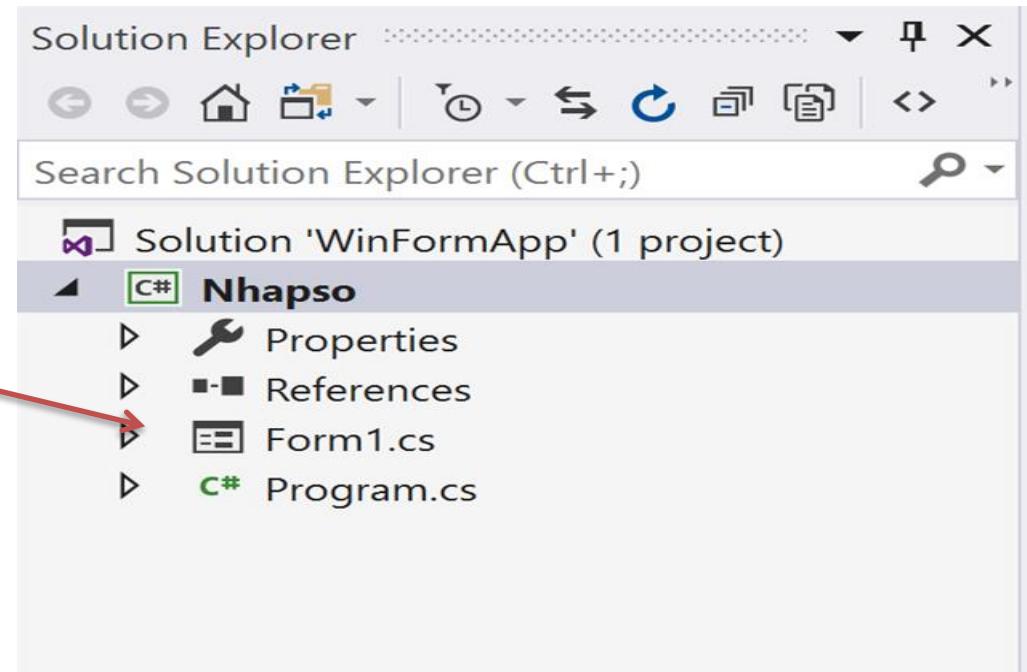
Properties là cửa sổ chứa
các thuộc tính và các sự
kiện của một điều khiển

Form là giao diện chính của chương trình
Là nơi thiết kế các điều khiển



Cửa sổ solution

- Là nơi chứa các project trong solution
- Trong mỗi project có các form
- Nháy kép chuột trái vào mỗi form sẽ mở ra khung thiết kế cho form đó





Mở các cửa sổ phụ trợ

- Để mở các cửa sổ bị tắt, lựa chọn thẻ View và chọn cửa sổ cần mở



View	Project	Build	Debug	Team	Tools	Test
Code					F7	
Solution Explorer					Ctrl+W, S	
Team Explorer					Ctrl+\, Ctrl+M	
Server Explorer					Ctrl+W, L	
SQL Server Object Explorer					Ctrl+\, Ctrl+S	
Call Hierarchy					Ctrl+W, K	
Class View					Ctrl+W, C	
Code Definition Window					Ctrl+W, D	
Object Browser					Ctrl+W, J	
Error List					Ctrl+W, E	
Output					Ctrl+W, O	
Task List					Ctrl+W, T	
Toolbox					Ctrl+W, X	
Notifications					Ctrl+W, N	
Find Results						▶
Other Windows						▶
Toolbars						▶
Full Screen					Shift+Alt+Enter	
All Windows					Shift+Alt+M	
Navigate Backward					Ctrl+-	
Navigate Forward					Ctrl+Shift+-	
Next Task						
Previous Task						
Properties Window					Ctrl+W, P	
Property Pages					Shift+F4	



Thêm các điều khiển vào form

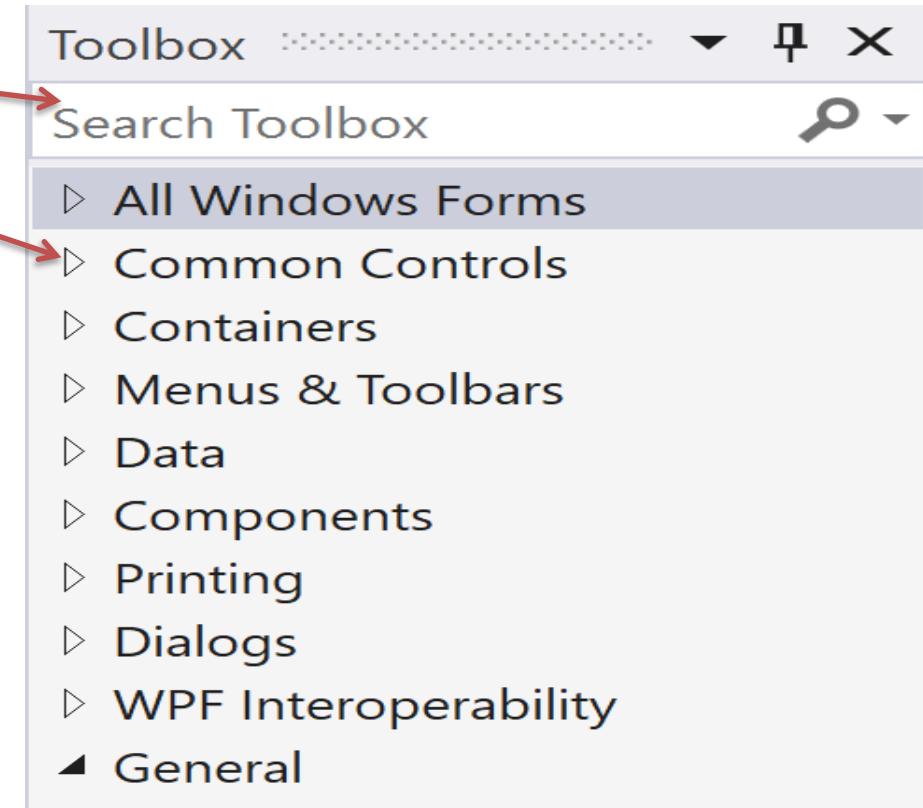
- Mở khung thiết kế
- Lựa chọn đối tượng trên cửa sổ Toolbox
- Kéo thả lên form
- Sắp xếp theo vị trí phù hợp





Cách tìm đối tượng trong ToolBox

- Gõ vào ô tìm kiếm
- Tìm trong từng nhóm





Tìm hiểu ý nghĩa của từng đối tượng

Common Controls

- Pointer
- Button
- Check Button
- Check Box
- ComboBox
- Date Time Picker

Button

Version 3.5.0.0 from Microsoft Corporation
.NET Component

Raises an event when the user clicks it.

Di chuột vào tên đối tượng sẽ xuất hiện tooltip mô tả ngắn gọn về ý nghĩa của đối tượng





Một số control thông dụng

Tên control	Mục đích sử dụng
Label	Nhãn hiển thị dữ liệu lên form
TextBox	Ô nhập dữ liệu
Button	Nút bấm
ComboBox	Lựa chọn trong một danh sách xổ xuống
CheckBox	Lựa chọn nhiều giá trị cùng lúc
RadioButton	Chỉ được lựa chọn 1 giá trị trong nhóm





Một số control thông dụng

Tên control	Mục đích sử dụng
DateTimePicker	Chọn ngày tháng năm
NumericUpDown	Lựa chọn số (tránh nhập nhầm)
RichTextBox	Nhập dữ liệu trên nhiều dòng
ListView	Hiển thị danh sách các mục
DataGridView	Hiển thị danh sách dưới dạng bảng





Một số control thông dụng

Tên control	Mục đích sử dụng
MenuStrip	Thực đơn
ContextMenuStrip	Thực đơn cho từng đối tượng, khi bấm chuột phải vào đối tượng đó
ToolTip	Hiển thị ghi chú khi di chuột vào đối tượng





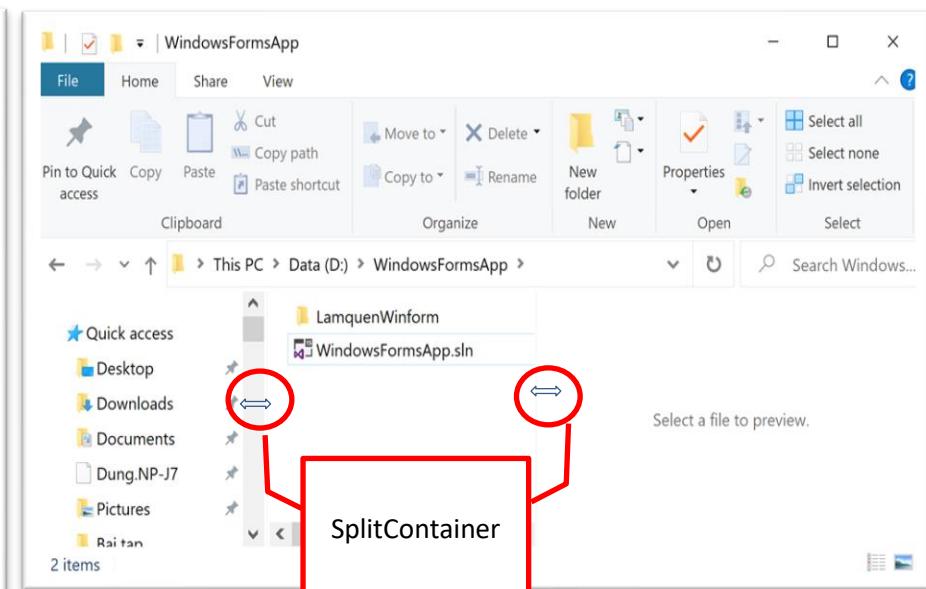
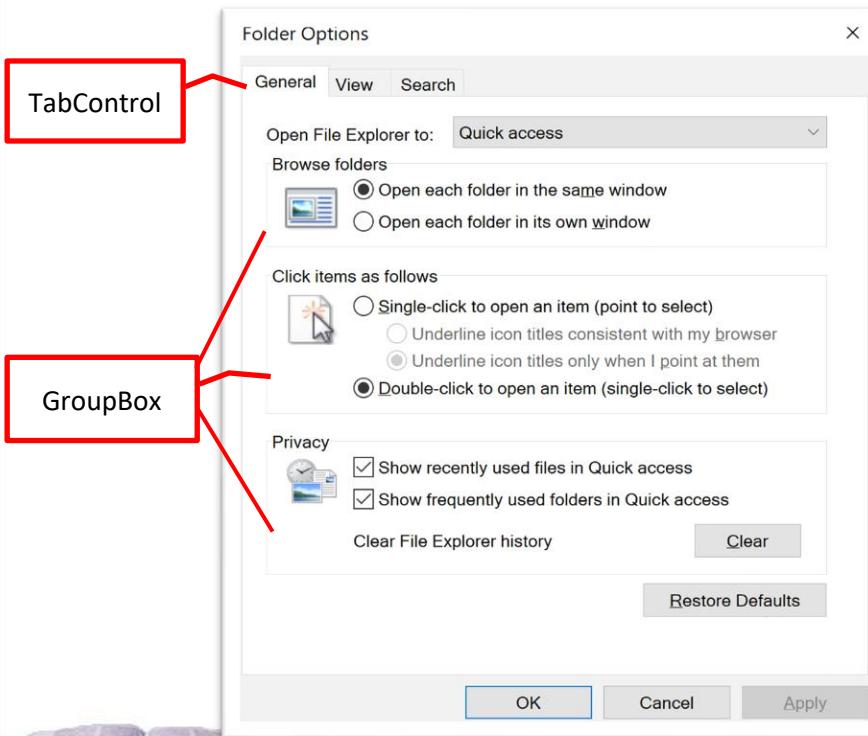
Một số control chứa thông dụng

Tên control	Mục đích sử dụng
GroupBox	Tạo nhóm có tiêu đề
Panel	Tạo nhóm, không có tiêu đề
SplitContainer	Tạo 1 nhóm gồm 2 panel, có thể thay đổi kích cỡ
TabControl	Tạo đối tượng chứa dưới dạng các thẻ
TableLayoutPanel	Thùng chứa ở dạng bảng
FlowLayoutPanel	Thùng chứa tự sắp đổi tương





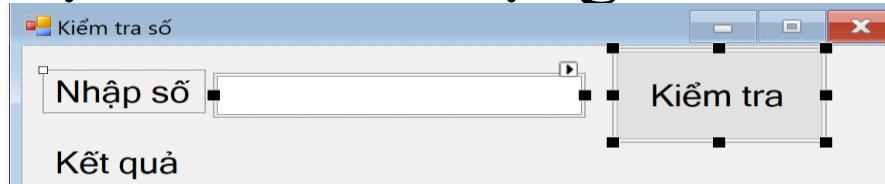
Ví dụ sử dụng thùng chứa





Sắp xếp các đối tượng trên giao diện

- Lựa chọn các đối tượng cần căn lề.



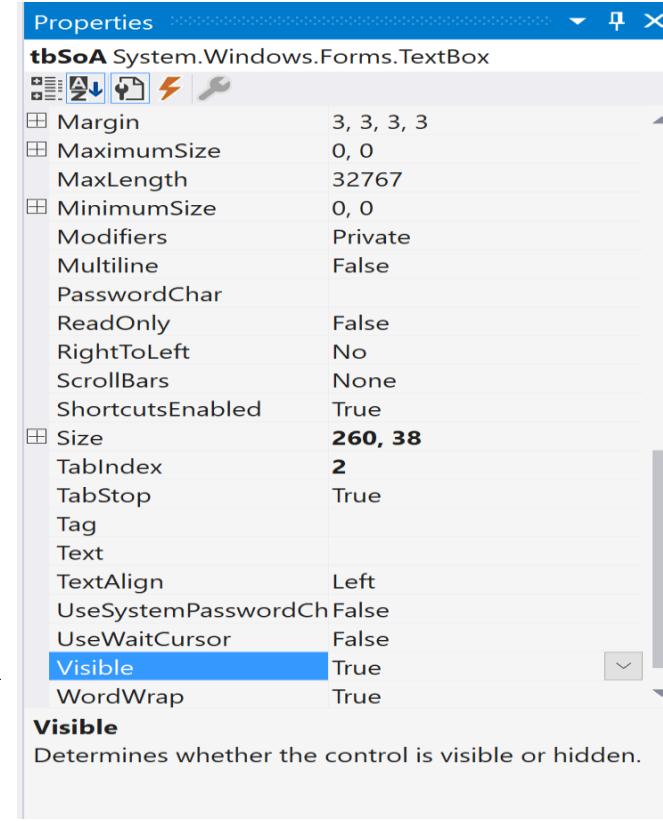
- Sử dụng nút lệnh căn lề trên thanh công cụ





Thiết lập thuộc tính cho các control

- Sử dụng cửa sổ Properties:
 - Chọn Control cần thay đổi thuộc tính.
 - Trên cửa sổ Properties, chọn thuộc tính cần thay đổi và gán lại giá trị thích hợp.
 - Lựa chọn vào thuộc tính nào trong cửa sổ thì phía dưới cửa sổ xuất hiện mô tả ngắn gọn về thuộc tính đó





Thiết lập thuộc tính cho các control

- Sử dụng mã lệnh:
 - Cú pháp: **ControlObject.PropertyName** để truy nhập vào thuộc tính của đối tượng control. Trong đó:
 - **ControlObject** là một thể hiện của Control
 - **PropertyName** là một thuộc tính của control
 - Vd: **tb_Xinchao.Text = "Hello";**





Một số thuộc tính của form

Tên	Ý nghĩa
Name	Tên của Control, đặt theo quy tắc đặt tên biến
Text	Là xâu ký tự hiển thị trên thanh tiêu đề của giao diện
ControlBox	Xác định sự xuất hiện của các nút điều khiển của form
MaximizeBox	Xác định nút max có được hiển thị hay không
MinimizeBox	Xác định nút min có được hiển thị hay không
MaximumSize	Xác định kích thước lớn nhất có thể thay đổi
MinimumSize	Xác định kích thước nhỏ nhất có thể thay đổi
StartPosition	Thiết lập vị trí bắt đầu hiển thị của form
BackColor	Thiết lập màu nền của form
BackgroundImage	Thiết lập ảnh nền của form



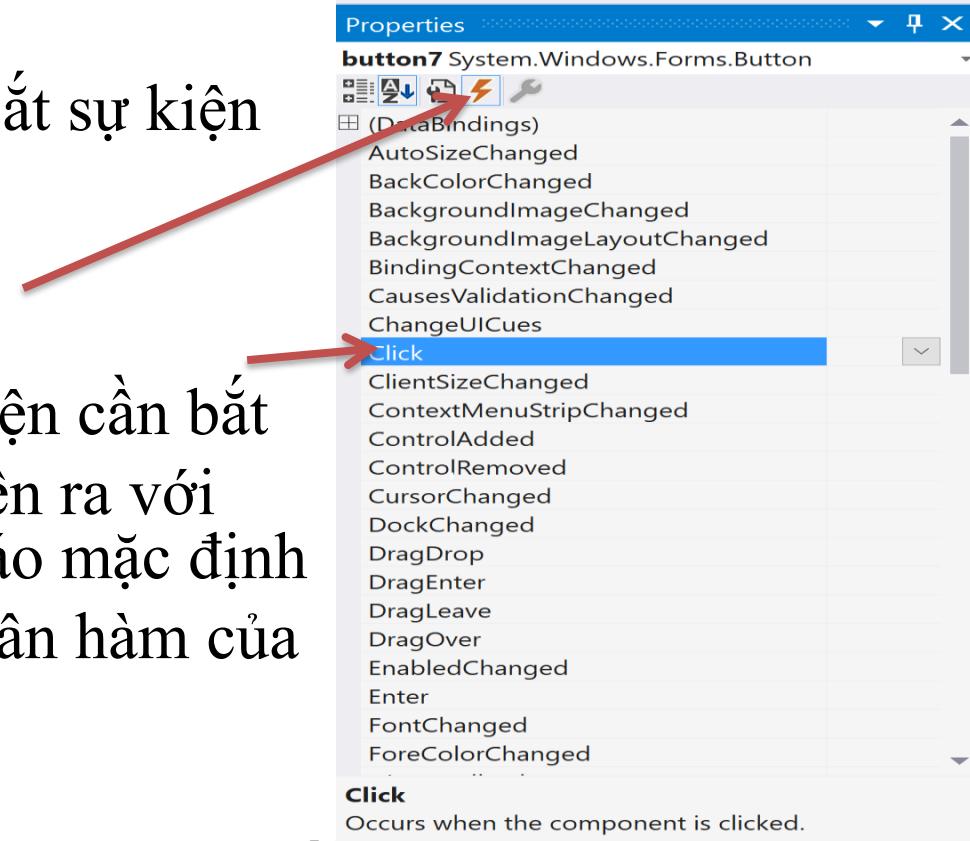
Một số thuộc tính chung của control

Tên	Ý nghĩa
Name	Tên của Control, đặt theo quy tắc đặt tên biến
Text	Là xâu ký tự hiển thị lên giao diện
Anchor	Bám Control theo các cạnh của đối tượng chứa, đảm bảo luôn giữ một khoảng cách cho trước với cạnh của đối tượng chứa.
Dock	Bám Control sát theo các cạnh của đối tượng chứa Nếu Dock = Top/ Left/Bottom/Right: bám sát theo 3 cạnh liên quan Nếu Dock = Fill: mở rộng Control ra toàn bộ khoảng trống của đối tượng chứa
Enabled	Để xác định khả năng tương tác của Control
Visible	Xác định khả năng nhìn thấy control vào lúc chạy chương trình.
TabIndex và TabStop	Xác định thứ tự nhận focus của control khi người sử dụng nhấn phím Tab. Nếu không muốn control nhận focus khi người sử dụng dùng phím Tab, gán thuộc tính TabStop của control đó bằng false.



Bắt sự kiện cho các đối tượng

- Lựa chọn đối tượng cần bắt sự kiện trên form
 - Mở cửa sổ properties
 - Chọn danh sách sự kiện
 - Nháy kép chuột vào sự kiện cần bắt
 - Cửa sổ viết code được hiện ra với một tên hàm được khai báo mặc định
 - Viết code cần thiết vào thân hàm của sự kiện





Một số sự kiện mặc định khi nháy kép chuột trái vào control

Kiểu control	Tên sự kiện mặc định	Ý nghĩa khi chạy chương trình
Form	Load	Sự kiện xảy ra ngay khi form được chạy lên
Button	Click	Sự kiện xảy ra khi bấm chuột lên nút
TextBox	TextChanged	Sự kiện xảy ra khi thay đổi nội dung của textbox
CheckBox	CheckedChanged	Khi thay đổi lựa chọn trong checkbox
ComboBox	SelectedIndexChanged	Khi thay đổi lựa chọn trong combobox
DatagridView	CellContentClick	Khi bấm chuột vào datagridview
NumericUpDown	ValueChanged	Khi thay đổi giá trị trong NumericUpDown





Xóa sự kiện

- Tuyệt đối không xóa trực tiếp đoạn mã lệnh tự sinh cho mỗi sự kiện
- Cần vào cửa sổ Properties, chọn danh sách các sự kiện, chọn vào sự kiện cần xóa và xóa liên kết tới sự kiện tại đây.





Bài tập



Xây dựng một chương trình Window Form để thực hiện:

- Nhập 2 số a và b
- Xuất ra tổng, hiệu, tích, thương của 2 số đó.

Yêu cầu:

- Form không có nút thu nhỏ và phóng to cửa sổ
- Khi chạy lên, form đứng giữa màn hình
- 2 nút Tính và Thoát tự động đứng sát lề bên phải của form khi form thay đổi kích thước
- Các TextBox tự động co giãn theo kích cỡ của form
- Chương trình cho phép sử dụng bàn phím thay vì chuột.
 - Khi ấn nút tab trên bàn phím thì chuyển điều khiển lần lượt từ các control số a, số b, nút Tính và nút Thoát
 - Khi ấn phím enter thì nút Tính được thực hiện
 - Khi ấn phím ESC thì nút Thoát được thực hiện





Một số thuộc tính/ sự kiện đặc trưng của một số control



Form

Thuộc tính	Ý nghĩa
Name	Tên của form
Text	Tiêu đề của form (hiển thị trên thanh tiêu đề)
BackColor	Thiết lập màu nền của form
BackgroundImage	Chọn ảnh nền cho form
BackgroundImageLayout	Bố cục để ảnh nền của form
FormBorderStyle	Thiết lập kiểu viền của form
Opacity	Thiết lập độ mờ của form (để tỉ lệ càng nhỏ form càng trong suốt)





Form

Thuộc tính	Ý nghĩa
Font	Thiết lập font chữ cho form và những điều khiển kế thừa từ lớp form
ForeColor	Thiết lập màu chữ
ControlBox	Thiết lập hộp điều khiển có xuất hiện hay không (Hộp điều khiển là các nút thu nhỏ, phóng to, đóng cửa sổ ở góc trên bên phải cửa sổ)
MaximizeBox	Cho phép nút phóng to cửa sổ có xuất hiện hay không
MinimizeBox	Cho phép nút thu nhỏ cửa sổ có xuất hiện hay không
StartPosition	Xác định vị trí của cửa sổ khi xuất hiện lần đầu trên màn hình
WindowState	Xác định trạng thái hiển thị form lần đầu





Form

Thuộc tính	Ý nghĩa
AutoScroll	Tự động xuất hiện thanh cuộn khi form bị thu nhỏ, che mất một số control
AutoSize	Chỉ cho phép co giãn cửa sổ đến mức tối thiểu bằng khung chứa các điều khiển hiện tại (không cho phép thu nhỏ đến mức xuất hiện thanh cuộn)
AutoSizeMode	Thiết lập kiểu form có được thay đổi kích cỡ hay không GrowOnly: Được thay đổi GrowAndShrink: Không được thay đổi





Form

Thuộc tính	Ý nghĩa
AutoSizeMode	Tự động thay đổi kích cỡ của control theo tỉ lệ của font chữ hoặc độ phân giải màn hình Font: tỉ lệ theo kích thước font chữ Dpi: tỉ lệ theo độ phân giải màn hình Inherit: tỉ lệ theo lớp cha (nếu có)
Size	Thiết lập kích cỡ cho form
MaximumSize	Thiết lập kích cỡ tối đa cho form
MinimumSize	Thiết lập kích cỡ tối thiểu cho form





Form

Thuộc tính	Ý nghĩa
AcceptButton	Nút mặc định kích hoạt khi ấn phím Enter
CancelButton	Nút mặc định kích hoạt khi ấn phím ESC
ShowIcon	Có hiển thị biểu tượng Icon ở góc trên bên trái cửa sổ hay không
ShowInTaskbar	Thiết lập biểu tượng chương trình khi chạy có xuất hiện ở thanh Taskbar hay không





Button

Sự kiện	Ý nghĩa
Click	<p>Sự kiện mặc định khi nháy kép chuột vào textbox trên màn hình thiết kế.</p> <p>Các lệnh viết trong sự kiện này sẽ được thực hiện khi bấm chuột vào nút trên giao diện chương trình</p>





Label

Thuộc tính	Ý nghĩa
AutoSize	True: Thiết lập chế độ tự động co giãn kích cỡ theo nội dung False: không co giãn kích cỡ theo nội dung, tự động xuống dòng khi nội dung quá dài
TextAlign	Thiết lập kiểu căn lề cho văn bản trong label





TextBox

Thuộc tính	Ý nghĩa
AutoSize	True: Thiết lập chế độ tự động co giãn kích cỡ theo nội dung False: không co giãn kích cỡ theo nội dung, tự động xuống dòng khi nội dung quá dài
ReadOnly	Thiết lập chế độ chỉ đọc cho đối tượng (không được sửa)
TextAlign	Thiết lập kiểu căn lề cho văn bản trong TextBox
PasswordChar	Thiết lập ký tự thay thế khi gõ mật khẩu
Multiline	Cho phép xuất hiện nhiều dòng nếu nội dung trong ô quá dài
ScrollBars	Cho phép xuất hiện thanh cuộn nếu nội dung dài hơn kích cỡ của ô
WordWrap	Thiết lập chữ trong ô tự động xuống dòng





TextBox

Sự kiện	Ý nghĩa
TextChanged	<p>Sự kiện mặc định khi nháy kép chuột vào textbox trên màn hình thiết kế.</p> <p>Các lệnh viết trong sự kiện này sẽ được thực hiện khi gõ văn bản vào Textbox</p>





CheckBox / CheckedListBox / RadioButton

Thuộc tính	Ý nghĩa
Checked	True: Ô lựa chọn đang được đánh dấu False: Ô lựa chọn không được đánh dấu
CheckState	Các trạng thái lựa chọn <ul style="list-style-type: none">- Checked: Có lựa chọn- UnChecked: Không lựa chọn- Indeterminate: Không xác định
ThreeState	Cho phép lựa chọn 3 trạng thái hay không





CheckBox / CheckedListBox / RadioButton

Sự kiện	Ý nghĩa
CheckedChanged	Sự kiện mặc định khi nháy kép chuột vào đối tượng trên màn hình thiết kế. Thực hiện khi thay đổi trạng thái lựa chọn





ComboBox

Thuộc tính	Ý nghĩa
Items	Tập hợp các lựa chọn
Sorted	Bật/tắt chế độ sắp xếp
DataSource	Thiết lập nguồn dữ liệu dạng DataTable
DisplayMember	Thiết lập trường dữ liệu hiển thị trong danh sách xổ xuống
ValueMember	Thiết lập trường dữ liệu giá trị ẩn tương ứng với dòng dữ liệu của DisplayMember
FormatString	Cho phép thiết lập định dạng hiển thị với một số loại dữ liệu
DropDownHeight	Độ cao khung xổ xuống
DropDownWidth	Độ rộng khung xổ xuống





ComboBox

Sự kiện	Ý nghĩa
SelectedIndexChanged	Sự kiện mặc định khi nháy kép chuột vào combobox trên màn hình thiết kế. Thực hiện khi thay đổi vị trí lựa chọn





Date Time Picker

Thuộc tính	Ý nghĩa
Format	Kiểu hiển thị
MaxDate	Ngày tháng tối đa
MinDate	Ngày tháng tối thiểu
Value	Giá trị hiện tại





Date Time Picker

Sự kiện	Ý nghĩa
ValueChanged	Sự kiện mặc định khi nháy kép chuột vào DateTimePicker trên màn hình thiết kế. Thực hiện khi thay đổi lựa chọn ngày tháng





DataGrid View

Thuộc tính	Ý nghĩa
Columns	Thiết lập các cột hiển thị trong DataGrid View
AutoSizeColumnsMode	Thiết lập độ rộng các cột (phủ kín độ rộng của DataGrid View nếu để là Fill)
MultiSelect	Cho phép được lựa chọn nhiều dòng hay không
SelectionMode	Thiết lập kiểu lựa chọn
ReadOnly	Cho phép chỉ đọc hoặc được sửa
DataSource	Thiết lập nguồn dữ liệu hiển thị vào DataGrid View





DataGridView

Sự kiện	Ý nghĩa
CellContentClick	<p>Sự kiện mặc định khi nháy kép chuột vào DataGridView trên màn hình thiết kế.</p> <p>Thực hiện khi bấm chuột vào nội dung mỗi ô trong DataGridView</p>
CellClick	<p>Sự kiện thực hiện khi lựa chọn từng dòng dữ liệu trong DataGridView (kể cả khi bấm chuột vào vùng trắng, không phải là dữ liệu)</p> <p>Sự kiện này thường được dùng để xác định vị trí dòng đang lựa chọn trên DataGridView (e.RowIndex)</p>





Bài tập



Tạo một form cho phép nhập danh sách sinh viên :

- Nhập thông tin của sinh viên: Mã SV, Họ tên, Giới tính(chỉ cho chọn 1 trong 2 giới), Quê quán (sử dụng Combobox để lựa chọn)
- Có nút Thêm để thêm dữ liệu vào một DataGridView
- Khi bấm vào DataGridView thì hiển thị dữ liệu tại dòng đang bấm lên các Control tương ứng
- Có nút Sửa để cập nhật dữ liệu vừa sửa vào DataGridView
- Có nút Xóa để xóa một dòng dữ liệu đang chọn tại DataGridView

Họ tên	Ngày sinh	Giới tính	Quê quán	Lớp	Khoa
*					





Các lớp hộp thoại và cách dùng



Các lớp hộp thoại trong C#

- ColorDialog: hiển thị các màu và trả về màu được chọn
- FontDialog: hiển thị các font và trả về font, kích thước, kiểu chữ được chọn
- OpenFileDialog: cho phép người sử dụng duyệt file và thư mục, sau đó chọn một hoặc nhiều file.
- PageSetupDialog: cho phép người sử dụng lựa chọn các thiết lập liên quan tới cách bố trí trang.





Các lớp hộp thoại trong C#

- PrintDialog: cho phép người sử dụng lựa chọn các vấn đề liên quan đến in và gửi tài liệu tới máy in đã chọn.
- PrintPreviewDialog: cho phép người sử dụng xem file trước khi in.
- SaveFileDialog: cho phép người sử dụng duyệt file và thư mục, sau đó chọn file cần lưu.
- MessageBox: cho phép người dùng hiển thị một thông báo với 12 cách hiển thị khác nhau





Phân loại lớp hộp thoại

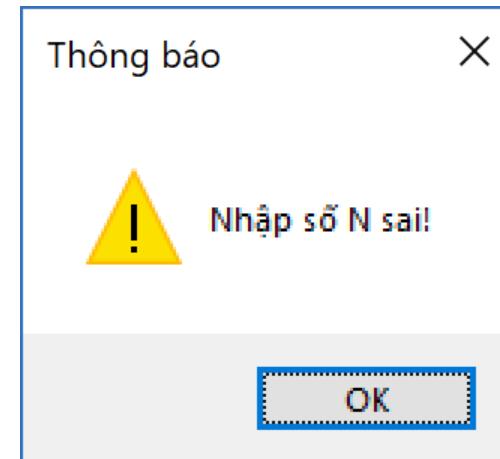
- Hộp thoại được chia thành 2 loại:
 - Modeless - Show()
 - Modal - ShowDialog()





Phân loại hộp thoại

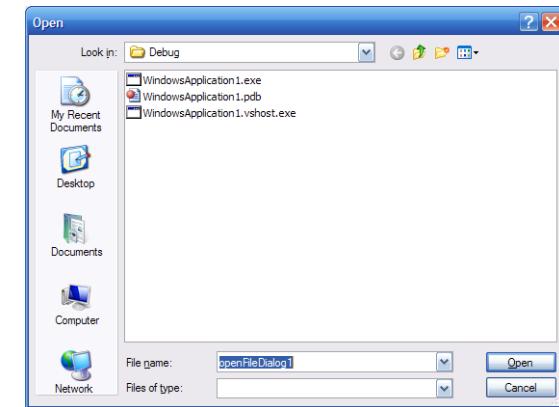
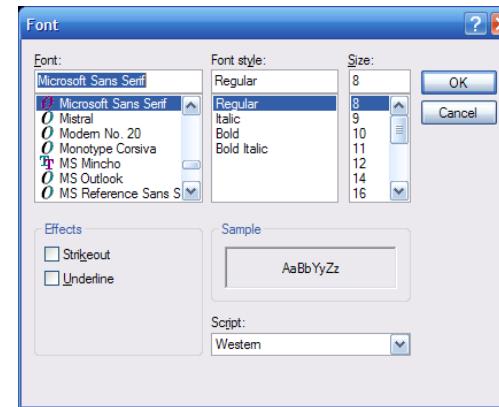
- Modeless - Show(): là loại hộp thoại thông báo, không cần kết quả trả về
- VD:





Phân loại hộp thoại

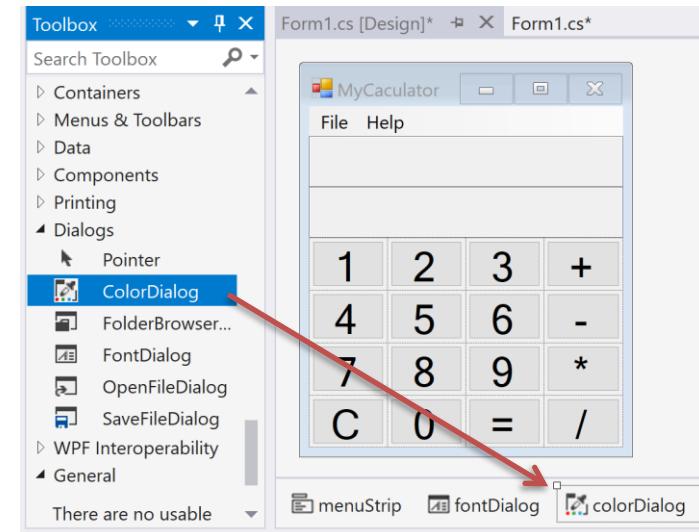
- Modal - ShowDialog(): là hộp thoại cần lấy kết quả trả về như: OK, Cancel,...
- VD:





Sử dụng hộp thoại

- VD: Sử dụng hộp thoại Color
 - Trên giao diện, kéo thả công cụ ColorDialog vào form và đặt tên cho đối tượng là ColorDialog





Sử dụng hộp thoại

- Thực hiện việc mở hộp thoại Color ra để lấy màu bằng câu lệnh như sau:

```
if(colorDialog.ShowDialog()==DialogResult.OK)
```

```
    bt1.BackColor = colorDialog.Color;
```





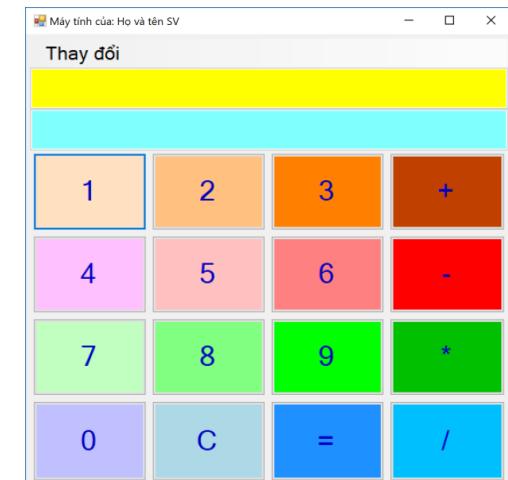
Bài tập



Hãy viết một chương trình mô phỏng chiếc máy tính cầm tay với giao diện như sau:

Yêu cầu:

- Tiêu đề của giao diện ghi: “**Máy tính của: < Họ và tên SV >**”
- Có một ô TextBox hiển thị kết quả, một ô TextBox hiển thị biểu thức
- Các TextBox có khả năng thay đổi độ dài theo form
- Các nút có kích cỡ bằng nhau và có khả năng thay đổi kích cỡ theo form
- Các nút trên giao diện hoạt động tương tự như máy tính cầm tay.
- Khi bấm vào menu “**Thay đổi màu sắc**” thì toàn bộ các nút Button sẽ đổi màu theo màu người dùng chọn
- Khi bấm vào menu “**Thay đổi font chữ**” thì toàn bộ font chữ của tất cả các đối tượng sẽ thay đổi theo font mà người dùng chọn





TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa CNTT – Bộ môn CNPM

LẬP TRÌNH WINDOWS



Giảng viên: Nguyễn Thị Phương Dung
Email: dungntp@tlu.edu.vn
SDT: [0946 079 903](tel:0946079903)



ADO.Net

(ActiveX Data Object.NET)



Giới thiệu về ADO.Net

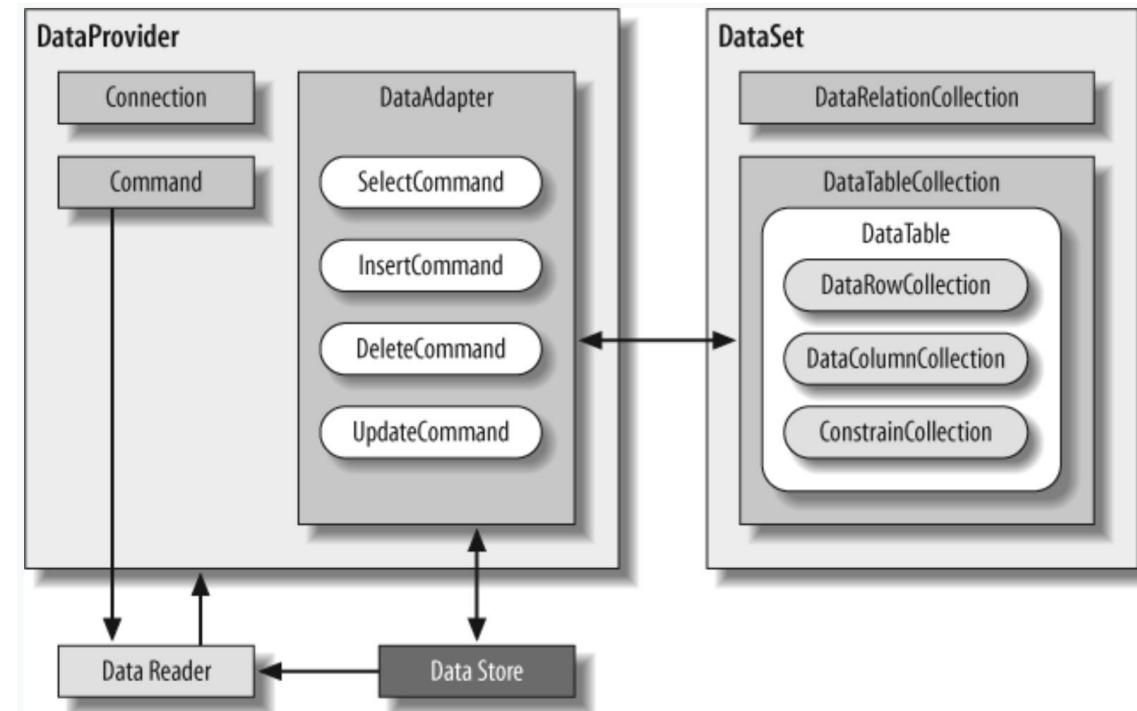
- Là một bộ thư viện hướng đối tượng (OOP) cho phép tương tác với dữ liệu nguồn.
- Dữ liệu nguồn thường là các cơ sở dữ liệu như: SQLServer, MySQL, Oracle Database,...
- Dữ liệu nguồn cũng có thể là các file text, XML, hoặc excel





Kiến trúc của ADO.Net

- ADO.Net được chia làm 2 phần rời rạc có thể sử dụng độc lập hoặc đồng thời:
 - DataProvider
 - DataSet





DataProvider

- Là các thư viện lớp cung cấp chức năng tạo kết nối đến nguồn dữ liệu, thi hành các lệnh trên nguồn dữ liệu đó.
- Mỗi thư viện hỗ trợ kết nối tới một loại cơ sở dữ liệu khác nhau.





Một số DataProvider phổ biến

Data Provider	API prefix	Loại cơ sở dữ liệu hỗ trợ
ODBC Data Provider	Odbc	Open Database Connectivity
OleDb Data Provider	OleDb	Object Linking and Embedding, Database. VD: Access hoặc Excel.
Oracle Data Provider	Oracle	Oracle Databases.
SQL Data Provider	Sql	Microsoft SQL Server.
Borland Data Provider	Bdp	Sử dụng chung cho nhiều loại cơ sở dữ liệu khác: Interbase, SQL Server, IBM DB2, and Oracle.





Các thành phần của DataProvider

- Bao gồm:
 - Connection: làm nhiệm vụ kết nối tới cơ sở dữ liệu
 - Command: thực hiện các thao tác với CSDL như; select, insert, update, delete
 - DataReader: cung cấp việc đọc từng dòng dữ liệu theo chiều tiến từ đầu đến cuối.
 - DataApdater: đóng vai trò như là cầu nối giữa Dataset và CSDL, tải dữ liệu lên dataset hoặc đồng bộ các thay đổi ở dataset về lại CSDL





DataSet

- Là các thư viện lớp tạo các đối tượng để quản lý dữ liệu không phụ thuộc vào nguồn dữ liệu.
- Có thể được coi là một cơ sở dữ liệu sao chép của cơ sở dữ liệu nguồn.
- Bao gồm nhiều DataTable





Kết nối cơ sở dữ liệu SQL Server với winform



Các đối tượng cần dùng

- SqlConnection: kết nối dữ liệu
- SqlCommand: thực hiện các câu lệnh sql
- SqlDataAdapter: truy vấn dữ liệu
- SqlDataReader: đọc dữ liệu
- DataTable: chứa dữ liệu được truy vấn ra





SqlConnection

- SqlConnection biểu diễn một kết nối liên tục tới nguồn dữ liệu SQL Server
- Trong đó chuỗi kết nối – ConnectionString, xác định server mà đối tượng SqlConnection sẽ sử dụng bao gồm 3 thuộc tính sau:
 - DataSource – xác định tên của server muốn kết nối tới.
 - Initial Catalog – xác định tên của CSDL sử dụng trên server.
 - Integrated Security – chế độ bảo mật.
 - Nếu khi kết nối với SQL server bằng chế độ xác thực của Windows thì Integrated Security = true.
 - Ngược lại, khi kết nối bằng chế độ xác thực của SQL Server thì phải ghi rõ tên đăng nhập và mật khẩu vào 2 thuộc tính UserID và Password (UserID=username;Password=password)





Ví dụ tạo một kết nối SqlConnection

- Ví dụ:

Khai báo biến bên
ngoài các hàm để dùng
chung trong nhiều
hàm (biến toàn cục)

```
string chuoiketnoi = "Data Source=LAPTOP-CJHIRH4S\\SQLEXPRESS; " +  
    " Initial Catalog=QLSV; " +  
    " Integrated Security=True";  
  
SqlConnection conn = null;  
  
private void Form1_Load(object sender, EventArgs e)  
{  
    conn = new SqlConnection(chuoiketnoi);  
    conn.Open();
```

Khởi tạo và Open biến
SqlConnection trong
sự kiện Load form để
đảm bảo form được
kết nối CSDL ngay khi
chương trình chạy

- Để lấy thông tin cho chuỗi kết nối, có thể thực hiện
theo 1 trong 2 cách sau:



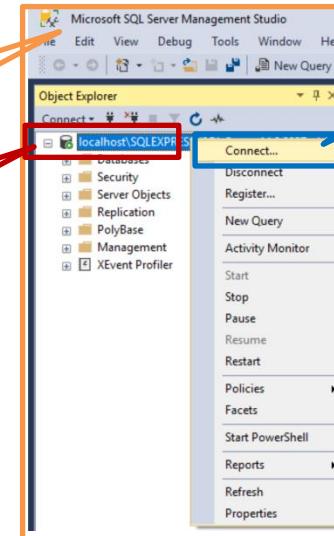


Cách 1:

- Tự viết chuỗi kết nối bằng tay => phải chú ý viết đúng chính tả
- Để lấy Data Source, thực hiện theo các bước sau:

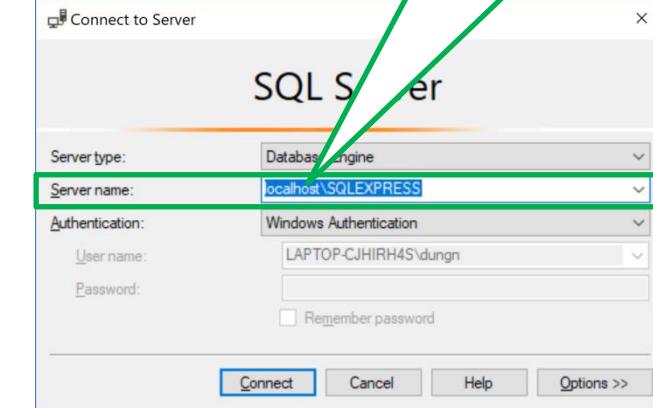
B1: Vào cửa sổ Microsoft SQL Server Management Studio

B2: Bấm chuột phải vào tên server



B3: Chọn Connect...

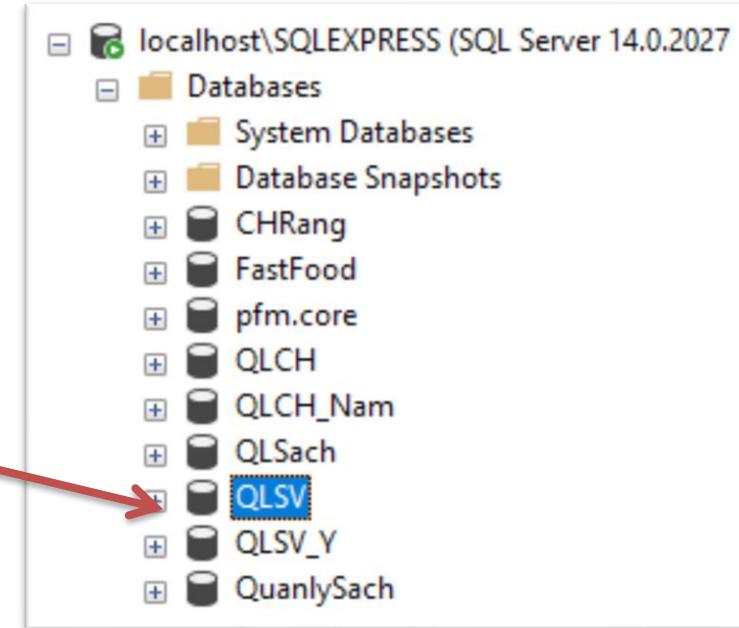
B4: Server name chính là giá trị sẽ gán cho thuộc tính Data Source





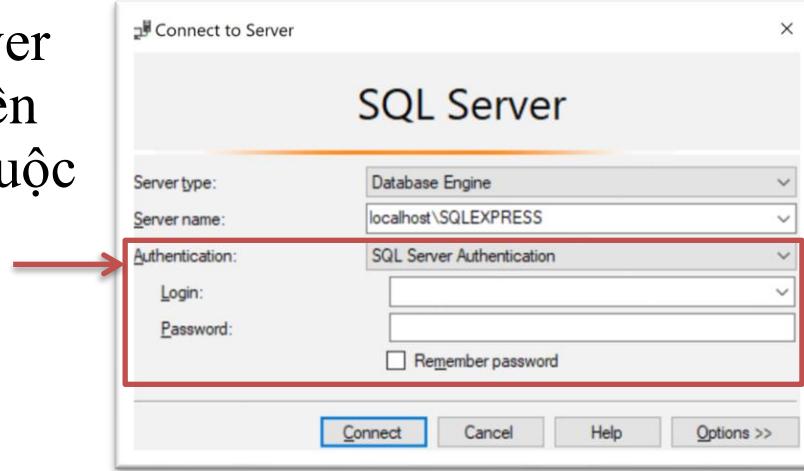
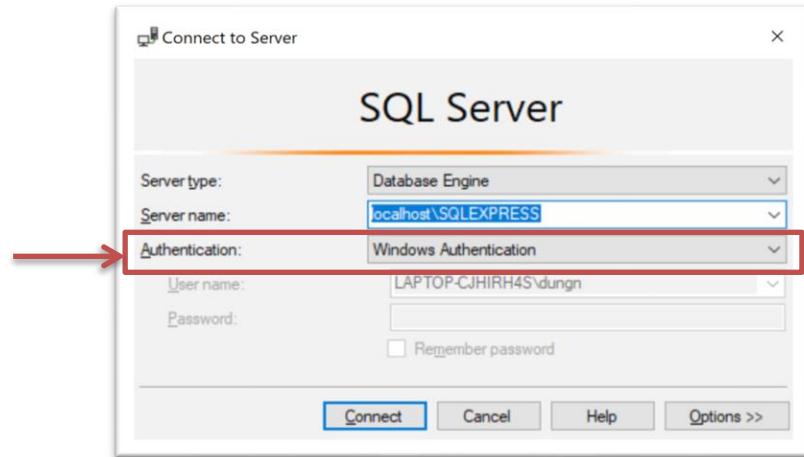
- Initial catalog:

Chính là tên của cơ sở dữ liệu cần kết nối (như trong ví dụ minh họa, Initial catalog = QLSV)





- Integrated Security:
- Thuộc tính này cần chú ý:
 - Nếu Authentication là Windows Authentication thì thiết lập Integrated Security = true.
 - Nếu Authentication là SQL Server Authentication thì phải viết rõ tên đăng nhập và mật khẩu vào 2 thuộc tính UserID và Password (UserID=username;Password=password)





Cách 2

- Thực hiện tạo đối tượng kết nối trong chương trình và copy chuỗi kết nối của đối tượng đó
- Các bước thực hiện:





B1: Mở cửa sổ SQL Server Object Explorer

The screenshot shows the SQL Server Object Explorer window. At the top, it displays the connection path: SQL Server \ (localdb)\MSSQLLocalDB \ (localdb)\ProjectsV13 \ SQL Server. Below this, there are several database nodes: Databases (System Databases, CHRang, FastFood, pfm.core, QLCH, QLCH_Nam, QLSach), Security, Server Objects, and Projects - WinFormApp. A red box highlights the 'QLSV' database node under the 'Databases' section.

B2: Chọn Database cần làm việc

The screenshot shows the Properties window for the 'QLSV' database. The left pane lists various properties like ANSI NULL Default, ANSI NULLS Enabled, etc. The right pane shows the 'Connection string' property, which is currently set to 'Data Source=LAPTOP-CJHIRH4S\SQLEXPRESS;Initial Catalog=QLSV;Integrated Security=True;Conn...'. A red box highlights this 'Connection string' entry. A callout bubble points to this box with the text: 'Lựa chọn thuộc tính Connection string' (Select the Connection string property) and 'Copy những dữ liệu cần dùng trong thuộc tính này' (Copy the data needed in this property).



SqlDataAdapter

- Được dùng để truy vấn dữ liệu
- Sử dụng kết hợp với DataTable để chứa dữ liệu
- Ví dụ:

```
private void Form1_Load(object sender, EventArgs e)
{
    conn = new SqlConnection(chuoiketnoi);
    conn.Open();
    string sql = "Select * from SV";
    SqlDataAdapter daSV = new SqlDataAdapter(sql, conn);
    DataTable dtSV = new DataTable();
    daSV.Fill(dtSV);
    dgvSV.DataSource = dtSV;
}
```





Ví dụ sử dụng DataAdapter

```
private void Form1_Load(object sender, EventArgs e)
{
    conn = new SqlConnection(chuoiketnoi)
    conn.Open();
    string sql = "Select * from SV";
    SqlDataAdapter daSV = new SqlDataAdapter(sql, conn);
    DataTable dtSV = new DataTable();
    daSV.Fill(dtSV);
    dgvSV.DataSource = dtSV;
}
```

Câu lệnh truy vấn dữ liệu

Đổ dữ liệu truy vấn được vào bảng

Khai báo và khởi tạo một biến kiểu SqlDataAdapter với câu lệnh truy vấn và connection đã tạo ở trên

Hiển thị dữ liệu từ bảng lên giao diện (hiển thị vào DataGridView) thông qua DataTable





Ví dụ sử dụng DataAdapter

- Ví dụ: //lấy dữ liệu vào DataAdapter
SqlDataAdapter daLoaisach = new SqlDataAdapter("SELECT * FROM LoaiSach", conn);
//đổ dữ liệu vừa lấy ra vào DataTable
DataTable dtLoaiSach = new DataTable();
daLoaiSach.Fill(dtLoaiSach);
//hiển thị dữ liệu lên combobox
cbLoaisach.DataSource = dtLoaiSach;
cbLoaisach.DisplayMember = "TenLoai";
cbLoaisach.ValueMember = "MaLoai";





SqlCommand

- Thực hiện một câu lệnh truy vấn trực tiếp tới cơ sở dữ liệu: Select, Insert, Update, Delete hoặc gọi một thủ tục (Store Procedure) từ cơ sở dữ liệu.





Thành phần của SqlCommand

- CommandText: xác định câu lệnh cần thực hiện.
- CommandType: xác định kiểu câu lệnh cần thực hiện.
- Connection: xác định đối tượng kết nối SqlConnection.
- ExecuteNonQuery(): thực hiện các câu lệnh insert, update, delete và trả về số dòng được thực hiện, ngược lại trả về giá trị -1.
- ExecuteReader(): thực hiện câu lệnh select và trả kết quả truy vấn được vào đối tượng SqlDataReader





Ví dụ SqlCommand

- Ví dụ thêm mới dữ liệu vào bảng Sinhviên

```
private void btThem_Click(object sender, EventArgs e)
{
    int gt = 0;
    if(rbNam.Checked == true)
        gt = 1;
    string sql_Insert = "Insert into SV values('" + tbMa.Text + "', N'"
                        + tbHoten.Text + "', '" + dtpNgaysinh.Value.Date
                        + "','" + gt.ToString() + "','" + cbQue.SelectedValue.ToString() + "')";
    SqlCommand cmd = new SqlCommand(sql_Insert, conn);
    //dùng SqlCommand để thực hiện các câu lệnh cập nhật dữ liệu như Insert, Update, Delete
    cmd.ExecuteNonQuery(); //thực hiện câu lệnh trong chuỗi sql_Insert
    //thực hiện cập nhật dữ liệu mới thêm vào datagridview
    dtSV.Rows.Clear(); //xóa dữ liệu trong datagridview
    daSV.Fill(dtSV); //hiển thị lại dữ liệu vào datagridview
}
```





DataTable

- DataTable biểu diễn một bảng trong DataSet.
- DataSet có thể chứa nhiều DataTable.
 - ChildRelations: xác định tập hợp các đối tượng DataRelation trỏ tới con của đối tượng DataTable.
 - Clear(): xoá tất cả dữ liệu trong đối tượng DataTable.
 - ColumnChanged(): sự kiện xảy ra khi dữ liệu tại một cột bị thay đổi.
 - ColumnChanging(): sự kiện xảy ra khi dữ liệu tại một cột đang bị thay đổi.
 - Columns: xác định tập hợp các đối tượng DataColumn





DataTable

- Constraints: xác định tập hợp các đối tượng Constraint
- NewRow(): tạo ra một hàng mới, rỗng trong DataTable
- ParentRelations: xác định tập hợp các đối tượng DataRelation trả tới cha của đối tượng DataTable.
- PrimaryKey: xác định một mảng các đối tượng DataColumn, cung cấp khoá chính cho đối tượng DataTable.
- RowChanged(): sự kiện xảy ra khi dữ liệu trong đối tượng DataRow bị thay đổi.
- RowChanging(): sự kiện xảy ra khi dữ liệu trong đối tượng DataRow đang bị thay đổi.





DataTable

- RowDeleted(): sự kiện xảy ra khi một hàng bị xoá.
- RowDeleting(): sự kiện xảy ra khi một hàng đang bị xoá.
- Rows: xác định tập hợp các đối tượng DataRow.
- Select(): xác định mảng các đối tượng DataRow thỏa mãn tiêu chuẩn lựa chọn.
- TableName: xác định tên của đối tượng DataTable.





Ví dụ

- Sử dụng SqlCommand để thực hiện câu lệnh insert into

```
//tạo một đối tượng SqlCommand
SqlCommand cmd = new SqlCommand();
cmd.Connection = conn; //gán Connection cho SqlCommand
cmd.CommandText = "insert into LoaiSach values(''" +
tbMaloai.Text + "','" + tbTenLoai.Text + "')";
cmd.ExecuteNonQuery(); //thực hiện câu lệnh
```





SqlDataReader

- SqlDataReader là đối tượng được thiết kế nhằm lấy dữ liệu từ CSDL một cách nhanh nhất.
- SqlDataReader được tạo ra bằng cách gọi phương thức ExecuteReader() của SqlCommand.
- Chỉ có thể đọc kết quả chứa trong đối tượng SqlDataReader từ đầu đến cuối và không thể thay đổi nó.
 - Khi gọi phương thức SqlDataReader.Read() thì nó sẽ load hàng dữ liệu kế tiếp. Nếu không có hàng nào được load thì phương thức Read() sẽ trả về false để báo hết dữ liệu.
 - SqlDataReader cung cấp các phương thức để lấy thứ tự của cột đó và trả về dữ liệu của cột; như GetString(), GetValue() ...
- Không thể thực hiện SqlCommand cùng với SqlConnection với SqlDataReader đang được mở.
- Nên sử dụng phương thức Close() của SqlDataReader sau khi đã lấy xong dữ liệu.



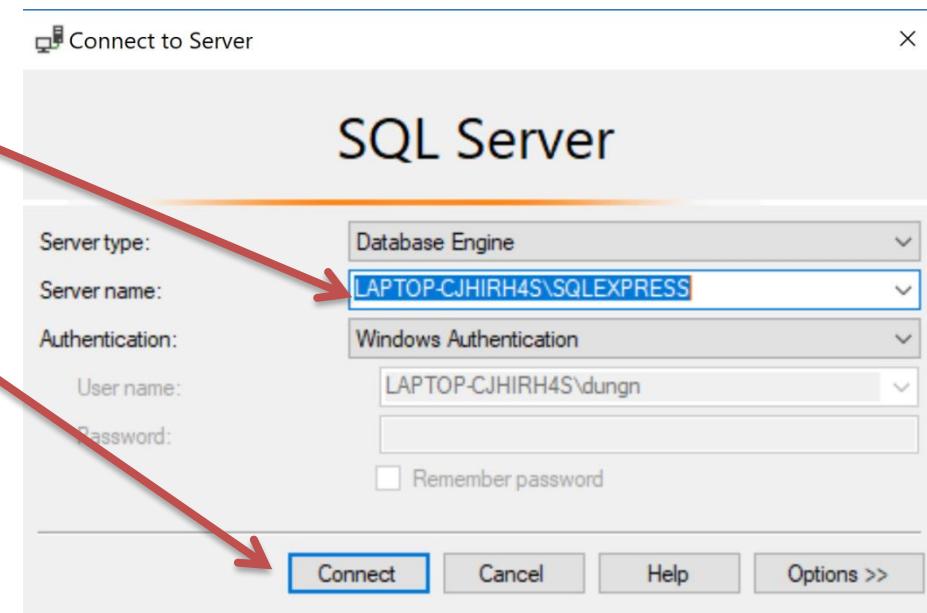


Tạo CSDL bằng công cụ Microsoft SQL Server Management Studio



Khởi động SQL Server Management Studio

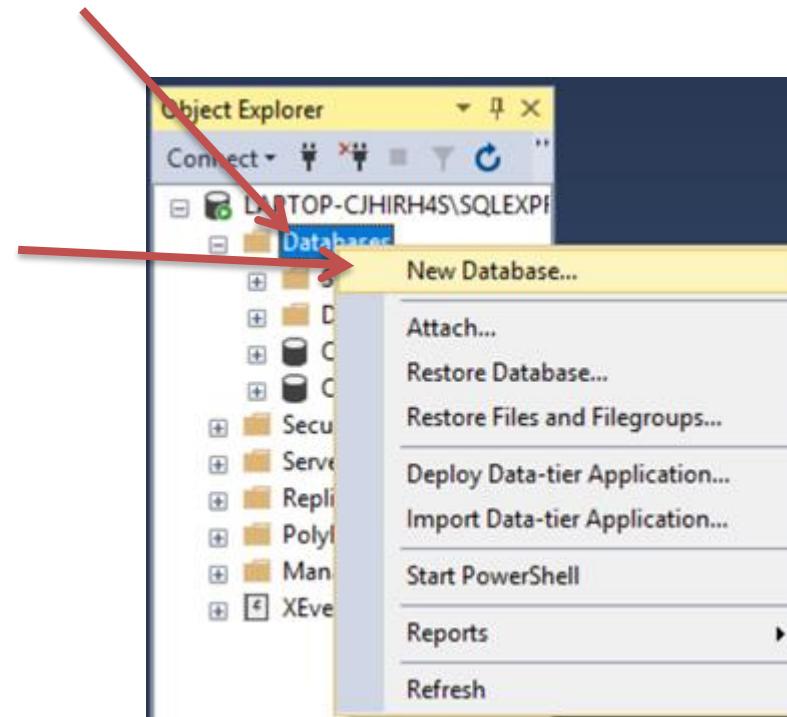
- B1: Chọn Server name rồi bấm vào nút Connect để kết nối đến SQL Server





Tạo mới một cơ sở dữ liệu

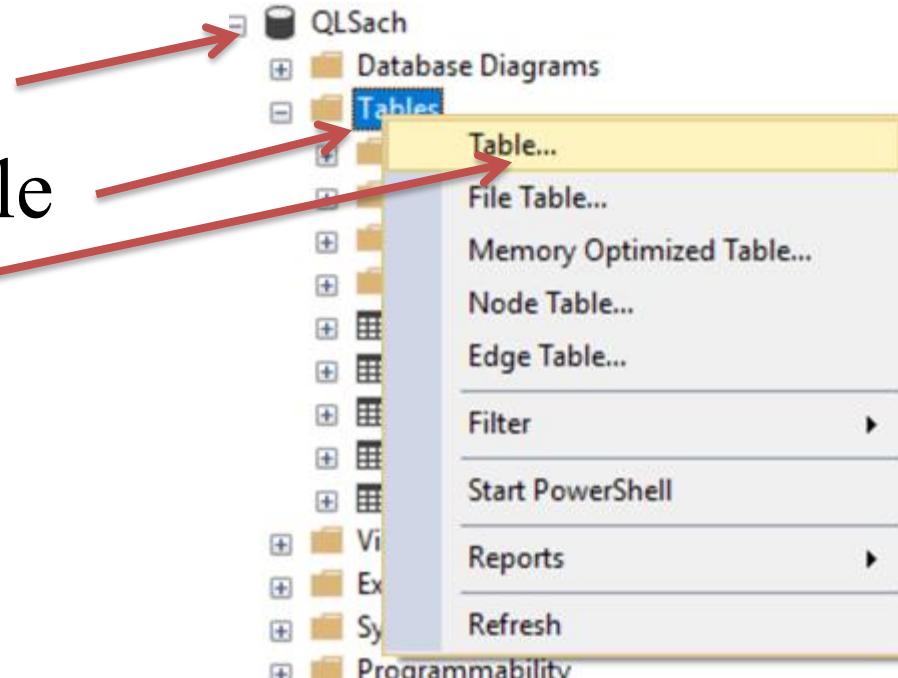
- B2: Bấm chuột phải vào Database rồi chọn New Database để tạo mới một CSDL





Tạo bảng

- Chọn cơ sở dữ liệu
- Chuột phải vào Table
- Chọn Table





Tạo bảng

- Khai báo các trường cho bảng
- Column Name: tên trường
- Data Type: kiểu dữ liệu
- Allow Nulls: cho phép rỗng hoặc không

LAPTOP-CJHIRH4S\SQL...ach - dbo.Table_1*

Column Name	Data Type	Allow Nulls
MaSach	nchar(10)	<input checked="" type="checkbox"/>

nchar(10)
nchar(10)
ntext
numeric(18, 0)
nvarchar(50)
nvarchar(MAX)
real
smalldatetime
smallint





Tạo bảng

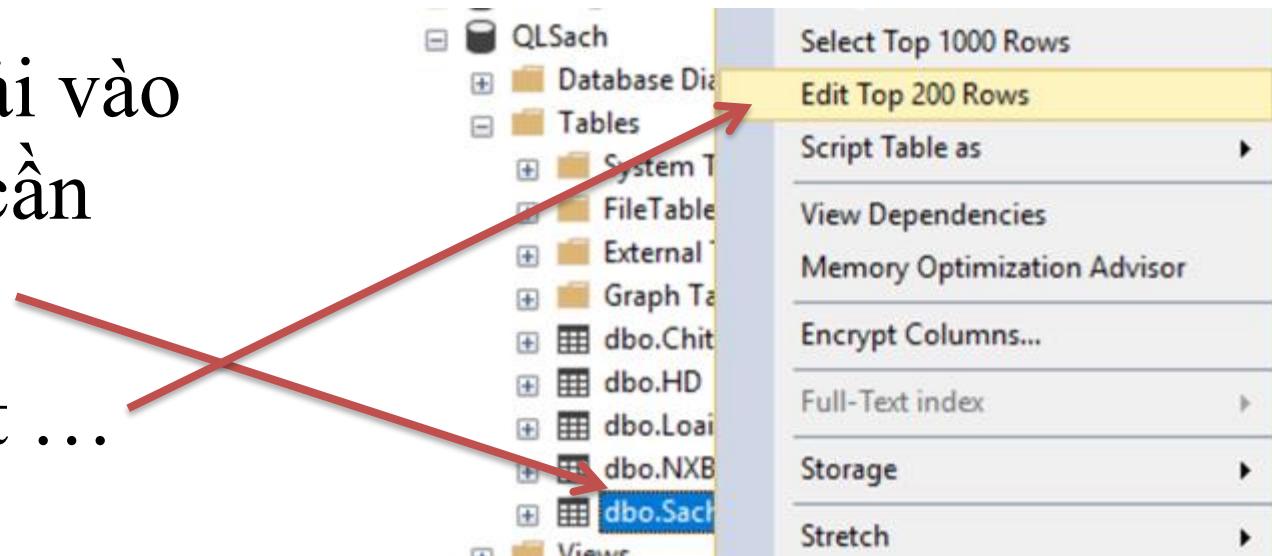
- Chú ý: Khi tạo bảng có quan hệ thì chỉ nên lưu trữ mã nhằm tránh dư thừa dữ liệu
- Ví dụ: trong phần mềm quản lý cửa hàng sách muốn quản lý sách theo thể loại, mỗi thể loại gồm rất nhiều đầu sách khác nhau => để tránh việc lặp đi lặp lại tên thể loại trong bảng sách thì sẽ tách ra làm 2 bảng Loại sách (mã loại, tên loại) và Sách(mã sách, mã loại, tên sách,) trong đó có trường mã loại được lưu trữ ở cả 2 bảng giúp kết nối tương ứng sách nào thuộc loại nào.





Chỉnh sửa dữ liệu trong bảng bằng cách trực tiếp

- Chuột phải vào tên bảng cần chỉnh sửa
- Chọn Edit ...





Chỉnh sửa dữ liệu trong bảng bằng cách trực tiếp

LAPTOP-CJHIRH4S\...QLSach - dbo.Sach ➔ X LAPTOP-CJHIRH4S\...ach - dbo.Table_1*						
	MaSach	MaLoai	TenSach	SoLuong	MaNXB	Dongia
	SV01	M02	Văn học lớp 8	5	NXBGD	15000
▶*	NULL	NULL	NULL	NULL	NULL	NULL

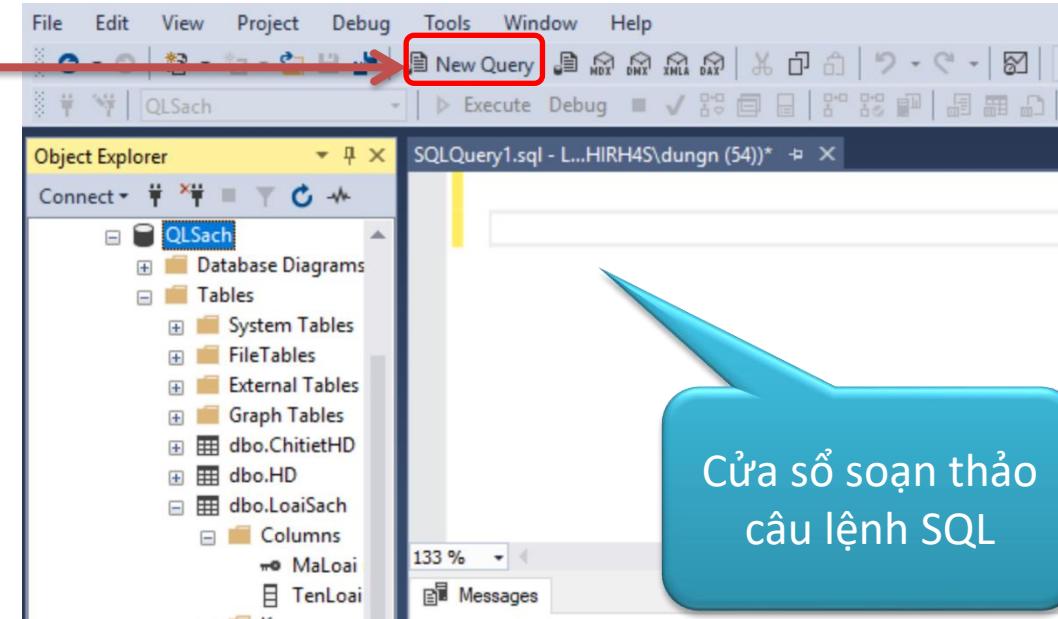
- Thực hiện thêm sửa xóa trực tiếp vào bảng





Sử dụng câu lệnh SQL để làm việc với cơ sở dữ liệu

- Chọn New Query để mở ra cửa sổ soạn thảo câu lệnh SQL





Một số câu lệnh SQL thường dùng

- INSERT INTO...
- UPDATE
- DELETE
- SELECT





INSERT INTO...

- Dùng để thêm dữ liệu vào bảng
- Cú pháp: Insert into <tên bảng> values(<các giá trị tương ứng với các trường>)





INSERT INTO...

- Ví dụ: Bảng Loại sách có trường mã loại sách và tên loại sách kiểu nchar và nvarchar

Column Name	Data Type	Allow Nulls
MaLoai	nchar(10)	<input type="checkbox"/>
TenLoai	nvarchar(50)	<input checked="" type="checkbox"/>

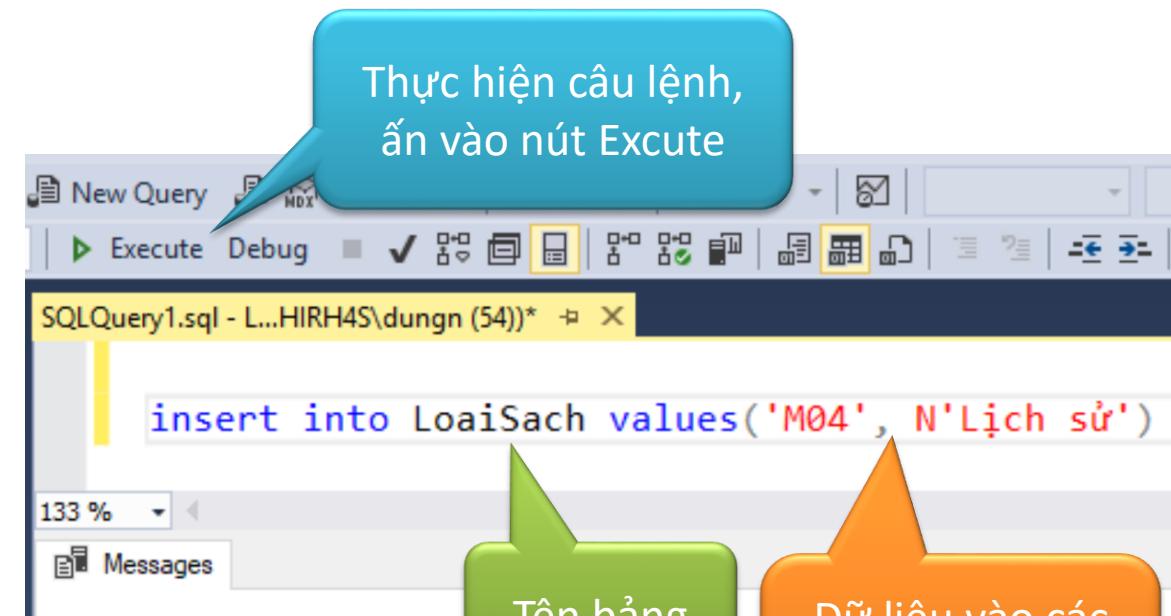




INSERT INTO...

=> Để thêm dữ liệu cho bảng Loại sách sử dụng câu lệnh như hình bên

Thông báo thực hiện thành công hay không



Thực hiện câu lệnh, ấn vào nút Execute

SQLQuery1.sql - L...HIRH4S\dungn (54)*

```
insert into LoaiSach values('M04', N'Lịch sử')
```

133 %

Messages

(1 row affected)

Tên bảng

Dữ liệu vào các trường tương ứng trong bảng



UPDATE

- Dùng để sửa dữ liệu trong bảng
- Cú pháp: Update <tên bảng> set <tên trường> = <giá trị mới> where <điều kiện>
- Ví dụ: sửa tên loại trong bảng loại sách tại dòng có mã M04 thành Sinh học

```
Update LoaiSach set TenLoai = N'Sinh học' where MaLoai = 'M04'
```





DELETE

- Dùng để xóa dòng dữ liệu khỏi bảng
- Cú pháp: Delete <tên bảng> where <điều kiện>
- Ví dụ: xóa dòng có mã là M04 trong bảng Loại sách:

```
Delete LoaiSach where MaLoai = 'M04'
```





SELECT

- Dùng để truy vấn dữ liệu từ cơ sở dữ liệu
- Cú pháp:

```
SELECT <danh sách các trường>
      FROM <danh sách các bảng>
        WHERE <điều kiện>
```
- Nếu lấy tất cả các trường trong bảng thì thay danh sách các trường bằng dấu *





SELECT

- Ví dụ:
- Lấy tất cả dữ liệu trong bảng Loại sách thì dùng câu lệnh: `select * from LoaiSach`
- Kết quả:

MaLoai	TenLoai
M01	Toán học
M02	Văn học
M03	Anh Ngữ





SELECT

- Ví dụ:
- Muốn tìm các sách thuộc loại toán học thì dùng câu lệnh:

```
select * from Sach where MaLoai = 'M01'
```





SELECT

- Ví dụ truy vấn dữ liệu từ nhiều bảng:
- Muốn đưa ra chi tiết một đơn hàng (tên sách, số lượng, giá bán, thành tiền) trong khi bảng chi tiết đơn hàng chỉ chứa mã sách => dùng lệnh:
`select S.TenSach, CT.SoLuong, CT.Giaban,
CT.SoLuong*CT.Giaban as Thanhtien
from Sach as S, ChitietHD as CT
where S.MaSach = CT.MaSach`





SELECT

- Trong đó:
 - Từ khóa **as** dùng để thay thế tên trường hoặc tên bảng bằng một tên mới
 - Có thể sử dụng các toán tử +, - , * , / trong câu lệnh SELECT





SELECT

- Các hàm có thể dùng trong câu lệnh SELECT:
- Sum: tính tổng giá trị các bản ghi
- Max: tìm giá trị lớn nhất trong các bản ghi
- Min: tìm giá trị nhỏ nhất trong các bản ghi
- Count: đếm số lượng các bản ghi
- Avg: tính trung bình cộng giá trị các bản ghi





SELECT

- Ví dụ:
- Tính tổng tiền phải trả của đơn hàng có mã HD01

```
select sum(soluong*giaban) as 'thành tiền'  
from ChitietHD  
where MaHD = 'HD01'
```



LẬP TRÌNH WINDOWS

Lưu trữ dữ liệu bằng file JSON



Giảng viên: Nguyễn Thị Phương Dung
Email: dungntp@tlu.edu.vn

Quản lý

Quản lý

Quản lý

Quản lý sinh viên

Tỉnh

Khoa

Tỉnh

Khoa

Tỉnh

Khoa

Mã tỉnh

Mã khoa

Tên tỉnh

Tên khoa

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Khoa

Mã lớp

Tên lớp

Mã SV

Họ tên

Giới tính

Ngày sinh

1/ 1/2000

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

SV

Họ

tên

Ngày

sinh

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Nam

Nữ

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

Mã

1

CN

2

CT

3

KT

4

QT

Mã

M

Chuẩn định dạng dữ liệu JSON

- JSON viết tắt của: **JavaScript Object Notation**
- JSON sử dụng định dạng key-value cho mỗi thông tin dữ liệu.
 - Key: ≈ tên trường hoặc tên thuộc tính
 - Value: ≈ dữ liệu của trường / giá trị của thuộc tính
- Lưu trữ dưới dạng file text thường có phần mở rộng là .json hoặc .js



Ví dụ

	Mã khoa	Tên khoa
►	CNTT	Công nghệ thông tin
█	CT	Công trình
{	KT	Tài chính kế toán
{	QTKD	Quản trị kinh doanh

Định dạng chuỗi JSON

- Mỗi đối tượng JSON được bao bọc bởi cặp dấu ngoặc nhọn {}
- Dấu : dùng để phân cách giữa key và value
- Dấu , để phân cách giữa các cặp key-value, hoặc giữa các đối tượng JSON trong mảng
- Các key và value được đặt trong cặp dấu nháy kép “” trừ dữ liệu dạng số và dạng Boolean.
- Nếu trong value có chứa dấu “ thì dùng dấu (\) trước dấu “ đó.
- Nếu value là một mảng các đối tượng JSON khác thì được bao bởi cặp ngoặc []



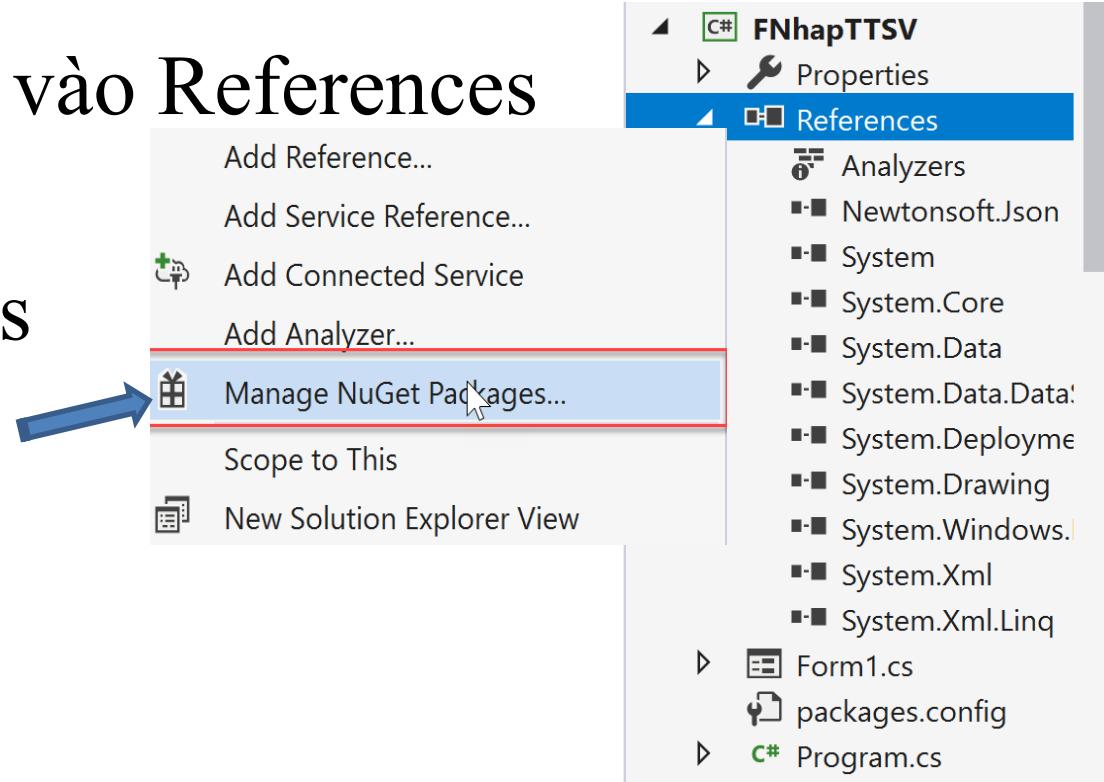
Kiểu dữ liệu trong JSON

- string: phải đặt trong cặp dấu ngoặc kép “”
- number: là một số nguyên hoặc là một số thực
- object: một đối tượng thuộc kiểu JSON
- array: một mảng được bao trong cặp dấu ngoặc vuông []
- Boolean
- NULL



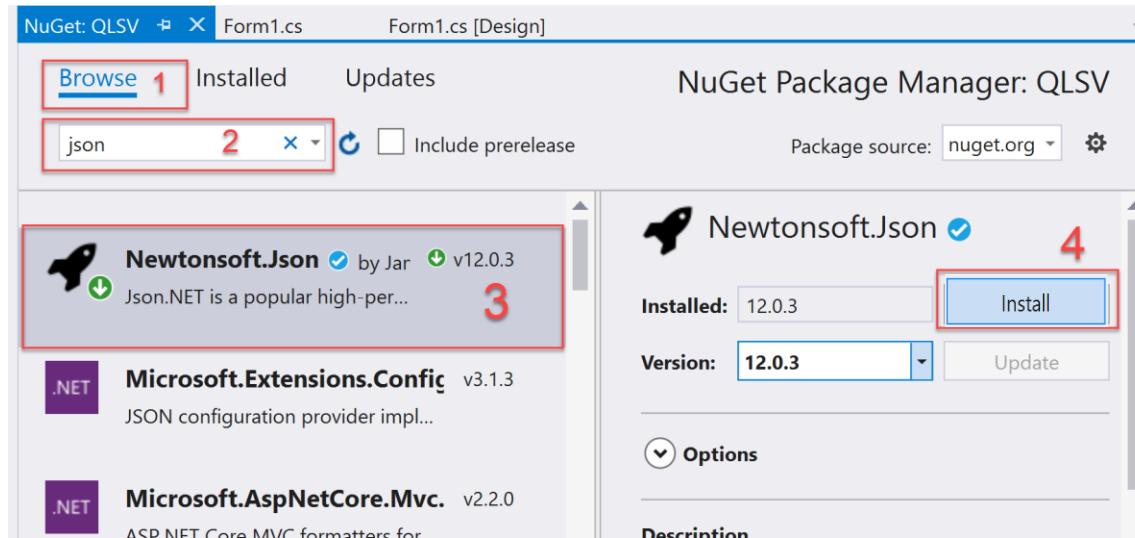
Cài đặt JSON vào C#

- Bấm chuột phải vào References
- Chọn Manage NuGet Packages

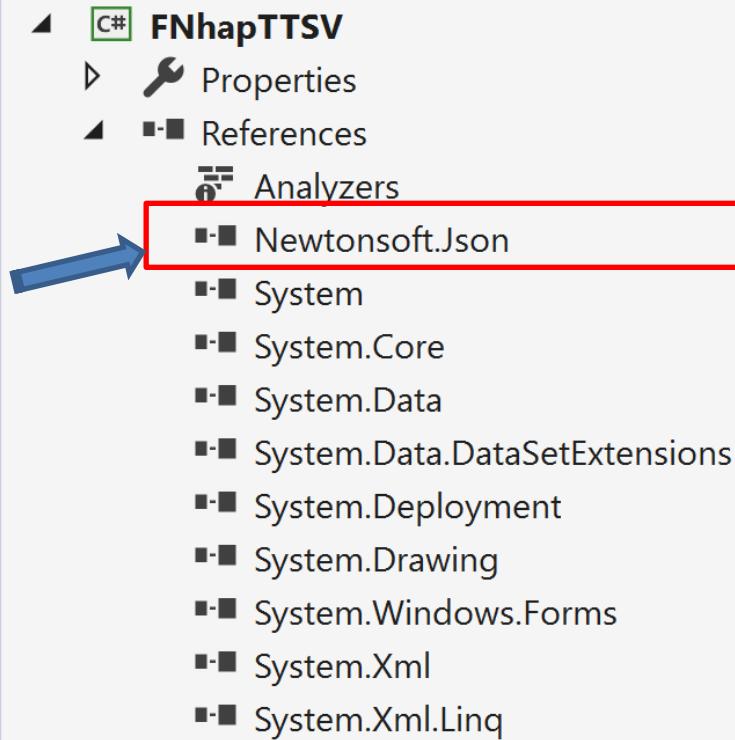


Cài đặt JSON vào C#

- Thực hiện tìm kiếm và cài đặt theo các bước sau:



Cài đặt JSON vào C#



Sử dụng JSON trong C#

- Khai báo: `using Newtonsoft.Json;`
- Sử dụng các lệnh chuyển đổi:
- `JsonConvert.SerializeObject(object)`: chuyển đổi object sang định dạng chuỗi JSON
- `JsonConvert.DeserializeObject<object>(json)`: chuyển đổi chuỗi JSON về dạng đối tượng



Cách đọc/ghi file JSON

- Sử dụng lệnh

`System.IO.File.WriteAllText(filepath, string)`

để ghi string vào file

- Ví dụ:

`System.IO.File.WriteAllText(@"data.json", jsonstr);`



Cách đọc/ghi file JSON

- Tạo đối tượng thuộc lớp **System.IO.StreamReader** để đọc file
- Ví dụ:

```
System.IO.StreamReader reader;  
reader = new System.IO.StreamReader(@"data.json");  
string jsonstr = reader.ReadToEnd();  
reader.Close(); //đóng file sau khi đọc
```



Ví dụ

- Viết chương trình cho phép:
 - Nhập danh sách các tỉnh thành
 - Lưu dữ liệu để lần sau có thể mở ra chỉnh sửa hoặc thêm tiếp
- => Cần xác định nội dung cần lưu trữ và xác định định dạng dữ liệu cần lưu trữ



Cách làm 1

- Sử dụng các đối tượng có sẵn như:
 - DataTable: cấu trúc dạng bảng, danh sách các dòng
 - DataSet: cấu trúc dạng danh sách các bảng



Cách làm 1

- Tạo bảng Tỉnh

```
DataTable dtTinh = new DataTable("Tỉnh");
```

- Thiết lập các cột cho bảng và gán DataSource của DataGridView bằng bảng vừa tạo

```
dtTinh.Columns.Add("ID");
```

```
dtTinh.Columns.Add("Tên tỉnh");
```

```
dgvTinh.DataSource = dtTinh;
```



Cách làm 1

- Thêm mới dữ liệu vào bảng thì datagridview cũng cập nhật theo:

```
dtTinh.Rows.Add(idtinh, textBox1.Text);
```



Cách làm 1

- Chuyển đổi bảng thành chuỗi JSON

```
string jsonstr;  
//chuyển đổi bảng thành chuỗi JSON  
jsonstr = JsonConvert.SerializeObject(dtTinh);
```



Cách làm 1

- Chuyển đổi chuỗi JSON thành bảng

```
//Chuyển đổi chuỗi json thành bảng
```

```
dtTinh = JsonConvert.DeserializeObject<DataTable>(jsonstr);
```

```
//Gán dữ liệu của bảng vào datagridview
```

```
dgvTinh.DataSource = dtTinh;
```



Cách làm 2

- Tạo lớp tĩnh để xác định thuộc tính và giá trị dữ liệu sẽ lưu trữ.

```
class Tinh
{
    int id;
    string name;
    public Tinh(int i, string n)
    {
        id = i; name = n;
    }
}
```



Cách làm 2

- Tạo 1 danh sách các đối tượng kiểu Tinh để có thể lưu trữ dưới dạng JSON

```
List<Tinh> dstinh= new List<Tinh>();
```



Cách làm 2

- Mỗi lần thêm mới một đối tượng Tinh thì cập nhật vào danh sách.

```
Tinh t = new Tinh(id, textBox1.Text);  
dstinh.Add(t);
```



Cách làm 2

- Sau đó chuyển danh sách thành dạng chuỗi JSON rồi lưu vào file.

```
string jsonstr = JsonConvert.SerializeObject(dstinh);  
System.IO.File.WriteAllText("tinh.json", jsonstr);
```



Cách làm 2

- Khi nào cần dùng thì đọc file ra để lấy dữ liệu:

```
//tạo đối tượng StreamReader để truy cập vào file
System.IO.StreamReader reader = new System.IO.StreamReader(@"tinh.json");
//sử dụng lệnh ReadToEnd() để lấy toàn bộ dữ liệu trong file
string jsonstr = reader.ReadToEnd();
//tạo đối tượng phù hợp với chuỗi JSON đã lưu
//và chuyển dữ liệu từ chuỗi json đã lưu vào đối tượng
List<Tinh> dstinhhs = JsonConvert.DeserializeObject<List<Tinh>>(jsonstr);
//sử dụng danh sách chuyển đổi từ chuỗi JSON
dstinhhs.ForEach(tinh => dataGridView.Rows.Add(tinh.ID, tinh.Name));
```



Bài tập

Quản lý sinh viên

Tỉnh Khoa Lớp Sinh viên

Mã SV	Nơi sinh	Hà Nội				
Họ tên	Khoa					
Giới tính	Nam	Nữ				
Ngày sinh	60TH1	Thêm Sửa Xóa				
1/ 1/2000						
Mã SV	Họ tên	Ngày sinh	Giới tính	Quê quán	Khoa	Lớp

- Tạo form cho phép thêm, sửa, xóa các loại dữ liệu Tỉnh, Khoa, Lớp và thông tin Sinh viên.
- Trong đó dữ liệu của Tỉnh, Khoa, Lớp trong giao diện thông tin sinh viên được cập nhật ngay khi cập nhật các thông tin đó ở các giao diện tương ứng.