

Chương 1. Cơ Bản về Ngôn ngữ C#

Mục tiêu

#2

- Ôn lại các khái niệm, các kiến thức lập trình cơ bản
- Cách khai báo biến toàn cục
- Cách khai báo biến cục bộ
- Cách sử dụng hàm và biến toàn cục
- Tham số và hàm
- Tái định nghĩa hàm, cách truyền tham chiếu đối với các hàm

Nội dung

#3

- Ôn tập các kiến thức lập trình cơ bản
- 1.1 Nhập xuất thông qua stream
- 1.2 Tham số mặc nhiên
- 1.3 Tái định nghĩa hàm
- 1.4 Hàm inline
- 1.5 Tham chiếu, truyền tham số bằng tham chiếu.

Ôn tập các kiến thức lập trình cơ bản

#4

- Máy tính dùng để giải quyết một loạt các bài toán.
- Mỗi bài toán có cách giải quyết khác nhau dựa vào thuật giải.
- Lập trình viên thể hiện các thuật giải theo một ngôn ngữ lập trình cụ thể.

Lập trình là gì?

#5

Máy tính chỉ hiểu được ngôn ngữ máy, do đó cần phải có giai đoạn chuyển ngôn ngữ lập trình sang ngôn ngữ máy thông qua trình biên dịch của ngôn ngữ lập trình.

Đặc điểm của ngôn ngữ C#

#6

- Ngôn ngữ lập trình được xây dựng dựa trên nền tảng những ngôn ngữ tương tự C (C, C++, Java) nhưng hoạt động trên .Net Framework.
- Hoạt động trên .NET Framework.

Đặc điểm của ngôn ngữ C#

#7

- Dựa trên phương pháp thiết kế hướng đối tượng.
- Ứng dụng : Console, Winform, Webform.
- Có tính diễn đạt ngữ nghĩa cao.
- Phân biệt chữ hoa thường.

.NET Framework

#8

- Framework là một tập hợp các thư viện để hỗ trợ cho người lập trình.
- Mỗi Framework được tạo ra có một kiến trúc khác nhau $\Rightarrow$ LTV phải tuân theo kiến trúc đó
- .NET Framework là thư viện tài nguyên của Microsoft, hỗ trợ cho các lập trình viên trong nhiều yêu cầu khác nhau.

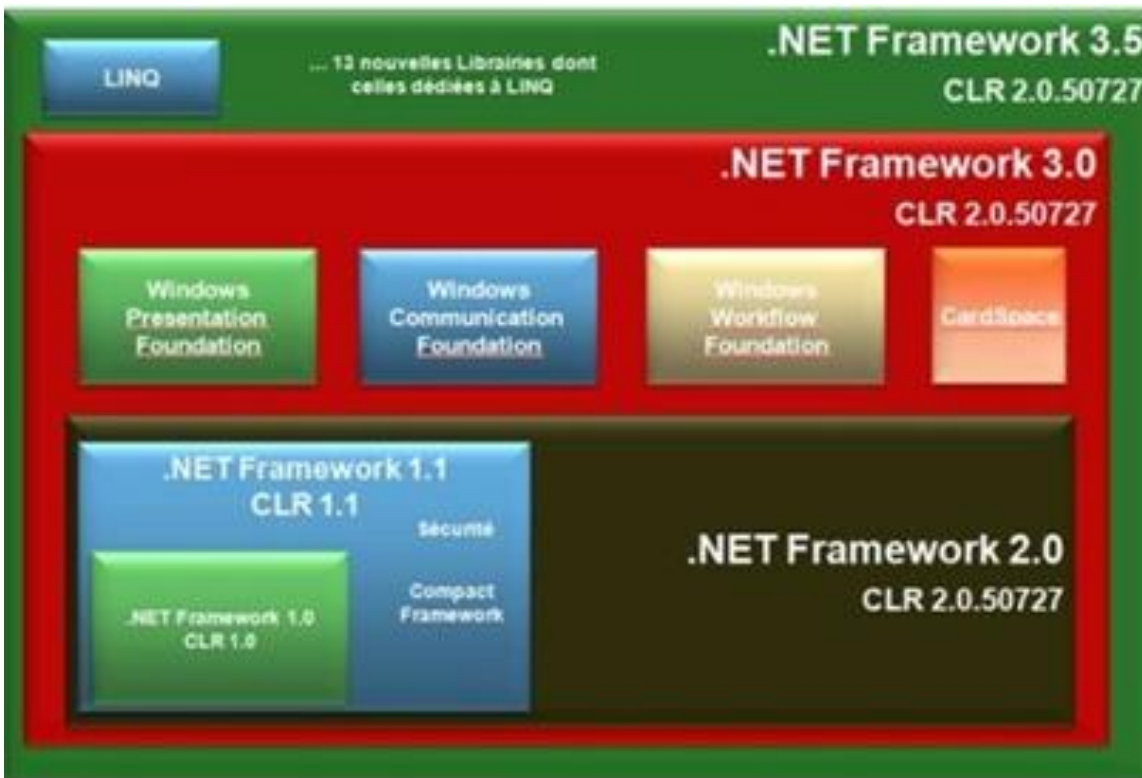
.NET Framework

#9

Version	Version Number	Release Date	Visual Studio	Default in Windows
1.0	1.0.3705.0	2002-02-13	Visual Studio .NET	
1.1	1.1.4322.573	2003-04-24	Visual Studio .NET 2003	Windows Server 2003
2.0	2.0.50727.42	2005-11-07	Visual Studio 2005	Windows Server 2003 R2
3.0	3.0.4506.30	2006-11-06		Windows Vista, Windows Server 2008
3.5	3.5.21022.8	2007-11-19	Visual Studio 2008	Windows 7, Windows Server 2008 R2
4.0	4.0.30319.1	2010-04-12	Visual Studio 2010	

.NET Framework

#10



The .NET Framework Stack

.NET Framework

#11

- Các ngôn ngữ : C#, VB.Net, J#, F#, VC++...
- Công cụ phát triển Visual Studio
- Lớp đặc tả ngôn ngữ dùng chung (CLS)
- Các thư viện để phát triển ứng dụng
- Bộ thực thi ngôn ngữ dùng chung (CLR)

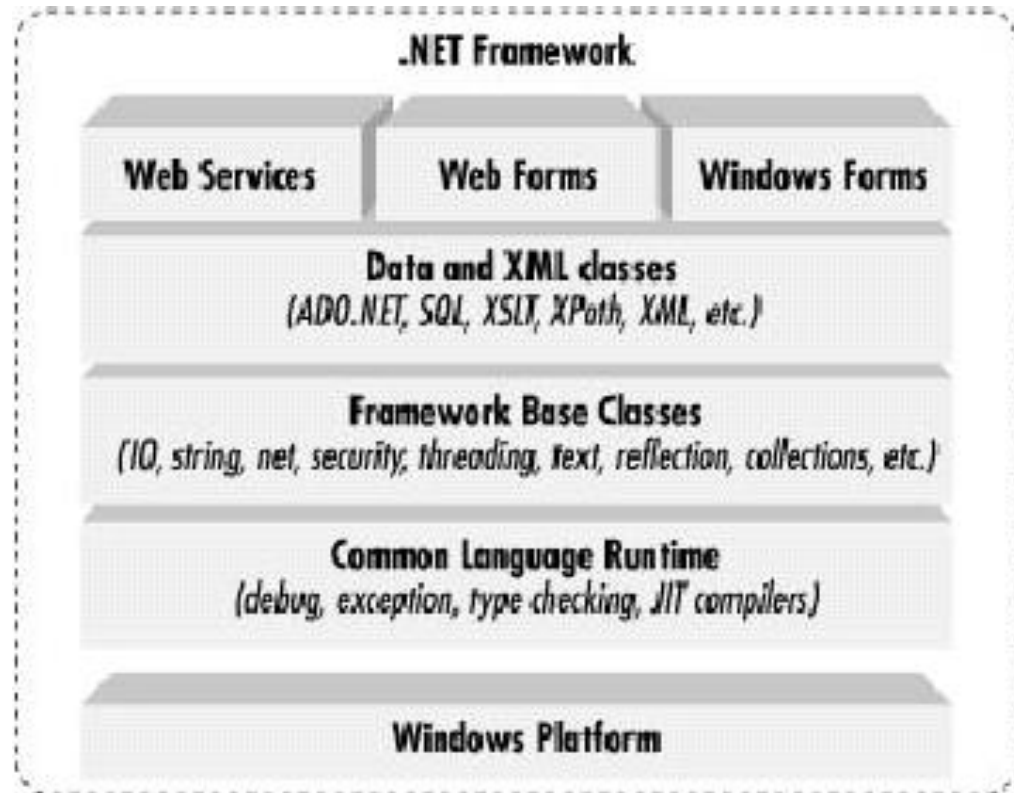
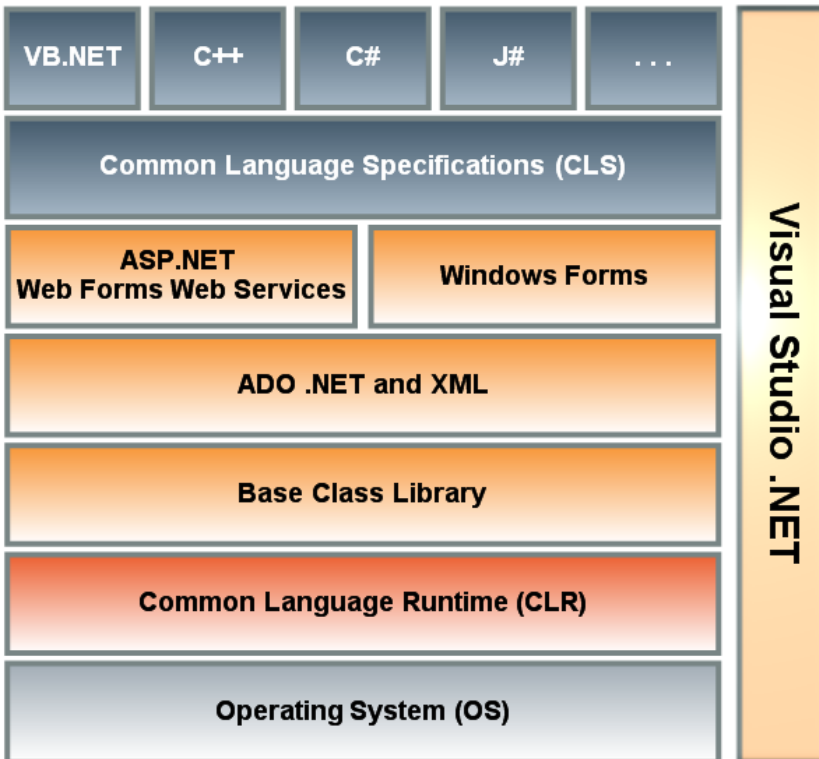
.NET Framework

#12

- Chương trình được biên dịch thành ngôn ngữ trung gian (MSIL - Microsoft Intermediate Language), sau đó chúng được CLR thực thi.
- Common Language Runtime - CLR, nền tảng hướng đối tượng cho phát triển ứng dụng Windows và Web mà các ngôn ngữ có thể chia sẻ sử dụng.
- Bộ thư viện Framework Class Library - FCL.

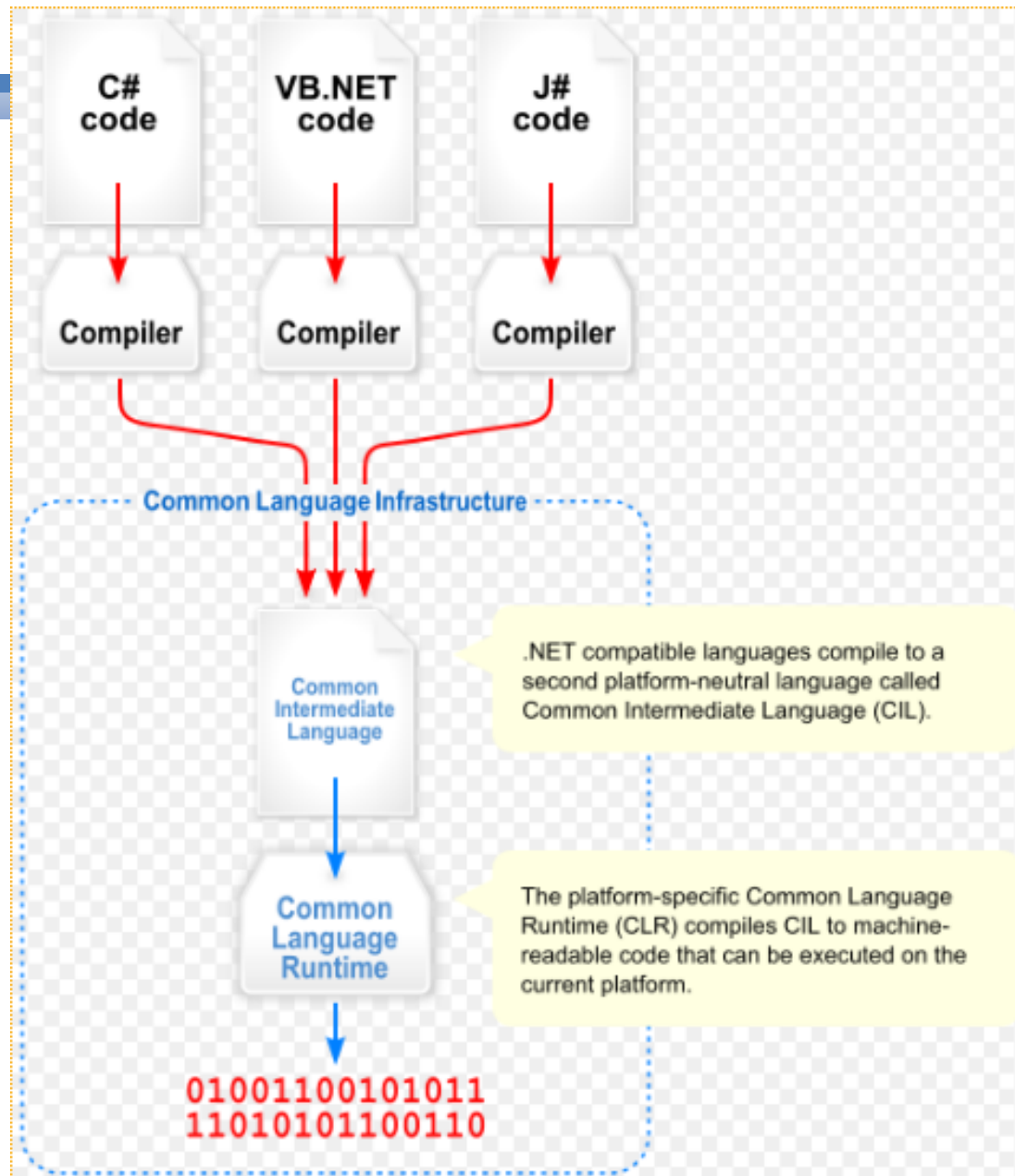
.NET Framework

#13



.NET Framework

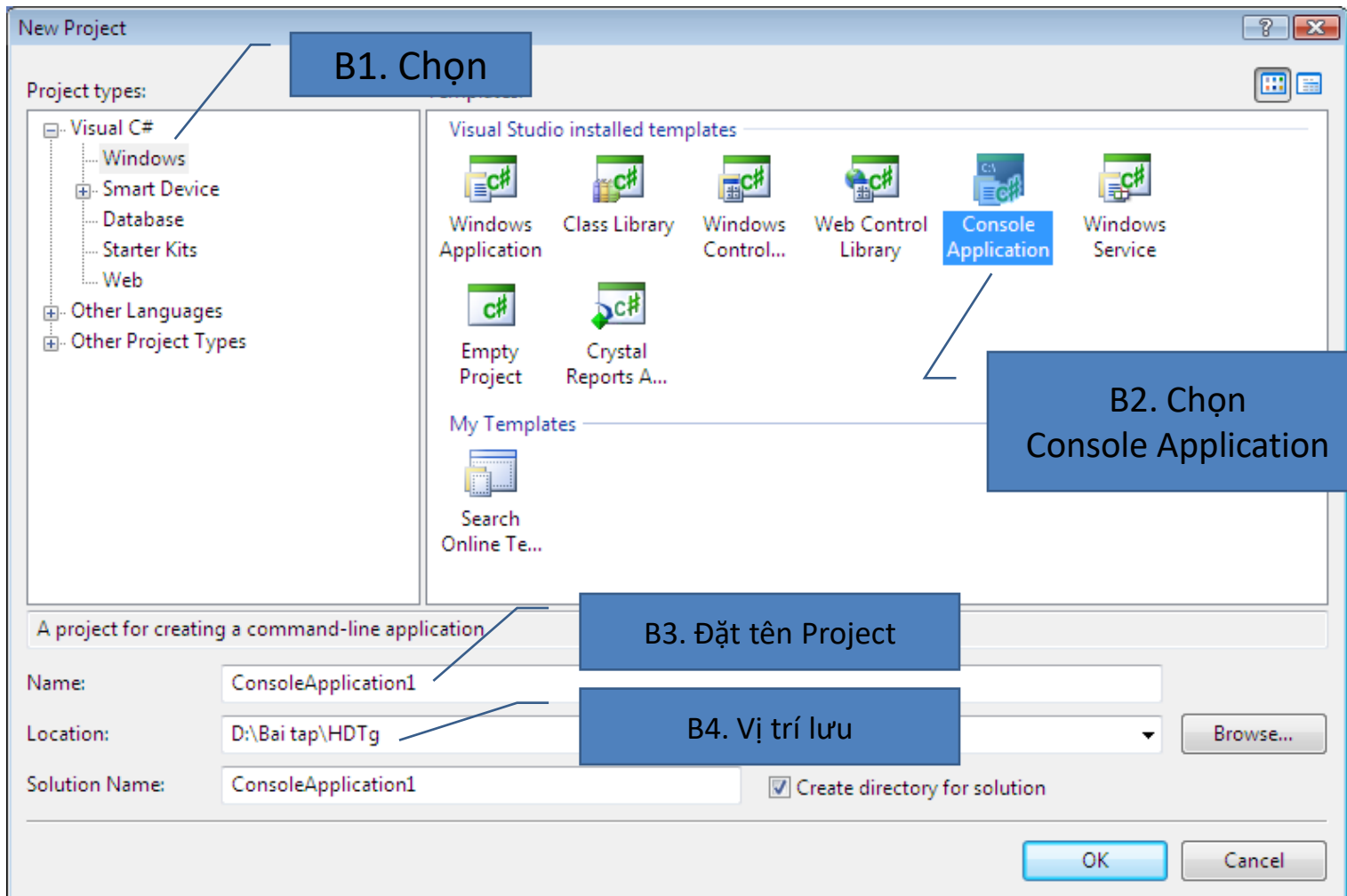
#14



Khởi Tạo Project

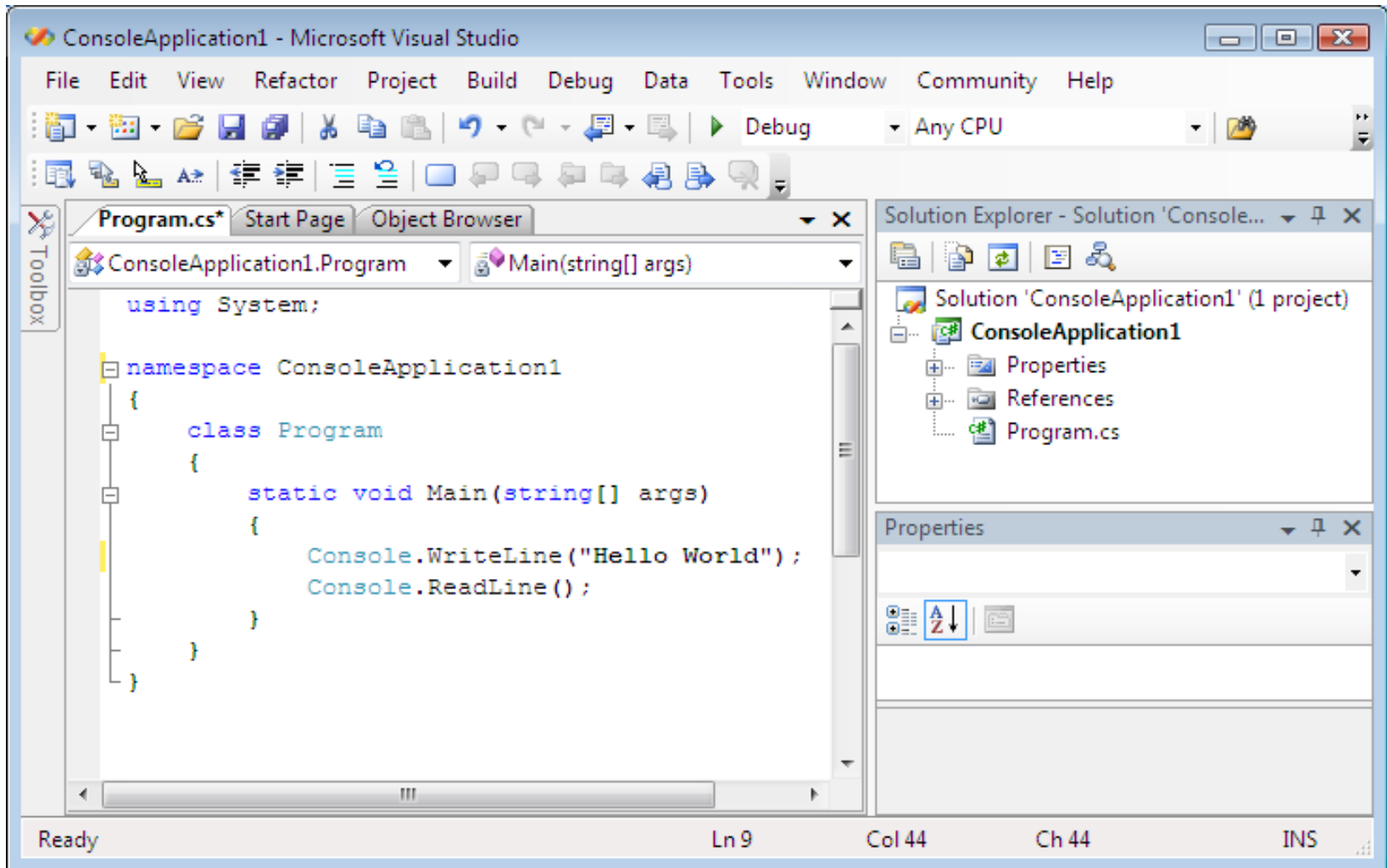
#15

Tạo project dạng Console: File ⇒ New ⇒ Project



Khởi Tạo Project

#16



File **Program.cs** là file mặc định chứa hàm Main của chương trình

Khởi Tạo Project

#17

- Cấu trúc một project :

```
using System; //khai báo thư viện sử dụng
namespace ConsoleApplication1
{
    class Program //tên lớp, tên file = tên lớp
    {
        static void Main(string[] args)
        {
            //Chương trình chính viết tại đây
        }
    }
}
```

Compile & chạy chương trình

#18

- Trình biên dịch (compiler) sẽ biên dịch các tập tin chứa ngôn ngữ C# thường là các file .cs trong project thành một tập tin chạy chương trình .exe
- Có 2 cách biên dịch :
 - Tại cửa sổ cmd, gõ : `csc.exe tenfile.cs`
 - Nhấn Build / Compile (hoặc Build / Build Solution) $\Rightarrow$ Biên dịch cả project.

Compile & chạy chương trình

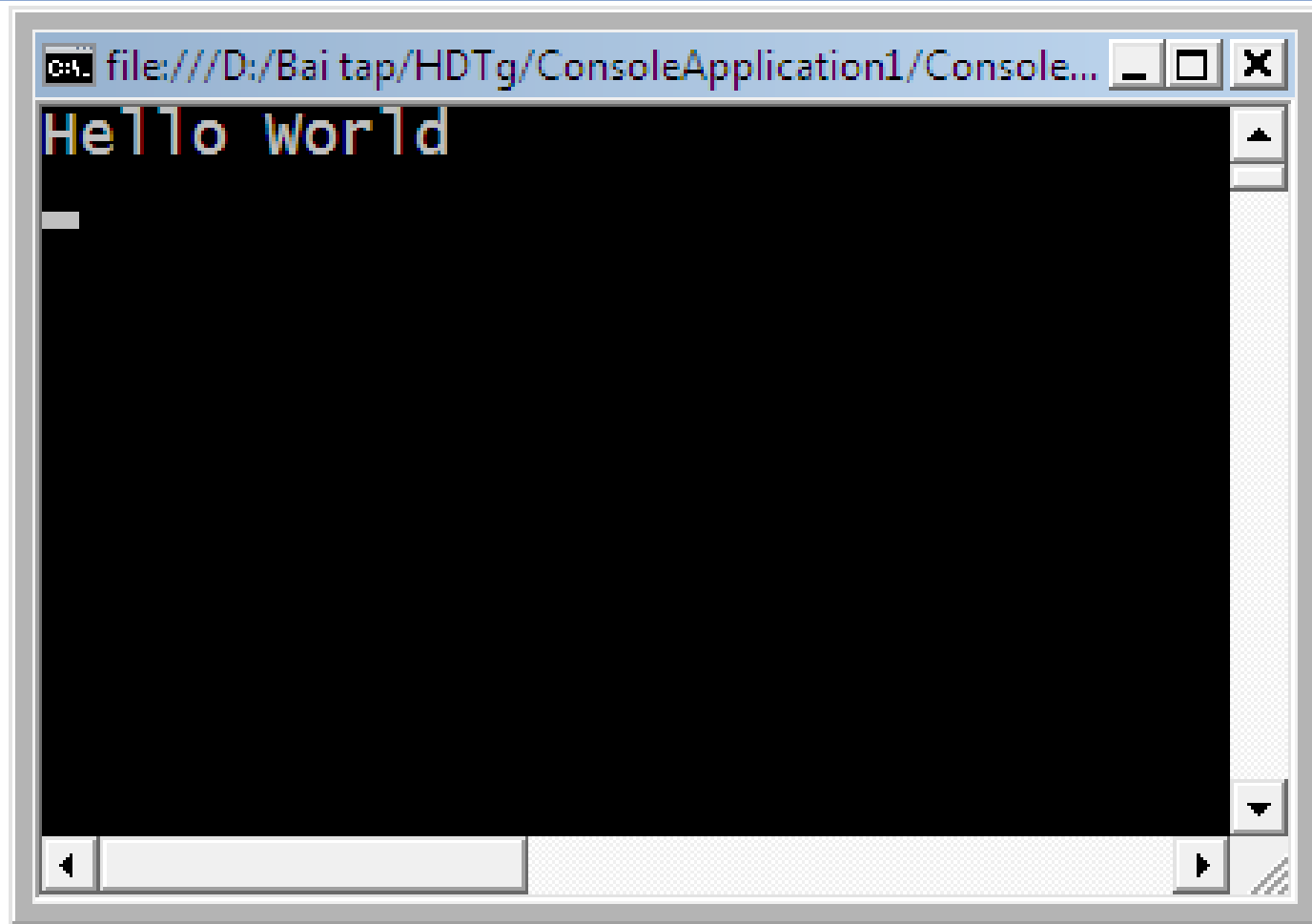
#19

Chạy chương trình

- Sử dụng file `tenfile.exe` trong thư mục `Bin\Debug`
- Hoặc click `Debug\ Start (Ctrl + F5)`

Kết quả

#20



A screenshot of a Windows console window. The title bar at the top reads "file:///D:/Bai tap/HDTg/ConsoleApplication1/Console...". The main area of the window is black and displays the text "Hello World" in a multi-colored font. A white cursor is visible on the line below the text. The window has standard Windows controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

```
file:///D:/Bai tap/HDTg/ConsoleApplication1/Console...  
Hello World  
_
```

Từ khoá – Keywords

#21

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

Namespace (không gian tên)

#22

Namespace là một khái niệm được sử dụng để phân nhóm các lớp đối tượng trong .Net Framework, tránh việc trùng tên giữa các lớp đối tượng

Namespace (không gian tên)

#23

Ví dụ:

`System.Drawing2D.Pen`

và `System.Drawing3D.Pen`

đều đề cập đến một lớp đối tượng Pen nhưng thuộc hai namespace khác nhau, do đó chúng là hai lớp đối tượng khác nhau.

Sử dụng Namespace

#24

Khai báo trực tiếp bằng cách ghi đầy đủ namespace.

VD:

```
System.Media.SoundPlayer spStart
```

```
=new
```

```
System.Media.SoundPlayer(“start.wav”);
```


Sử dụng Namespace

#25

Sử dụng từ khóa `using` để khai báo trước namespace sẽ được tham chiếu đến.

VD:

```
using System.Media;
```

```
SoundPlayer spStart = new SoundPlayer();
```

Lệnh & Khối lệnh

#27

- Một câu lệnh thực hiện một chức năng nào đó (gán, xuất, nhập, ...) và được kết thúc bằng dấu chấm phẩy (;)
- Khối lệnh gồm nhiều lệnh và được đặt trong cặp dấu ngoặc nhọn { }

Data Types (Kiểu dữ liệu - KDL)

#28

- KDL là các loại dữ liệu và phạm vi giá trị của chúng trong bộ nhớ mà người lập trình sử dụng để lưu trữ.
- Có 2 loại : KDL dựng sẵn & KDL tự định nghĩa.

Data Types (Kiểu dữ liệu - KDL)

#29

C# cũng chia tập dữ liệu thành hai kiểu: giá trị và tham chiếu. Biến kiểu giá trị được lưu trong vùng nhớ stack, còn biến kiểu tham chiếu được lưu trong vùng nhớ heap.

KDL định sẵn

#30

- Số nguyên
- Số thực
- Ký tự, logic
- Chuỗi ký tự

Số nguyên

#31

Kiểu C#	Kích Thước (byte)	Kiểu .NET	Miền giá trị	Mô tả
byte	1	Byte	[0..255]	Số nguyên dương không dấu
sbyte	1	Sbyte	[-128..127]	Số nguyên có dấu
short	2	Int16	[0..65.535]	Số nguyên không dấu
int	4	Int32	Từ -2.147.483.647 đến 2.147.483.646	Số nguyên có dấu
uint	4	UInt32	Từ 0 đến 4.294.967.295	Số nguyên không dấu
long	8	Int64	Từ -9.223.370.036.854.775.808 đến 9.223.370.036.854.775.807	Số nguyên có dấu
ulong	8	UInt64	Từ 0 đến 0xffff ffff ffff ffff	Số nguyên không dấu

Ký tự và logic

#32

Kiểu C#	Kích thước (byte)	Kiểu .NET	Miền giá trị	Mô tả
bool	1	Boolean	true hoặc false	Giá trị logic
char	2	Char		Ký tự Unicode

Số thực

#33

Kiểu C#	Kích thước (byte)	Kiểu .NET	Miền giá trị	Mô tả
float	4	Single	Từ $3,4E-38$ đến $3,4E+38$	Kiểu dấu chấm động, với 7 chữ số có nghĩa
double	8	Double	Từ $1,7E308$ đến $1,7E+308$	Kiểu dấu chấm động có độ chính xác gấp đôi, với 15 chữ số có nghĩa
decimal	8	Decimal		Có độ chính xác đến 28 con số, phải có hậu tố “m” hoặc “M” theo sau giá trị

String (kiểu chuỗi)

#34

- Kiểu **string** có thể chứa nội dung không giới hạn, vì đây là kiểu dữ liệu đối tượng được chứa ở bộ nhớ heap.

- Khai báo:

```
string s = “Nguyen Van A”;
```

Kiểu mảng

#35

- Mảng là một tập hợp các phần tử cùng một kiểu dữ liệu liên tiếp nhau và được truy xuất thông qua chỉ số.
- Chỉ số bắt đầu từ 0.

Kiểu mảng

#36

- Mảng một chiều

`<KDL> []<Tên mảng>=new <KDL>[Số phần tử];`

- **VD:** Khai báo mảng số nguyên **arr** gồm 5 phần tử

```
int [] arr = new int [5];
```

Kiểu mảng

#37

- Mảng hai chiều

<KDL> [,]<Tên mảng>=new <KDL>[Số dòng, số cột];

- **VD:** Khai báo ma trận số nguyên **mt** gồm 5 dòng và 3 cột
long [,] mt = new long [5, 3];

Enum (kiểu liệt kê)

#38

Enum là một cách thức để đặt tên cho các trị nguyên (các trị kiểu số nguyên, theo nghĩa nào đó tương tự như tập các hằng), làm cho chương trình rõ ràng, dễ hiểu hơn

Enum (kiểu liệt kê)

#39

■ VD1:

```
enum Ngay {Hai, Ba, Tu, Nam, Sau, Bay,  
ChuNhat};
```

Hai = 0; Ba = 1; ... ; ChuNhat = 6

■ VD2:

```
enum Ngay {Hai = 1, Ba, Tu, Nam, Sau,  
Bay, ChuNhat};
```

Hai = 1; Ba = 2; ... ; ChuNhat = 7

Enum (kiểu liệt kê)

#40

■ VD3:

```
enum Ngay {Hai = 1, Ba, Tu, Nam, Sau=10,  
Bay, ChuNhat};
```

```
Hai = 1; Ba = 2; ... ; Sau=10;
```

```
Bay=11;ChuNhat = 12
```

Struct (kiểu cấu trúc)

#41

- **Struct** dùng để nhóm các dữ liệu cùng liên quan đến một đối tượng nào đó.

- Khai báo :

```
struct <Tên cấu trúc>  
{  
    Danh sách các thuộc tính;  
}
```

- Truy xuất: <tên biến cấu trúc>.thuộc tính

Struct (kiểu cấu trúc)

#42

```
struct SV
{
    public string ten;
    public string maso;
}
static void Main(string[] args)
{
    SV a;
    a.ten = "Le Van Teo";
    a.maso = "002";
    Console.WriteLine("Ten: "+a.ten+" Ma so: "+a.maso);
    Console.ReadLine();
}
```

Identifier (định danh)

#43

- Định danh là việc xác định tên: biến, hàm, hằng, ...
- Phân biệt chữ hoa thường

Identifier (định danh)

#44

Quy ước đặt tên :

- Sử dụng 26 chữ cái (thường/ hoa), 10 chữ số
- Dấu nối (_)
- Không dùng chữ số ở đầu
- Không trùng với từ khoá

Biến & khai báo biến

#45

- Biến dùng để lưu trữ dữ liệu. Mỗi biến thuộc về một KDL nào đó.
- Khai báo:
 <KDL> tên biến;
- VD:
 int x;
 int a, b;

Biến & khai báo biến

#46

VD: Khai báo và khởi tạo:

```
int x = 5;
```

```
int y = x;
```

```
float z = 3.7;
```

```
float k = (float) x/2;
```

Toán tử số học

#47

Ký hiệu	Ý nghĩa	Ghi chú
+	Cộng	
-	Trừ	
*	Nhân	
/	Chia	Đối với số chia & bị chia là nguyên thì cho kết quả là phần nguyên
%	Chia lấy phần dư	Chỉ áp dụng cho số chia & bị chia là số nguyên
++x; x++	Tăng x 1 đơn vị	
--x; x--	Giảm x 1 đơn vị	

Ký hiệu so sánh và phép toán bit

#48

Ký hiệu	Ý nghĩa
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
==	Bằng
!=	Khác
&&	Và
	Hoặc
!	Phủ định

Ký hiệu	Ý nghĩa
&	Và bit
	Hoặc bit
>>	Dịch phải
<<	Dịch trái
^	Xor bit

1.1 Nhập xuất thông qua stream

#49

- Về cơ bản, stream là dãy các byte truyền qua path. Có hai stream quan trọng: **Input stream** và **Output stream**. Input stream được sử dụng để đọc dữ liệu (hoạt động read) và Output stream được sử dụng để ghi dữ liệu (hoạt động write).
- `Console.ReadLine();` // đọc dữ liệu
- `Console.WriteLine();` // ghi dữ liệu

Hàm xuất – Console.System

#50

- Write (Xuất xong không xuống hàng)
- WriteLine (Xuất xong xuống hàng)
- Xuất không định dạng

int a = 5;

double x = 7.534;

string s = "ABC";

Console.WriteLine("a = " +a);

Console.WriteLine("x = "+x+"; s = "+s);

Hàm xuất – Console.System

#51

Xuất có định dạng thập phân

```
float x = 7.53489F;
```

```
double y = 5.6482;
```

```
Console.WriteLine("x = {0: 0.0000};  
                    y = {1: 0.00} ", x, y);
```

Xuất ký tự đặc biệt

#52

Ký tự	Ý nghĩa
\'	Dấu nháy đơn
\''	Dấu nháy đôi
\\	Dấu chéo ngược “\”
\0	Null
\a	Alert : Tiếng bip
\b	Lùi về trước
\f	Form feed
\n	Xuống dòng
\r	Về đầu dòng
\t	Tab ngang

Hàm nhập – Console.System

#53

```
string s;
```

```
int n;
```

```
s = Console.ReadLine();
```

```
n = int.Parse(s);
```

Hoặc

```
int n;
```

```
n = int.Parse(Console.ReadLine());
```

Hàm nhập – Console.System

#54

Mẫu chung:

<KDL> Biến;

Biến = <KDL>.Parse(Console.ReadLine());

Hoặc

<KDL> Biến;

Biến = Convert.To<KDL.Net>(Console.ReadLine());

Bài tập

#55

Viết chương trình nhập vào thông tin:

- Mã số nhân viên
- Họ tên
- Hệ số lương (hs)
- Lương cơ bản (lcb)
- Phụ cấp (pc)

In tổng lương ra màn hình theo công thức:

$$\text{Tổng lương} = \text{lcb} * \text{hs} + \text{pc}$$

1.2 Tham số mặc nhiên

#56

- Khi khai báo tham số cho hàm (phương thức) chúng ta khai báo 2 loại tham số như sau:
 - Tham số hằng.
 - Tham số mặc định.

Cú pháp định nghĩa tham số hằng

#57

- `const <KDL> <ten_bien> = <giá_trị>;`
- hoặc `<KDL> const < ten_bien > = < giá_trị >;`
- Ví dụ:

```
void receiveInput(const int a, const float b)
{
    //.....
}
```


Cú pháp định nghĩa tham số hằng

#58

- `const <KDL> <ten_bien> = <giá_trị>;`
- hoặc `<KDL> const < ten_bien > = < giá_trị >;`
- Ví dụ:

```
void receiveInput(const int a, const float b)
{
//.....
}.
```

Khác với việc khởi tạo giá trị cho biến hằng số thông thường, tham số hằng sẽ được khởi tạo giá trị lúc gọi hàm. Khi gọi hàm và truyền đối số hằng vào hàm, hệ điều hành lúc đó mới cấp phát vùng nhớ cho các tham số hằng và biến cục bộ bên trong hàm.

Mục đích của việc khai báo tham số hằng là để đảm bảo rằng hàm đó không thể thay đổi giá trị của đối số truyền vào (cho dù truyền bằng tham biến (truyền địa chỉ)). Khi chương trình phát sinh lỗi, chúng ta có thể loại bỏ bớt trường hợp lỗi do hàm có tham số hằng gây ra, giúp chúng ta dễ dàng sửa lỗi hơn.

Tham số mặc định (default parameter)

#59

- Tham số mặc định là tham số có một giá trị khởi tạo tại thời điểm khai báo.
 - Khi người dùng không cung cấp đối số cho tham số mặc định, giá trị mặc định sẽ được sử dụng.
 - Khi người dùng cung cấp đối số cho tham số mặc định, tham số sẽ được gán lại bằng giá trị của đối số được truyền vào.
- Cách khai báo tham số mặc định**
 - Để khai báo tham số mặc định cho hàm, bạn chỉ cần sử dụng toán tử assignment (=) như lúc các bạn khởi tạo cho biến thông thường.
 - Ví dụ `void printValue(int x=5, int y = 10)`

```
{  
    Console.Write(x);  
    Console.Write(y);  
}
```

Vì tham số mặc định là tham số tùy chọn, chương trình không ràng buộc người dùng phải truyền đối số cho tham số mặc định khi gọi hàm.

```
// Gọi hàm  
printValue();
```

Cấu trúc điều khiển

#60

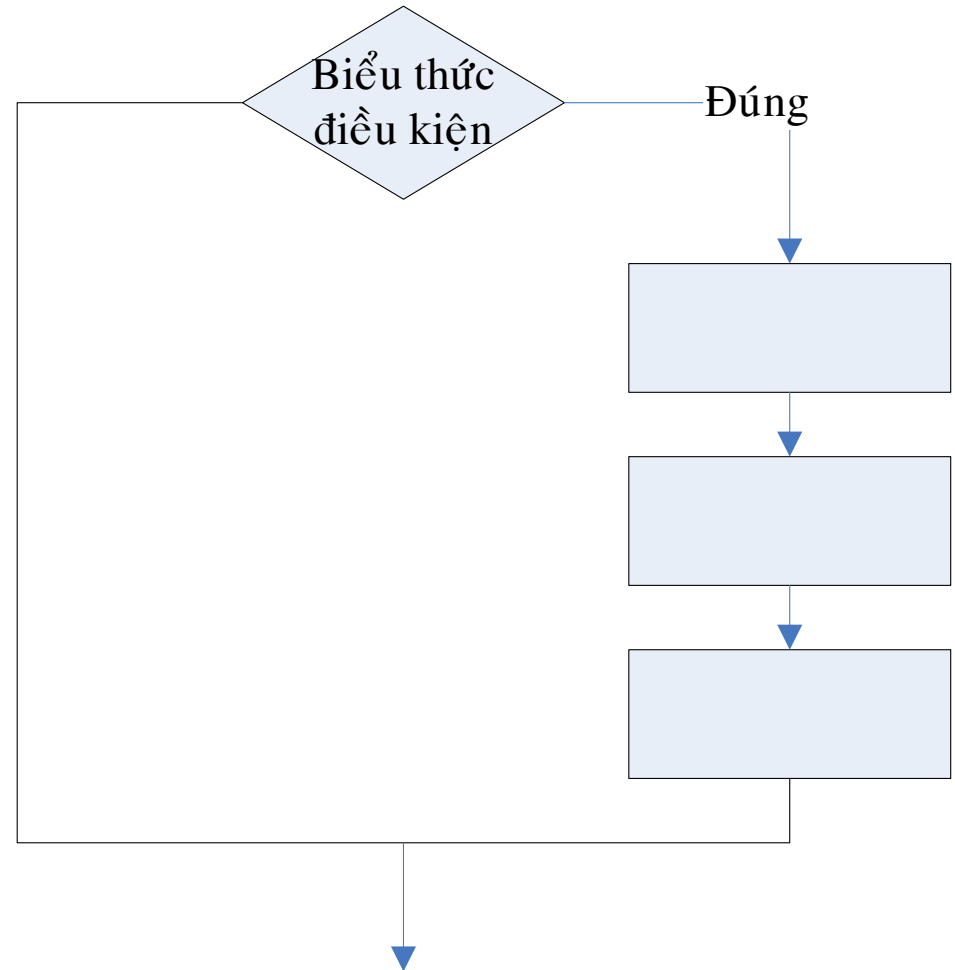
- Rẽ nhánh : if...else
- Lựa chọn : switch...case
- Lặp : for, while, do...while, **foreach**
- Các cấu trúc khác : goto, break, continue

Cấu trúc rẽ nhánh

#61

```
if (biểu thức điều kiện)  
{  
    <khối lệnh> ;  
}
```

Nếu biểu thức điều kiện cho kết quả khác không (true) thì thực hiện khối lệnh.



Cấu trúc rẽ nhánh (tt)

#6 *Ví dụ: Nhập vào số nguyên n. Kiểm tra nếu $n > 0$ tăng n lên 1 đơn vị. Xuất kết quả.*

```
static void Main(string[] args)
{
    int n;

    Console.Write("Nhap vao mot so nguyen: ");
    n = int.Parse(Console.ReadLine());

    if (n > 0)
        n++;

    Console.WriteLine("Ket qua: n = " + n);
}
```

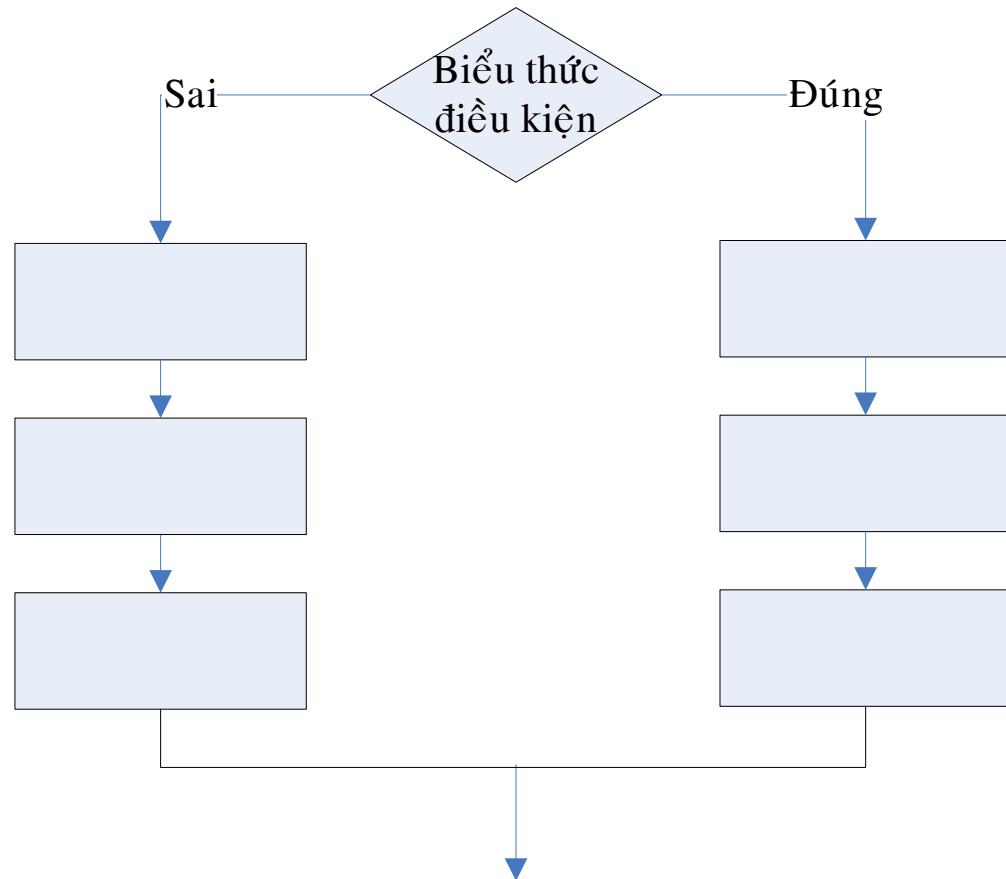
Cấu trúc rẽ nhánh (tt)

#63

if (biểu thức điều kiện)

```
{  
    <khối lệnh 1>;  
}  
else  
{  
    <khối lệnh 2>;  
}
```

Nếu biểu thức điều kiện cho kết quả khác không thì thực hiện khối lệnh 1, ngược lại thì cho thực hiện khối lệnh thứ 2.



VD: Giải và biện luận PT: $ax+b=0$

#64

```
static void Main(string[] args)
{
    int a, b;
    Console.Write("Nhap vao a: ");
    a = int.Parse(Console.ReadLine());
    Console.Write("Nhap vao b: ");
    b = int.Parse(Console.ReadLine());
    if (a == 0)
        if (b == 0)
            Console.WriteLine("PT VSN");
        else
            Console.WriteLine("PT VN");
    else
        Console.WriteLine("Ng cua PT: {0:0.00}", (float)-b/a);
}
```

Cấu trúc lựa chọn

#65

switch (biểu thức)

{

case n1:

các câu lệnh ;

break ;

case n2:

các câu lệnh ;

break ;

.....

case nk:

<các câu lệnh> ;

break ;

[default: các câu lệnh]

break;

}

KQ phải là nguyên

VD: Nhập vào số nguyên n có giá trị từ 1 đến 5.
In cách đọc của số đó ra màn hình

#66

```
static void Main(string[] args)
{
    int n;
    Console.Write("Nhap vao n (1<=n<=5): ");
    n = int.Parse(Console.ReadLine());
    switch (n)
    {
        case 1: Console.WriteLine("So mot");    break;
        case 2: Console.WriteLine("So hai");    break;
        case 3: Console.WriteLine("So ba");     break;
        case 4: Console.WriteLine("So bon");    break;
        case 5: Console.WriteLine("So nam");    break;
        default : Console.WriteLine("Gia tri khong hop le");
                break;
    }
}
```

Bài tập

#67

- Nhập vào hai số nguyên a , b . In ra màn hình giá trị lớn nhất.
- Cho ba số a , b , c đọc vào từ bàn phím. Hãy tìm giá trị lớn nhất của ba số trên và in ra kết quả.

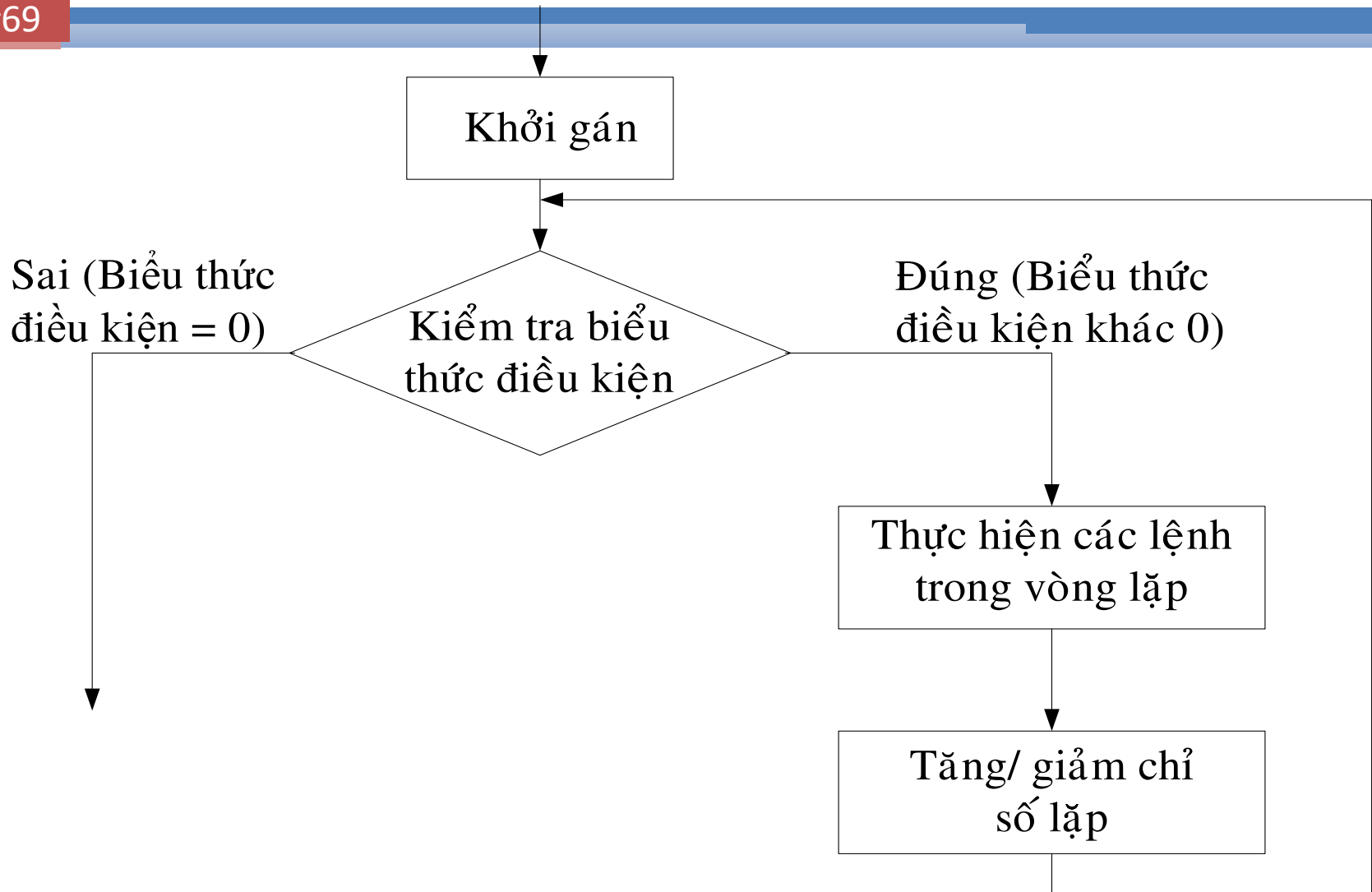
Cấu trúc lặp

#68

- while
- for
- do...while
- foreach

Cấu trúc lặp while và for

#69



Hoạt động cấu trúc lặp while và for

#70

- Bước 1: Thực hiện khởi gán
- Bước 2: Kiểm tra biểu thức điều kiện
 - Nếu kết quả là true thì cho thực hiện các lệnh của vòng lặp, thực hiện biểu thức tăng/ giảm.
Quay trở lại bước 2.
 - Ngược lại kết thúc vòng lặp.

Cấu trúc lặp while

#71

< Khởi gán>;

while (<biểu thức điều kiện>)

{

lệnh/ khối lệnh;

<tăng/giảm chỉ số lặp>;

}

VD: Xuất ra màn hình 10 dòng chữ ABC

#72

```
static void Main(string[] args)  
{  
    int d = 1;  
    while (d <= 10)  
    {  
        Console.WriteLine("Dong {0}: ABC", d);  
        d++;  
    }  
}
```

Cấu trúc lặp for

#73

```
for (<Khởi gán>;<biểu thức ĐK>;<tăng/giảm>)  
{  
    <khối lệnh>;  
}
```


VD: Xuất ra màn hình 10 dòng chữ ABC

#74

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

{

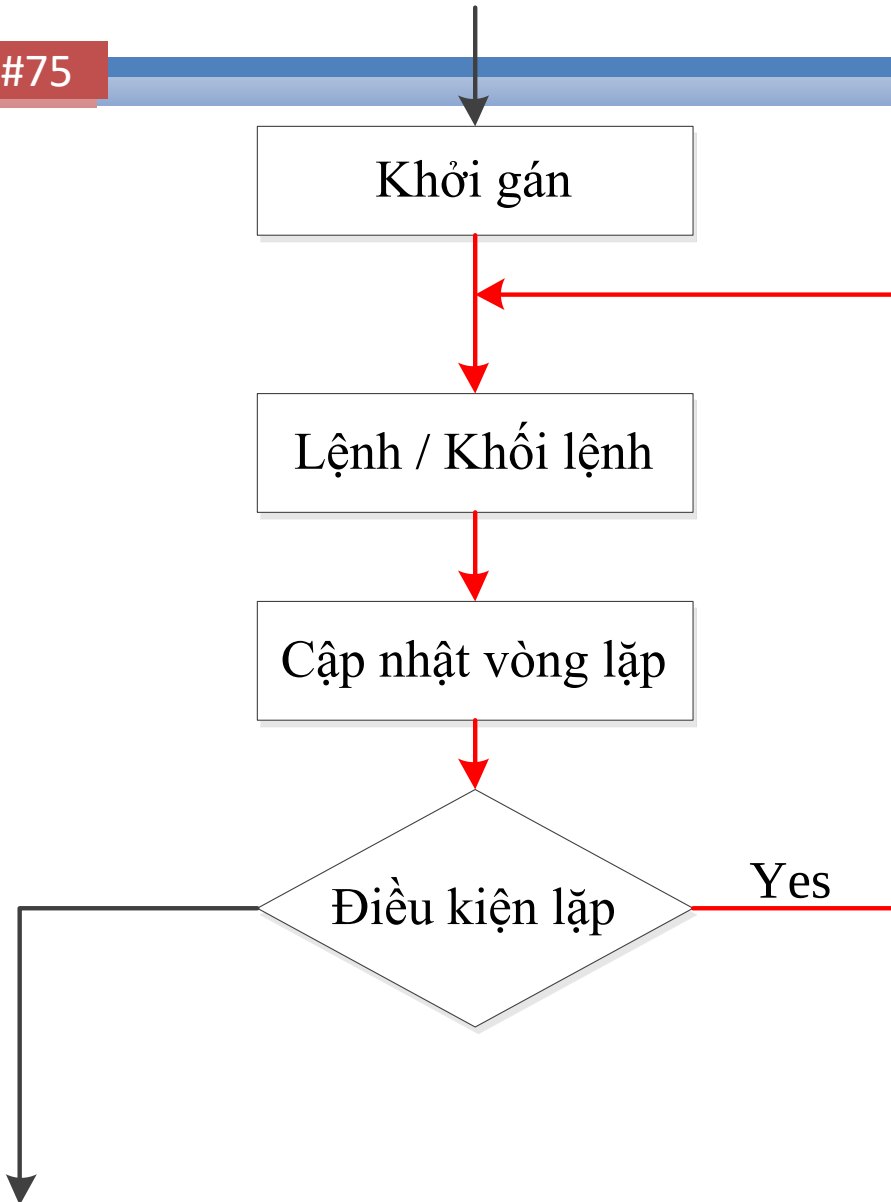
    for (int d = 1; d <= 10; d++)

        Console.WriteLine("Dong {0}: ABC", d);

}
```

Cấu trúc lặp do...while

#75



do

{

< khối lệnh > ;

} while (biểu thức ĐK);

Thực hiện khối lệnh cho đến khi biểu thức điều kiện là false

Cấu trúc lặp foreach

#76

- Sử dụng cho mảng

foreach (<KDL mảng> <tên biến> in <tên mảng>)

{

 Khối lệnh;

}

Xét từng phần tử trong mảng

VD: Tính tổng các phần tử chẵn trong mảng

#77

```
static void Main(string[] args)
{
    int s=0;
    int [ ] a = new int [5] {3, 8, 7, 1, 6};
    foreach(int m in a)
        if(m%2==0)
            s+=m;
    Console.WriteLine("Tong chan = " +s);
}
```

Bài tập

#78

- Viết chương trình đếm số ước số của số nguyên dương.
- Viết chương trình in ra màn hình hình chữ nhật đặc kích thước (m, n) nhập từ bàn phím).

Ví dụ: Nhập $m=5, n=4$

```
*   *   *   *   *  
  
*   *   *   *   *  
  
*   *   *   *   *  
  
*   *   *   *   *
```

1.3 Tái định nghĩa hàm

#79

- Xem xét một hàm `GetTime` trả về thời gian hiện tại của ngày theo các tham số truyền vào của nó, và giả sử rằng cần có hai biến thể của hàm này: một trả về thời gian theo giây tính từ nửa đêm, và một trả về thời gian theo giờ, phút, giây. Rõ ràng 2 hàm này phục vụ cùng một mục đích là trả về thời gian, nên không có lý do gì lại đặt tên hàm cho nó với những cái tên khác nhau.
- C# cho phép các hàm được tái định nghĩa, nghĩa là cùng hàm có thể có hơn một định nghĩa:
 - `long GetTime (void); // số giây tính từ nửa đêm`
 - `DateTime GetTime (out int hours, out int minutes, out int seconds);`
- Khi hàm `GetTime` được gọi, trình biên dịch so sánh số lượng và kiểu các đối số trong lời gọi với các định nghĩa của hàm `GetTime` và chọn một hàm khớp với lời gọi.
- Ví dụ: `int h, m,s;`
 - `long t = GetTime(); // gọi hàm GetTime(void)`
 - `DateTime t=GetTime(out h, out m,out s); // gọi hàm GetTime (out int hours, out int minutes, out int seconds);`

Method - Phương thức

#80

Phương thức (hay còn gọi là hàm) là một đoạn chương trình độc lập thực hiện trọn vẹn một công việc nhất định sau đó trả về giá trị cho chương trình gọi nó, hay nói cách khác hàm là sự chia nhỏ của chương trình.

Phương thức

#81

Mục đích sử dụng phương thức:

- Khi có một công việc giống nhau cần thực hiện ở nhiều vị trí.
- Khi cần chia một chương trình lớn phức tạp thành các đơn thể nhỏ (hàm con) để chương trình được trong sáng, dễ hiểu trong việc xử lý, tính toán.

Phương thức

#82

Mẫu tổng quát của phương thức

<phạm vi> <KDL> TênPhươngThức([tham số]);

Phạm vi

- Xác định phạm vi hay cách phương thức được gọi (sử dụng)
- Các từ khoá phạm vi : protected, private, public, static

Phương thức

#83

KDL của phương thức (đầu ra), gồm 2 loại

- void: Không trả về giá trị
- float / int / long / string / kiểu cấu trúc / ... :
Trả về giá trị có KDL tương ứng với kết quả xử lý

Phương thức

#84

- Tên phương thức : Đặt tên theo qui ước sao cho phản ánh đúng chức năng thực hiện của phương thức
- Danh sách các tham số (nếu có) : đầu vào của phương thức (trong một số trường hợp có thể là đầu vào và đầu ra của phương thức nếu kết quả đầu ra có nhiều giá trị - Tham số này gọi là tham chiếu)

Khi hàm xử lý biến toàn cục thì không cần tham số

#85

```
static int a, b;
static void Nhap()
{
    Console.Write("Nhap a: ");
    a = int.Parse(Console.ReadLine());
    Console.Write("Nhap b: ");
    b = int.Parse(Console.ReadLine());
}
static void Xuat()
{
    Console.WriteLine("a = {0}; b = {1}", a, b);
}
static void Main(string[] args)
{
    Nhap();
    Xuat();
}
```

Phương thức không trả về giá trị

#86

static void TênPhươngThức([danh sách các tham số])

{

Khai báo các biến cục bộ

Các câu lệnh hay lời gọi đến phương thức khác.

}

- Gọi hàm: TênPhươngThức(danh sách tên các đối số);
- Những phương thức loại này thường rơi vào những nhóm chức năng: Nhập / xuất dữ liệu, thống kê, sắp xếp, liệt kê

Viết chương trình nhập số nguyên dương n và in ra màn hình các ước số của n

#87

- Input: số nguyên dương (Xác định tham số)
- Output: In ra các ước số của n (Xác định KDL trả về của phương thức)
 - Xuất $\rightarrow$ Không trả về giá trị $\rightarrow$ KDL là ***void***.
 - Xác định tên phương thức: Phương thức này dùng in ra các US của n nên có thể đặt là *LietKeUocSo*

static void LietKeUocSo(uint n)

```
static void LietKeUocSo(uint n)
{
```

#88

```
    for (int i = 1; i <= n; i++)
        if (n % i == 0)
            Console.Write("{0}\t", i);
}
```

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

```
    uint n;
```

```
    Console.Write("Nhap so nguyen duong n: ");
```

```
    n=uint.Parse(Console.ReadLine());
```

```
    Console.Write("Cac uoc so cua {0}: ", n);
```

```
    LietKeUocSo(n);
```

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

```
}
```

Phương thức có trả về kết quả

static <KDL> TênPhươngThức([tham số])

#89

{

<KDL> kq;

Khai báo các biến cục bộ

Các câu lệnh hay lời gọi đến phương thức khác.

return kq;

}

Gọi hàm:

<KDL> Tên biến = TênPhươngThức(tên các đối số);

Những phương thức này thường rơi vào các nhóm: *Tính tổng, tích, trung bình, đếm, kiểm tra, tìm kiếm*

Viết chương trình nhập số nguyên dương n và tính
$$S_n = 1 + 2 + 3 + \dots + n \quad ; n > 0$$

#90

- Input: số nguyên dương n (Xác định tham số)
- Output: Tổng S (Xác định KDL trả về của phương thức)
 - Trả về giá trị tổng (S).
 - S là tổng các số nguyên dương nên S cũng là số nguyên dương $\rightarrow$ Kiểu trả về của hàm là ulong.
- Xác định TênPhươngThức: Dùng tính tổng S nên có thể đặt là TongS

static ulong TongS(uint n)

```
static ulong TongS(uint n)
{
```

#91

```
    ulong kq = 0;
    for (uint i = 1; i <= n; i++)
        kq += i;
    return kq;
}
```

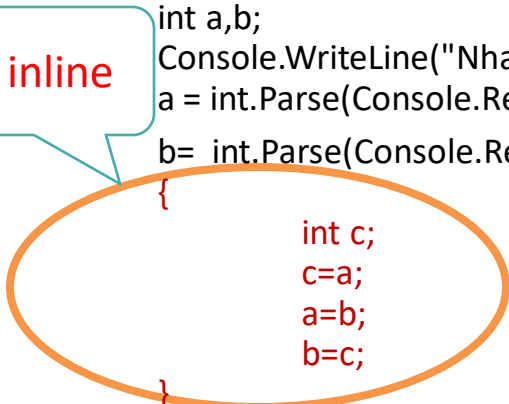
```
static void Main(string[] args)
{
```

```
    ulong S;
    uint n;
    Console.Write("Nhap vao so nguyen n: ");
    n = uint.Parse(Console.ReadLine());
    S = TongS(n);
    Console.Write("Tong tu 1 den n: " + S);
    Console.ReadLine();
}
```

1.4 Hàm inline

- ❖ Một đoạn code được định nghĩa trong cặp { } trong một hàm mà không thông qua gọi hàm.
- ❖ Khi trình biên dịch gặp cặp { } bên trong một hàm thì nó hiểu đó là hàm inline.

```
static void Main(string[] args)
{
    int a,b;
    Console.WriteLine("Nhap vao a va b : ");
    a = int.Parse(Console.ReadLine());
    b= int.Parse(Console.ReadLine());
    {
        int c;
        c=a;
        a=b;
        b=c;
    }
    Console.WriteLine(a +" "+b);
    Console.ReadKey();
}
```



```
Static HoanVi(out int a, out int b)
{
    int c;
    c=a;
    a=b;
    b=c;
}

static void Main(string[] args)
{
    int a,b;
    Console.WriteLine("Nhap vao a va b : ");
    a = int.Parse(Console.ReadLine());
    b= int.Parse(Console.ReadLine());
    HoanVi(out a,out b);
    Console.WriteLine(a +" "+b);
    Console.ReadKey();
}
```

Trong ví dụ trên thay vì chúng ta phải viết một hàm hoán vị 2 số a và b, và phải gọi hàm hoán vị này để thực hiện hoán vị 2 số làm như vậy thì trình biên dịch sẽ nhảy đến hàm hoán vị này để thực hiện, sau khi kết thúc hàm này thì trình biên dịch sẽ quay lại chương trình chính sau lời gọi hàm. Điều này sẽ làm chương trình chạy chậm hơn. Do đó hàm inline sẽ giúp tiết kiệm thời gian hơn và giúp chương trình chạy nhanh hơn.

1.5 Tham chiếu, truyền tham số bằng tham chiếu

#93

- Tham số: Các thông số đầu vào của một hàm được gọi là tham số của hàm.
- Phân loại tham số: có 2 loại tham số là tham trị và tham biến.
 - + Tham trị: giá trị không đổi.
 - + Tham biến: giá trị thay đổi.
- Cấp phát bộ nhớ:
 - + Tham trị: Cấp phát.
 - + Tham biến: Không cấp phát bộ nhớ khi hàm được gọi thực hiện mà sử dụng bộ nhớ của đối số tương ứng.

Tham số và hàm

#94

- Tham số lưu kết quả xử lý của hàm: **out** (thường dùng cho trường hợp nhập dữ liệu, kết quả hàm có nhiều giá trị)
- Tham số vừa làm đầu vào và đầu ra: **ref**
- Dùng từ khóa **ref** hoặc **out** trước KDL của khai báo tham số và trước tên đối số khi gọi phương thức.

Tham số là tham chiếu

#95

Dùng từ khóa **ref** bắt buộc phải khởi gán giá trị ban đầu cho đối số trước khi truyền vào khi gọi phương thức (Nếu dùng **out** thì không cần thiết)

Hoán vị 2 số nguyên a, b cho trước

#96

Đánh giá kết quả khi viết chương trình với hai trường hợp sau

1. Trường hợp không dùng tham chiếu
2. Trường hợp dùng tham chiếu: **ref**

Không dùng tham chiếu

```
static void HoanVi(int a, int b)
```

```
#97
```

```
{
```

```
    int tam = a;
```

```
    a = b;
```

```
    b = tam;
```

```
    Console.WriteLine("Trong HoanVi: a = " + a + ";b = " + b);
```

```
}
```

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

```
{
```

```
    int a = 5, b = 21;
```

```
    Console.WriteLine("Truoc HoanVi: a = {0}; b = {1}", a, b);
```

```
    HoanVi(a, b);
```

```
    Console.WriteLine("Sau HoanVi: a = " + a + ";b = " + b);
```

```
}
```


Dùng tham chiếu

```
static void HoanVi(ref int a, ref int b)
```

```
#98
```

```
{
```

```
    int tam = a;
```

```
    a = b;
```

```
    b = tam;
```

```
    Console.WriteLine("Trong HoanVi: a = " + a + ";b = " + b);
```

```
}
```

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

```
{
```

```
    int a = 5, b = 21;
```

```
    Console.WriteLine("Truoc HoanVi: a = {0}; b = {1}", a, b);
```

```
    HoanVi(ref a, ref b);
```

```
    Console.WriteLine("Sau HoanVi: a = " + a + ";b = " + b);
```

```
}
```

Ví dụ - sử dụng tham chiếu out

```
static void Nhap(out int a, out int b)
```

```
#99 {
```

```
    Console.Write("Nhap a: ");  
    a = int.Parse(Console.ReadLine());  
    Console.Write("Nhap b: ");  
    b = int.Parse(Console.ReadLine());
```

```
}
```

```
static int Tong(int a, int b)
```

```
{
```

```
    return a + b;
```

```
}
```

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

```
{
```

```
    int a, b;
```

```
    Nhap(out a, out b);
```

```
    s=Tinh(a, b);
```

```
    Console.WriteLine("{0}+{1}={2}", a, b, s);
```

```
}
```

Bài tập

#100

- Viết chương trình tính diện tích và chu vi của hình chữ nhật.
- Viết chương trình tính diện tích và chu vi hình tròn.
- Nhập vào 3 số thực a , b , c và kiểm tra xem chúng có lập thành 3 cạnh của một tam giác hay không? Nếu có hãy tính diện tích, chiều dài mỗi đường cao của tam giác và in kết quả ra màn hình.

Bài tập

#101

- Viết chương trình nhập 2 số nguyên dương a, b . Tìm USCLN & BSCNN.
- Viết chương trình nhập số nguyên dương n , tính tổng các ước số của n .

Ví dụ: Nhập $n=6$

Tổng các ước số từ 1 đến n : $1+2+3+6=12$.

- Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không?

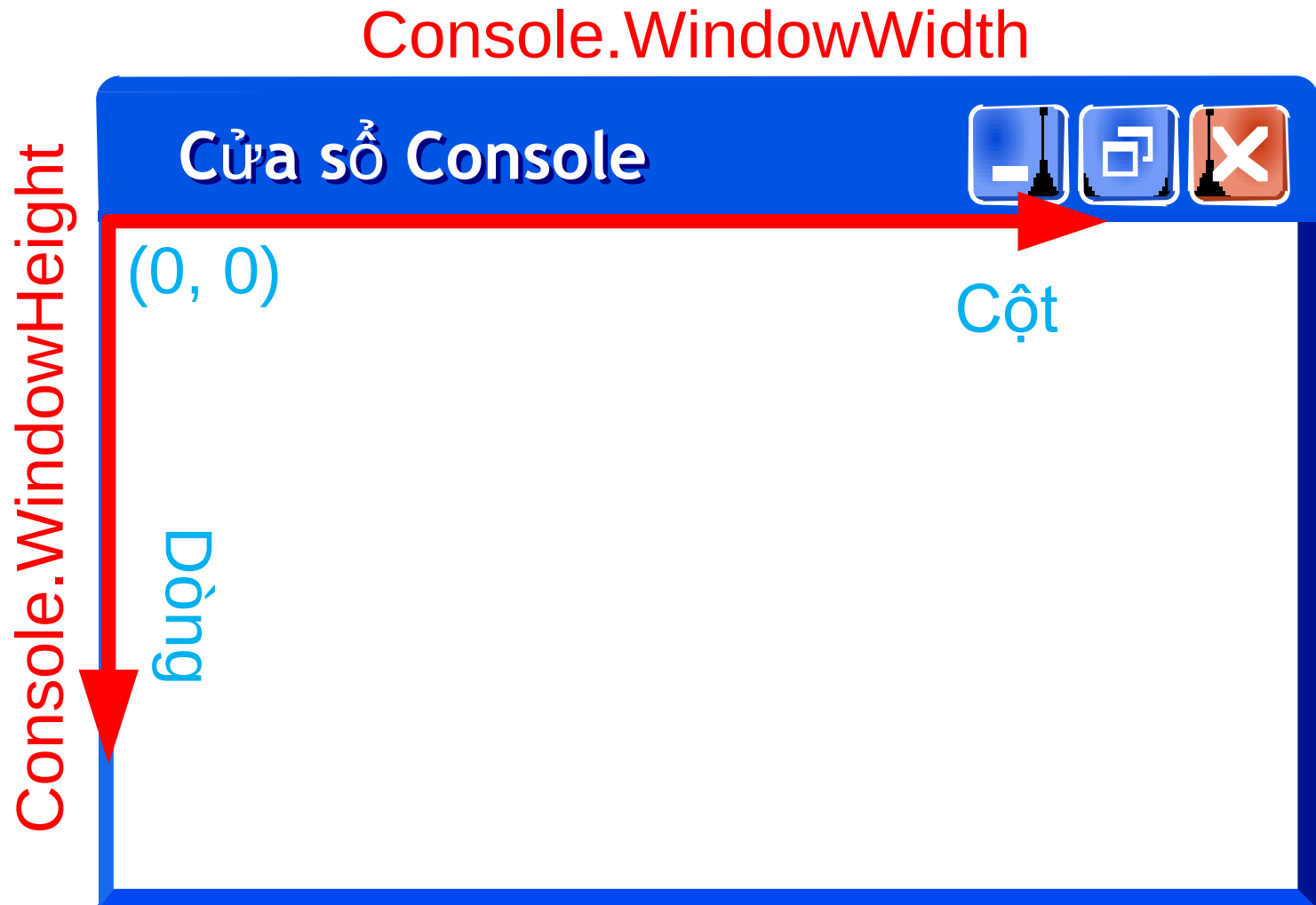
Thao tác trên Console

#102

- Các ứng dụng Console là môi trường tốt cho những người mới bắt đầu học lập trình và giải thuật.
- Các ứng dụng Console chạy trên môi trường DOS, và có giao diện dòng lệnh.
- Sử dụng: ***Console.<tên thao tác>***

Thao tác trên Console

#103



Di chuyển đầu nháy

#104

Console.SetCursorPosition(cột, dòng);

VD: Xuất chuỗi Hello vị trí dòng 20 và cột 10

`Console.SetCursorPosition(10, 20);`

`Console.Write("Hello");`

Lấy số ngẫu nhiên

#105

Sử dụng lớp Random

Phương thức	Miền giá trị phát sinh
Next()	[0...2,147,483,646]
Next(max)	[0...max -1]
Next(min, max)	[min..max -1]
NextDouble()	Số thực từ 0.0 đến 1.0

#106

```
int songaunhien;
```

```
double sothuc;
```

```
Random rd = new Random();
```

```
songaunhien = rd.Next();
```

```
Console.WriteLine(songaunhien);
```

```
songaunhien = rd.Next(100);
```

```
Console.WriteLine(songaunhien);
```

```
songaunhien = rd.Next(10, 100);
```

```
Console.WriteLine(songaunhien);
```

```
sothuc = rd.NextDouble();
```

```
Console.WriteLine(sothuc);
```

Lấy ngày giờ hệ thống

#107

■ Sử dụng lớp DateTime

DateTime d = DateTime.Now

Thuộc tính	Ý nghĩa
<code>int iNg = d.Day;</code>	Lấy ngày
<code>int iTh = d.Month;</code>	Lấy tháng
<code>int iNm = d.Year;</code>	Lấy năm
<code>int iGio = d.Hour;</code>	Lấy giờ
<code>int iPhut = d.Minute;</code>	Lấy phút
<code>int iGiay = d.Second;</code>	Lấy giây
<code>int iMGiay = d.Millisecond;</code>	Lấy phần ngàn của giây

Xác định phím nhấn trong Console

#108

```
ConsoleKeyInfo k = Console.ReadKey()
```

→ Nếu không muốn hiển thị phím được nhấn thì truyền true vào tham số phương thức

ReadKey:

```
ConsoleKeyInfo k = Console.ReadKey(true)
```

Xác định phím nhấn trong Console

#109

Cấu trúc ConsoleKeyInfo

- Thuộc tính **Key**: Chứa tên phím được nhấn.
Xác định bằng cách so giá trị với các phím trong ConsoleKey
- Thuộc tính **Modifiers**: Cho biết phím **Ctrl**, **Shift** hoặc **Alt** có được nhấn kèm

Xác định phím nhấn trong Console

#110

Xác định phím Ctrl, Alt hay Shift được nhấn bằng cách dùng phép toán AND (&) thuộc tính Modifiers với :

- ConsoleModifiers.Ctrl
- ConsoleModifiers.Alt
- ConsoleModifiers.Shift

Kiểm tra xem người dùng nhấn phím A, Shift + A hoặc phím up arrow

```
ConsoleKeyInfo k;
```

```
k = Console.ReadKey(true);
```

#111

```
switch(k.Key)
```

```
{
```

```
    case ConsoleKey.A:
```

```
        if (k.Modifiers!=0 && (k.Modifiers&ConsoleModifiers.Shift)!=0)
```

```
            Console.Write("Ban nhan phim Shift + A");
```

```
        else
```

```
            Console.Write("Ban nhan phim A");
```

```
        break;
```

```
    case ConsoleKey.UpArrow:
```

```
        Console.Write("Ban nhan phim mui ten huong len");
```

```
        break;
```

```
    default: Console.Write("Ban nhan phim khac");
```

```
}
```

Kiểm tra phím có được nhấn hay ko?

#112

- Có nhấn phím:

`Console.KeyAvailable = true`

- Không nhấn phím:

`Console.KeyAvailable = false`

Hiển thị giờ hệ thống cho đến khi nhấn phím bất kỳ

#113

```
while (!Console.KeyAvailable)

{

    DateTime d = DateTime.Now;

    Console.Write("{0:00}:{1:00}:{2:00}",

                  d.Hour, d.Minute, d.Second);

    Thread.Sleep(1000); //using System.Threading;

    Console.Clear();

}
```


Thiết lập màu

#114

- Màu nền của chữ

`Console.BackgroundColor = hằng số màu;`

VD: `Console.BackgroundColor = ConsoleColor.Green;`

- Màu chữ

`Console.ForegroundColor = hằng số màu;`

VD: `Console.ForegroundColor = ConsoleColor.Red;`

- Xóa nội dung của sổ console: `Console.Clear();`

Hằng số màu

#115

Các hằng số màu

ConsoleColor.Black	ConsoleColor.DarkRed
ConsoleColor.Blue	ConsoleColor.DarkYellow
ConsoleColor.Cyan	ConsoleColor.Gray
ConsoleColor.DarkBlue	ConsoleColor.Green
ConsoleColor.DarkCyan	ConsoleColor.Magenta
ConsoleColor.DarkGray	ConsoleColor.Red
ConsoleColor.DarkGreen	ConsoleColor.White
ConsoleColor.DarkMagenta	ConsoleColor.Yellow

Ẩn hiện dấu nháy

#116

- Ẩn dấu nháy

```
Console.CursorVisible = false;
```

- Hiện dấu nháy

```
Console.CursorVisible = true;
```

VD: Hiển thị giờ phút giây trên màn hình với màu nền là màu xanh và màu chữ là đỏ

#117

```
Console.CursorVisible = false;
ConsoleColor mauchu = Console.ForegroundColor;
ConsoleColor maunen = Console.BackgroundColor;
Console.BackgroundColor = ConsoleColor.Green;
while (!Console.KeyAvailable)
{
    Console.ForegroundColor = ConsoleColor.Red;
    DateTime d = DateTime.Now;
    Console.SetCursorPosition(5, 5);
    Console.Write("{0:00}:{1:00}:{2:00}", d.Hour, d.Minute, d.Second);
    Thread.Sleep(1000); //using System.Threading;
    Console.ForegroundColor = Console.BackgroundColor;
    Console.SetCursorPosition(5, 5);
    Console.Write("{0:00}:{1:00}:{2:00}", d.Hour, d.Minute, d.Second);
}
```

Thao tác cơ bản trên mảng 1 chiều

#118

- Khởi tạo mảng

`int [] a = new int[5] {4, 7, 1, 21, 9};`

hoặc: `int [] a = {4, 7, 1, 21, 9};`

- Truy xuất mảng: `<Tên_Mảng>[vị trí];`

Vị trí được đánh số từ 0 đến số phần tử $n-1$

Ví dụ: `a[4]` sẽ cho giá trị là 9

Thao tác cơ bản trên mảng 1 chiều

Nhập xuất mảng 1 chiều

#119

```
public static void Main()
{
    int n;
    Console.Write("Nhap kích thước mảng: ");
    n = int.Parse(Console.ReadLine());
    int[] a = new int[n];
    Nhap(a, n);
    Xuat(a, n);
}
}
```

Thao tác cơ bản trên mảng 1 chiều

Nhập xuất mảng 1 chiều

#120

```
static void Nhap(int[] a, int n)
{
    for (int i = 0; i < n; i++)
    {
        Console.Write("Nhap phan tu thu {0}:", i);
        a[i] = int.Parse(Console.ReadLine());
    }
}
```

Thao tác cơ bản trên mảng 1 chiều

Nhập xuất mảng 1 chiều

#121

```
static void Xuat(int[] a, int n)
{
    for(int i = 0; i < n; i++)
    {
        Console.Write(a[i] + "\t");
    }
}
```


Thao tác cơ bản trên mảng 1 chiều

Nhập xuất mảng 1 chiều

#122

Hoặc

```
static void Xuat(int[] a)
{
    foreach(int k in a)
    {
        Console.Write(k + "\t");
    }
}
```

Bài tập

#123

Cho mảng 1 chiều số nguyên, hãy viết các hàm sau:

- Tìm phần tử có giá trị x
- Tìm phần tử có giá trị nhỏ nhất
- Tìm vị trí của phần tử có giá trị lớn nhất

Thao tác cơ bản trên ma trận

- Khởi tạo ma trận

#124

```
int[,] mt = new int[3, 5] { {2, 4, 8, 9, 7},  
                             {4, 8, 11, 10, 3},  
                             {21, 7, 6, 5, 0}};
```

hoặc

```
int[,] mt = { {2, 4, 8, 9, 7},  
              {4, 8, 11, 10, 3},  
              {21, 7, 6, 5, 0}};
```

- Truy xuất ma trận: <Tên_Mảng>[vị trí dòng, vị trí cột];

Vị trí được đánh số từ 0

VD: mt[1, 3] sẽ cho giá trị là 10

Thao tác cơ bản trên ma trận

Nhập xuất ma trận

#125

```
public static void Main()
{
    int d, c;
    Console.Write("Nhap so dong: ");
    d = int.Parse(Console.ReadLine());
    Console.Write("Nhap so cot: ");
    c = int.Parse(Console.ReadLine());
    int[,] mt = new int [d, c];
    Nhap(mt, d, c);
    Xuat(mt, d, c);
}
```

Thao tác cơ bản trên ma trận

Nhập xuất ma trận

#126

```
static void Nhap(int[,] mt, int d, int c)
{
    for (int i = 0; i < d; i++)
        for (int j = 0; j < c; j++)
        {
            Console.Write("Nhap phan tu [{0},{1}]: ", i, j);
            mt[i,j] = int.Parse(Console.ReadLine());
        }
}
```

Thao tác cơ bản trên ma trận

Nhập xuất ma trận

#127

```
static void Xuat(int[,] mt, int d, int c)
{
    for (int i = 0; i < d; i++)
    {
        for (int j = 0; j < c; j++)
            Console.Write(mt[i,j] + "\t");
        Console.WriteLine();
    }
}
```

Bài tập

#128

Cho ma trận số nguyên, hãy viết các hàm sau:

- Tìm phần tử có giá trị x
- Tìm phần tử có giá trị nhỏ nhất
- Tìm vị trí của phần tử có giá trị lớn nhất
- Tính giá trị trung bình các phần tử trong ma trận

Thao tác cơ bản trên chuỗi ký tự

#129

Ghép chuỗi: +

```
string a = "Xin";
```

```
string b = "chào";
```

```
string c = a + " " + b; // c = "Xin chào"
```

Hoặc

```
string.Concat(a, " ", b); → a = "Xin Chào"
```


Thao tác cơ bản trên chuỗi ký tự

#130

Lấy chuỗi con: Substring()

string s;

s = "Lay chuoi con".Substring(4);

Lấy chuỗi con tính từ vị trí thứ 4 trở về sau: s
= "*chuoi con*"

s = "Lay chuoi con".Substring(4, 5);

Lấy chuỗi con từ vị trí thứ 4 và lấy chuỗi con
có chiều dài là 5:

s = "*chuoi*"

Thao tác cơ bản trên chuỗi ký tự

#131

- Thay thế chuỗi con

Replace(chuỗi cần thay, chuỗi thay thế)

string s;

s = "thay the chuoì.".Replace('t', 'T');

→ s = "Thay The chuoì"

s = "thay the chuoì.".Replace("th", "TH");

→ s = "THay THe chuoì"

Thao tác cơ bản trên chuỗi ký tự

#132

▪ Định dạng chuỗi:

Format (định dạng, đổi số cần định dạng);

Ký tự	Mô tả	Ví dụ	Kết quả
C hoặc c	Tiền tệ (Currency)	<code>string.Format("{0:C}", 2.5);</code> <code>string.Format("{0:C}", -2.5);</code>	\$2.50 (\$2.50)
D hoặc d	Decimal	<code>string.Format("{0:D5}", 25);</code>	00025
E hoặc e	Khoa học (Scientific)	<code>string.Format("{0:E}", 250000);</code>	2.500000E+005
F hoặc f	Cố định phần thập phân (Fixed-point)	<code>string.Format("{0:F2}", 25);</code> <code>string.Format("{0:F0}", 25);</code>	25.00 25
G hoặc g	General	<code>string.Format("{0:G}", 2.5);</code>	2.5
N hoặc n	Số (Number)	<code>string.Format("{0:N}", 2500000);</code>	2,500,000.00
X hoặc x	Hệ số 16 (Hexadecimal)	<code>string.Format("{0:X}", 250);</code> <code>string.Format("{0:X}", 0xffff);</code>	FA FFFF

Thao tác cơ bản trên chuỗi ký tự

#133

- Chiều dài chuỗi: Thuộc tính Length

```
string s = "Xin chao";
```

```
int n = s.Length; // n = 8
```

Thao tác cơ bản trên chuỗi ký tự

#134

- Tách chuỗi con theo ký hiệu phân cách cho trước
string.**Split**(danh sách ký tự phân cách)

VD: In ra màn hình các từ trên từng dòng, mỗi từ cách nhau bằng dấu phẩy (,) hoặc khoảng trắng

```
string s = "Hom nay, ngay 02 thang 03 nam 2010";
```

```
foreach (string tu in s.Split(' ', ','))
```

```
    if (tu != "")
```

```
        Console.WriteLine(tu);
```

Thao tác cơ bản trên chuỗi ký tự

#135

VD: In ra từ có độ dài dài nhất trong chuỗi cho trước (từ cách nhau bằng khoảng trắng hoặc dấu chấm câu)

```
string s = "Hom nay, ngay 02 thang 03 nam 2010";
```

```
string tumax = "";
```

```
foreach (string tu in s.Split(' ', ',', '!', '?', ';'))
```

```
{
```

```
    if (tu != "" && tu.Length > tumax.Length)
```

```
        tumax = tu;
```

```
}
```

```
Console.WriteLine("Tu dai nhat: " + tumax);
```

Bài tập

#136

- Viết hàm đếm số ký tự x cho trước trong chuỗi s
- Viết hàm đảo các ký tự trong chuỗi s
- Viết hàm đảo các từ của chuỗi s
- Viết hàm xóa từ x có trong chuỗi s

Thao tác trên File - System.IO

#137

Gồm 2 loại file: Văn bản (text) và nhị phân (binary)

- Bước 1: Khai báo đối tượng file
- Bước 2: Mở file (đọc/ ghi)
- Bước 3: Thao tác trên file
- Bước 4: Đóng file

File text

#138

- Đọc file: đối tượng StreamReader

Phương thức đọc: `ReadLine()`;

- Ghi file: đối tượng StreamWriter

Phương thức ghi: `WriteLine()`;

- Đóng file: Phương thức `Close()`;

File Text – Ví dụ

#139

```
static void TaoFile(string tenfile)
{
    StreamWriter sw = new StreamWriter(tenfile);
    sw.WriteLine(70);
    sw.WriteLine("abc");
    sw.WriteLine(3.45);
    sw.Close();
}

static void DocFile(string tenfile)
{
    StreamReader sr = new StreamReader(tenfile);
    string str;
    while ((str = sr.ReadLine()) != null)
        Console.WriteLine(str);
    sr.Close();
}
```

```
public static void Main()
{
    string tenfile = @"d:\test.txt";
    TaoFile(tenfile);
    Console.WriteLine("Du lieu doc tu file:");
    DocFile(tenfile);
}
}
```

Kết quả

Du lieu doc tu file:
70
abc
3.45

File Binary

#140

- Ghi: Đối tượng BinaryWriter

Phương thức: Write(giá trị)

- Đọc: Đối tượng BinaryReader

Phương thức:

- ReadByte()
- ReadChar()
- ReadInt32()
- ReadString()
- ReadDouble()

File Binary – Ví dụ

```
static void TaoFile(string tenfile)
```

```
#141
```

```
{
```

```
    FileStream f = new FileStream(tenfile, FileMode.Create,  
                                  FileAccess.Write, FileShare.Write);
```

```
    BinaryWriter bw = new BinaryWriter(f);
```

```
    byte so = 140;
```

```
    string str = "This is a test";
```

```
    float sothuc = 6.542f;
```

```
    bw.Write(so);
```

```
    bw.Write(str);
```

```
    bw.Write(sothuc);
```

```
    f.Close();
```

```
}
```

File Binary – Ví dụ

```
static void DocFile(string tenfile)
{
    #142
    FileStream f = new FileStream(tenfile, FileMode.Open, FileAccess.Read, FileShare.Read);
    BinaryReader br = new BinaryReader(f);
    byte so;           string str;           float sothuc;
    so = br.ReadByte();
    str = br.ReadString();
    sothuc = br.ReadSingle();
    Console.WriteLine("{0}\t{1}\t{2}", so, str, sothuc);
    f.Close();
}

public static void Main()
{
    string tenfile = @"d:\test.bin";
    TaoFile(tenfile);
    Console.WriteLine("Du lieu doc tu file:");
    DocFile(tenfile);
}
```

Kết quả

Du lieu doc tu file:

140 This is a test 6.542

Exception (biệt lệ) – Khái niệm

#143

- Là các lỗi khi chạy chương trình, thông thường xảy ra do người dùng nhập liệu không phù hợp
- Xử lý các biệt lệ này tránh chương trình kết thúc giữa chừng trong quá trình thực thi chương trình

Exception (tt) – Ví dụ

#144

Xét chương trình: Nhập vào số nguyên a, tính căn bậc 2 của a

```
int a;  
double can;  
Console.Write("Nhap vao so a: ");  
a = int.Parse(Console.ReadLine());  
can = Math.Sqrt((double)a);  
Console.WriteLine("Ket qua = " + can);
```

Exception (tt) – Ví dụ

#145

Giả sử người dùng nhập ký tự ‘k’ thay vì nhập số nguyên thì chương trình sẽ thông báo lỗi sau và kết thúc.

Unhandled Exception:

System.FormatException: Input string was not in a correct format.

Một số Exception thường gặp

#146

Exception	Ý nghĩa
FormatException	Sai định dạng dữ liệu
OverflowException	Miền giá trị không phù hợp
OutOfMemoryException	Lỗi không thể cấp phát bộ nhớ
DivideByZeroException	Lỗi chia cho 0
IndexOutOfRangeException	Truy cập phần tử ngoài mảng

Bắt Exception - Mục đích

#147

- Cho phép sửa chữa những lỗi sai trong quá trình nhập
- Cho phép chương trình có thể tiếp tục thực thi đối với những lỗi không quá nghiêm trọng gây ảnh hưởng đến chương trình
- Tạo sự thân thiện cho người dùng: Những thông báo lỗi dễ hiểu

Bắt Exception – Cú pháp

#148

try

{

Các câu lệnh có thể gây ra biệt lệ

}

catch (biệt lệ 1)

{

Các câu lệnh xử lý biệt lệ 1

}

...

catch (biệt lệ n)

{

Các câu lệnh xử lý biệt lệ n

}

Bắt Exception – Ví dụ 1

#149

```
byte k = 0;
```

NHAPLAI:

```
try {
```

```
    Console.Write("Nhap so nguyen duong 1 byte [0..255]: ");
```

```
    k = byte.Parse(Console.ReadLine());
```

```
}
```

```
catch (OverflowException) {
```

```
    Console.WriteLine("Gia tri nhap ngoai mien gia tri, nhap lai!");
```

```
    goto NHAPLAI; }
```

```
catch (FormatException) {
```

```
    Console.WriteLine("Gia tri nhap sai kieu du lieu, nhap lai!");
```

```
    goto NHAPLAI; }
```

```
Console.WriteLine("Da nhap thanh cong, gia tri k = " + k);
```

Bắt Exception – Ví dụ 2

#150

> *Biệt lệ trong catch có thể để trống nếu muốn xử lý lỗi chung chung*

```
byte k = 0;
```

```
NHAPLAI:
```

```
try {
```

```
    Console.Write("Nhap so nguyen duong 1 byte [0..255]: ");
```

```
    k = byte.Parse(Console.ReadLine());
```

```
}
```

```
catch {
```

```
    Console.WriteLine("Nhap khong hop le, nhap lai!");
```

```
    goto NHAPLAI;
```

```
}
```

```
Console.WriteLine("Da nhap thanh cong, gia tri k = " + k);
```

Thread (Tiểu trình) System.Threading

#151

- Cho phép chương trình cùng lúc thực hiện nhiều tác vụ
- Tạo Thread

Thread <tên thread> = new Thread(TênPhươngThức);

- Thực thi Thread

<tên thread>.Start();

Thread – Ví dụ

#152

```
using System.Threading;
```

```
const int stop = 1000;
```

```
static void Ham1()
```

```
{  
    for (int i = 0; i < 10; i++)  
    {  
        Console.Write("1");  
        Thread.Sleep(stop);  
    }  
}
```

```
static void Ham2()
```

```
{  
    for (int i = 0; i < 10; i++)  
    {  
        Console.Write("2");  
        Thread.Sleep(stop);  
    }  
}
```

Thread – Ví dụ

#153

```
static void Main(string[] args)
{
    Thread t1 = new Thread(Ham1);
    Thread t2 = new Thread(Ham2);

    t1.Start();
    t2.Start();
}
```


BÀI TẬP

#154

1. Viết chương trình cho từng dòng chữ rơi từng ký tự xuống phía dưới màn hình
 2. Viết chương trình hiển thị hai dòng chữ:
 - 1 dòng chữ phía trên chạy từ trái sang phải
 - 1 dòng chữ phía dưới chạy từ phải sang trái
- Hai dòng chữ chạy cùng lúc

Chương 2. Lý thuyết về Lập trình hướng đối tượng

TRẦN VIỆT KHÁNH

Email: tranvietkhanh79@gmail.com

Mục tiêu

#156

Trong chương này giới thiệu cho sinh viên một cái nhìn sơ bộ về các phương pháp lập trình khác nhau:

- ☐ Phương pháp lập trình tuyến tính
- ☐ Phương pháp lập trình cấu trúc
- ☐ Phương pháp lập trình hướng đối tượng

Nội dung

#157

1. Lập trình tuyến tính
2. Lập trình cấu trúc
3. Sự trừu tượng hóa dữ liệu
4. Lập trình hướng đối tượng

1. Lập trình tuyến tính

#158

- Xây dựng phần mềm bao gồm rất nhiều công đoạn: phân tích & thiết kế, cài đặt, kiểm tra/thử nghiệm và bảo trì.
- Cài đặt (programming/coding) chỉ là 1 phần trong quá trình trên.

Phương pháp lập trình?

#159

- C++/C#/Java/v.v... là NNLT để viết chương trình.
- PPLT là hệ thống hướng dẫn các giai đoạn cần thiết, cấu trúc của một chương trình.
- PPLT là các cách tiếp cận giúp cho quá trình cài đặt hiệu quả hơn.

Các yêu cầu chính của phần mềm

#160

- Tính tái sử dụng (reusability)
- Tính mở rộng (extensibility)
- Tính mềm dẻo (flexibility)

Phương pháp lập trình tuyến tính

#161

Lập trình tuyến tính

- Chương trình là một dãy các lệnh.
- Lập trình là xây dựng các lệnh trong dãy lệnh.
- Không mang tính thiết kế.

2. Lập trình cấu trúc

#162

Lập trình thủ tục / hàm

- Chương trình là một hệ thống các thủ tục/ hàm. Mỗi thủ tục/ hàm là một dãy các lệnh.
- Lập trình là xác định xem chương trình gồm bao nhiêu thủ tục/ hàm.
- Kết quả là hệ thống cấu trúc và mối quan hệ giữa các hàm/ thủ tục.

2. Lập trình cấu trúc

#163

Lập trình đơn thể

- Chương trình là một hệ thống những đơn thể.
Đơn thể là một hệ thống các thủ tục/ hàm.
- Phân tích và tìm ra các đơn thể.
- Gom nhóm các thành phần tương tự nhau về ý nghĩa, phạm vi...

Bài tập

#164

VD: Xét chương trình nhập vào họ tên, điểm văn, điểm toán của một học sinh và xuất điểm trung bình tương ứng. Hãy viết chương trình trên bằng các phương pháp đã học (lập trình tuyến tính & lập trình cấu trúc).

Cài đặt với pp lập trình tuyến tính (chỉ dùng 1 hàm main & biến toàn cục)

#165

```
static string hoten;  
static int toan, van;  
static float dtb;  
static void Main(string[] args)  
{  
    Console.Write("Nhap ho ten: ");  
    hoten = Console.ReadLine();  
    Console.Write("Nhap diem toan: ");  
    toan = int.Parse(Console.ReadLine());  
    Console.Write("Nhap diem van: ");  
    van = int.Parse(Console.ReadLine());  
    dtb = (float)(toan + van) / 2;  
    Console.WriteLine("Diem trung binh: {0: 0.00}", dtb);  
}
```

Cài đặt với pp lập trình tuyến tính (chỉ dùng 1 hàm main và biến cục bộ)

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

```
#166
```

```
{
```

```
    string hoten;
```

```
    int van, toan;
```

```
    float dtb;
```

```
    Console.Write("Nhap ho ten: ");
```

```
    hoten = Console.ReadLine();
```

```
    Console.Write("Nhap diem toan: ");
```

```
    toan = int.Parse(Console.ReadLine());
```

```
    Console.Write("Nhap diem van: ");
```

```
    van = int.Parse(Console.ReadLine());
```

```
    dtb = (float)(toan + van) / 2;
```

```
    Console.WriteLine("Diem trung binh: {0: 0.00}", dtb);
```

```
}
```

Cài đặt với pp lập trình tuyến tính (chỉ dùng 1 hàm main và cấu trúc toàn cục)

#167

```
struct HOCSINH
{
    public string hoten;
    public int van, toan;
    public float dtb;
}
```

```
static HOCSINH hs;
static void Main(string[] args)
{
    Console.Write("Nhap ho ten: ");
    hs.hoten = Console.ReadLine();
    Console.Write("Nhap diem toan: ");
    hs.toan = int.Parse(Console.ReadLine());
    Console.Write("Nhap diem van: ");
    hs.van = int.Parse(Console.ReadLine());
    hs.dtb = (float)(hs.toan + hs.van) / 2;
    Console.WriteLine("Diem trung binh: {0:0.00}", hs.dtb);
}
```

Cài đặt với pp lập trình tuyến tính (chỉ dùng 1 hàm main và cấu trúc cục bộ)

#168

```
struct HOCSINH
```

```
{
```

```
    public string hoten;
```

```
    public int van, toan;
```

```
    public float dtb;
```

```
}
```

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

```
{
```

```
    HOCSINH hs;
```

```
    Console.Write("Nhap ho ten: ");
```

```
    hs.hoten = Console.ReadLine();
```

```
    Console.Write("Nhap diem toan: ");
```

```
    hs.toan=int.Parse(Console.ReadLine());
```

```
    Console.Write("Nhap diem van: ");
```

```
    hs.van=int.Parse(Console.ReadLine());
```

```
    hs.dtb = (float)(hs.toan + hs.van) / 2;
```

```
    Console.WriteLine("Diem trung binh:  
{0: 0.00}", hs.dtb);
```

```
}
```

Cài đặt với pp lập trình thủ tục (dùng biến toàn cục)

#169

```
static string hoten;  
static int van, toan;  
static float dtb;  
static void Main(string[] args)  
{  
    Nhap();  
    TinhTrungBinh();  
    Xuat();  
}
```

```
static void Nhap()  
{  
    Console.Write("Nhap ho ten: ");  
    hoten = Console.ReadLine();  
    Console.Write("Nhap diem toan: ");  
    toan = int.Parse(Console.ReadLine());  
    Console.Write("Nhap diem van: ");  
    van = int.Parse(Console.ReadLine());  
}
```


Cài đặt với pp lập trình thủ tục (dùng biến toàn cục)

#170

```
static void Xuat()
{
    Console.WriteLine("Diem trung binh: {0: 0.00}", dtb);
}
static void TinhTrungBinh()
{
    dtb = (float)(toan + van) / 2;
}
```

Cài đặt với pp lập trình thủ tục (dùng biến cục bộ)

#171

```
static void Nhap(out string ht, out int v, out int t)
{
    Console.Write("Nhap ho ten: ");
    ht = Console.ReadLine();
    Console.Write("Nhap diem toan: ");
    t = int.Parse(Console.ReadLine());
    Console.Write("Nhap diem van: ");
    v = int.Parse(Console.ReadLine());
}
```

Cài đặt với pp lập trình thủ tục (dùng biến cục bộ)

#172

```
static void Xuat(string hoten, int van, int toan, float dtb)
{
    Console.WriteLine("Diem trung binh: {0: 0.00}", dtb);
}

static float TinhTrungBinh(int van, int toan)
{
    return (float)(toan + van) / 2;
}
```

Cài đặt với pp lập trình thủ tục (dùng biến cục bộ)

#173

```
static void Main(string[] args)
{
    string hoten; int van, toan; float dtb;
    Nhap(out hoten, out van, out toan);
    dtb = TinhTrungBinh(van, toan);
    Xuat(hoten, van, toan, dtb);
}
```

→ Phải quan tâm đến tham số: Trị, chiều và giá trị trả về của mỗi phương thức.

Cài đặt với pp lập trình thủ tục (dùng biến cấu trúc cục bộ)

#174

```
struct HOCSINH
{
    public string hoten;
    public int van, toan;
    public float dtb;
}
static void Main(string[] args)
{
    HOCSINH hs;
    Nhap(out hs);
    Xuat(hs);
}
```

Cài đặt với pp lập trình thủ tục (dùng biến cấu trúc cục bộ)

#175

```
static void Nhap(out HOCSINH hs)
```

```
{
```

```
    Console.Write("Nhap ho ten: ");
```

```
    hs.hoten = Console.ReadLine();
```

```
    Console.Write("Nhap diem toan: ");
```

```
    hs.toan = int.Parse(Console.ReadLine());
```

```
    Console.Write("Nhap diem van: ");
```

```
    hs.van = int.Parse(Console.ReadLine());
```

```
    hs.dtb = (float)(hs.toan + hs.van) / 2;
```

```
}
```

```
static void Xuat(HOCSINH hs)
```

```
{
```

```
    Console.WriteLine("Diem trung binh: {0: 0.00}", hs.dtb);
```

```
}
```

Cài đặt với pp lập trình thủ tục (dùng biến cấu trúc toàn cục)

#176

```
struct HOCSINH
{
    public string hoten;
    public int van, toan;
    public float dtb;
}
static HOCSINH hs;
static void Main(string[] args)
{
    Nhap();
    Xuat();
}
```

```
static void Nhap()
{
    Console.Write("Nhap ho ten: ");
    hs.hoten = Console.ReadLine();
    Console.Write("Nhap diem toan: ");
    hs.toan=int.Parse(Console.ReadLine());
    Console.Write("Nhap diem van: ");
    hs.van = int.Parse(Console.ReadLine());
    hs.dtb = (float)(hs.toan + hs.van) / 2;
}
static void Xuat()
{
    Console.WriteLine("Diem trung binh: {0: 0.00}",
        hs.dtb);
}
```

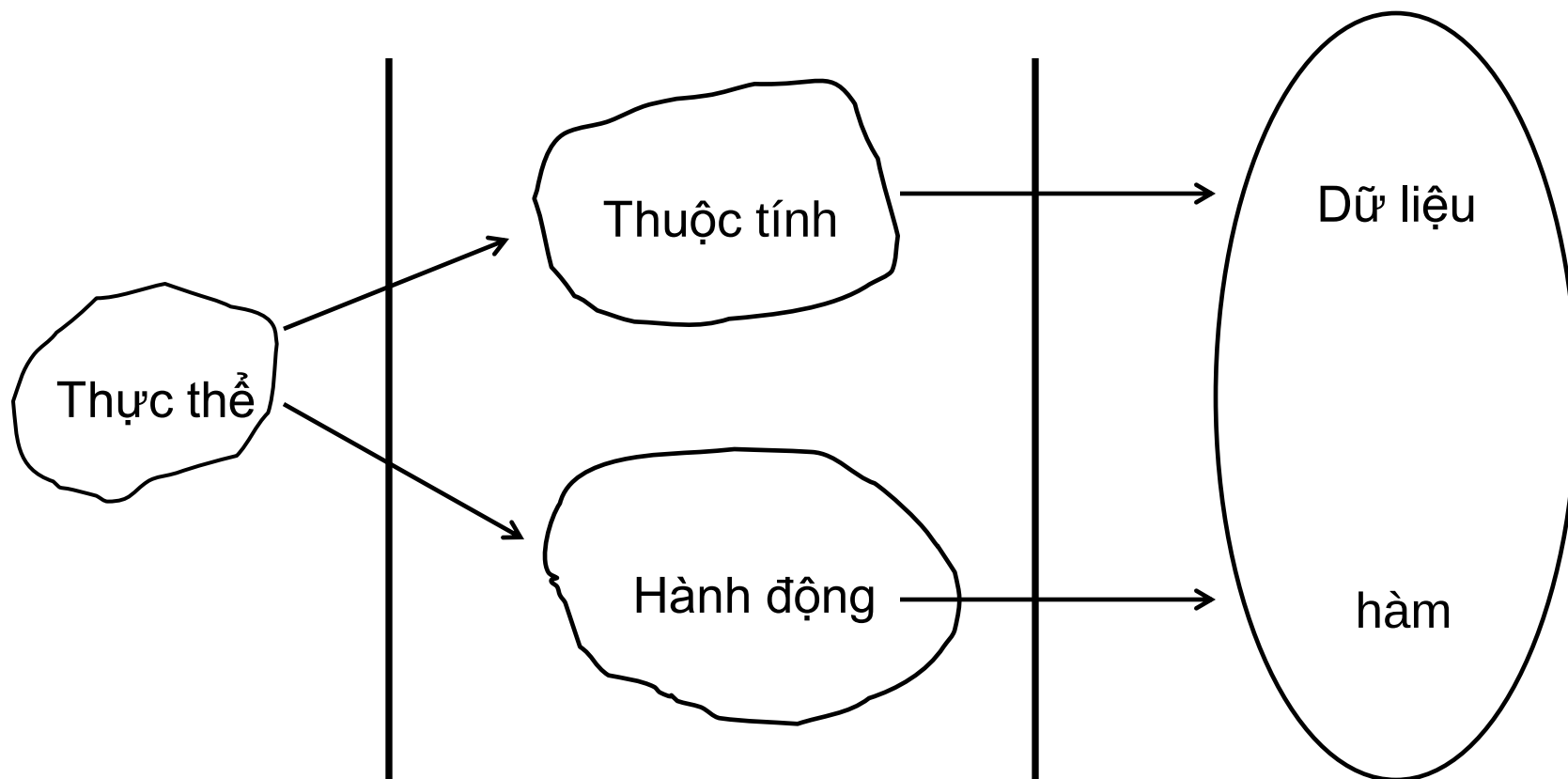
3. Sự trừu tượng hoá dữ liệu

#177

Thế giới thực

Trừu tượng hóa

Phần mềm



Một số khái niệm

#178

Thế giới thực	PPLT	Ngôn ngữ lập trình
Đối tượng trong thế giới thực	Đối tượng	Biến có kiểu cấu trúc
Khái niệm chung về đối tượng	Lớp đối tượng	Kiểu dữ liệu cấu trúc
Các thông tin được quan tâm về 1 đối tượng	Thuộc tính	Thành phần thuộc tính của kiểu cấu trúc
Các khả năng của đối tượng	Hành động	Các phương thức
Phân công giữa các đối tượng	Yêu cầu	Gọi thực hiện phương thức

4. Lập trình hướng đối tượng

#179

- Chương trình là một hệ thống những lớp đối tượng. Mỗi một lớp đối tượng về mặt thực tế tương ứng với những đối tượng có xuất hiện trong thực tế.

PP Lập trình hướng đối tượng

#180

- LT hướng đối tượng là xây dựng những lớp đối tượng và yêu cầu chúng thực hiện những trách nhiệm của mình.
- LT hướng đối tượng là phương pháp LT dựa trên kiến trúc lớp (class) và đối tượng (object)

Đối tượng là gì ?

#181

- Đối tượng trong thế giới thực: là một thực thể cụ thể mà ta có thể sờ, nhìn thấy hay cảm nhận được.
- Đối tượng phần mềm: dùng để biểu diễn các đối tượng trong thế giới thực.
- Mỗi đối tượng bao gồm 2 thành phần: thuộc tính và hành động.

Đối tượng là gì ?

#182

VD: một người A

- Một người có các thuộc tính: tên, tuổi, địa chỉ, màu mắt, ...
- Các hành động: đi, nói, thở, ...

**Một đối tượng là 1 thực thể bao gồm
thuộc tính & hành động**

Lớp đối tượng là gì ?

#183

- Lớp đối tượng thể hiện cho một nhóm các đối tượng giống nhau (cùng thuộc tính & hành động)
- VD: học sinh A, học sinh B, học sinh C...

Thiết kế phần mềm hướng đối tượng

#184

- Trừu tượng hóa dữ liệu và các hàm/ thủ tục liên quan
- Chia hệ thống ra thành các lớp/ đối tượng
- Mỗi lớp/ đối tượng có các tính năng và hành động chuyên biệt
- Các lớp có thể được sử dụng để tạo ra nhiều đối tượng cụ thể

Đặc điểm của pp lập trình HĐT

#185

- **Tính đóng gói (Encapsulation):** Khả năng cất giữ riêng biệt dữ liệu và phương thức tác động lên dữ liệu đó. Do vậy chúng ta không phải quan tâm tới “**phải làm như thế nào**” mà chỉ điều khiển bằng “**làm việc gì**”. Đóng gói giúp đồng nhất giữa dữ liệu và các thao tác tác động lên dữ liệu đó.

Đặc điểm của pp lập trình HĐT

#186

- **Tính thừa kế (inheritance):** Giúp tạo đối tượng mới từ đối tượng có sẵn, bổ sung những đặc tính cần thiết trong đối tượng mới.
- Lớp đối tượng đã có được sử dụng lại gọi là *lớp cơ sở*.
- Lớp thừa kế lớp cơ sở gọi là *lớp dẫn xuất*.

Đặc điểm của pp lập trình HĐT

#187

- **Tính đa hình (polymorphism):** Cho phép gọi cùng một thông điệp đến những đối tượng khác nhau cùng có chung một đặc điểm.

Một số ngôn ngữ lập trình HĐT

#188

- C++
- C#, VB.Net, J#, VC++
- Java
- JavaScript
- PHP
- ...

Các bước thiết kế đối tượng

#189

- Bước 1: Xây dựng sơ đồ đối tượng

- Xác định các lớp đối tượng
- Xác định các quan hệ giữa các lớp

- Bước 2: Thiết kế các lớp

Thiết kế thuộc tính, các hành động

- Bước 3: Cài đặt các lớp

- Bước 4: Sử dụng các lớp để tạo ra các đối tượng

Chương 3: Lớp và đối tượng

TRẦN VIỆT KHÁNH

Email: tranvietkhanh79@gmail.com

Mục tiêu

#191

- ☐ Hiểu được kế thừa trong lập trình hướng đối tượng là gì?
- ☐ Hiểu được khái niệm cây kế thừa.
- ☐ Hiểu được khái niệm sơ đồ lớp

Nội dung

#192

1. Cách cài đặt 1 lớp trong C#
2. Khái niệm hàm khởi tạo (constructors), hàm hủy (destructors)
3. Cách khai thác và sử dụng 1 lớp

1. Cách cài đặt 1 lớp trong C#

#193

- Lớp đối tượng: Định nghĩa các đặc điểm/ thông tin (thuộc tính) và hành động/ chức năng/ (phương thức) chung cho tất cả các đối tượng của cùng một loại.
- Đối tượng: Thể hiện (instance) cụ thể của một lớp đối tượng.

Ví dụ

#194

VD: Lớp SINHVIEN gồm

- Thuộc tính: Họ tên, giới tính, ngày tháng năm sinh, điểm tb, đối tượng ưu tiên, ...
- Phương thức: Học bài, làm bài thi, bài tập, ...

Sinh viên Nguyễn Văn A, Lý Thị B là đối tượng thuộc lớp SINHVIEN

Đối tượng trong LTHĐT

#195

Tách biệt giữa giao tiếp và cài đặt cụ thể

Làm cái gì?

interface

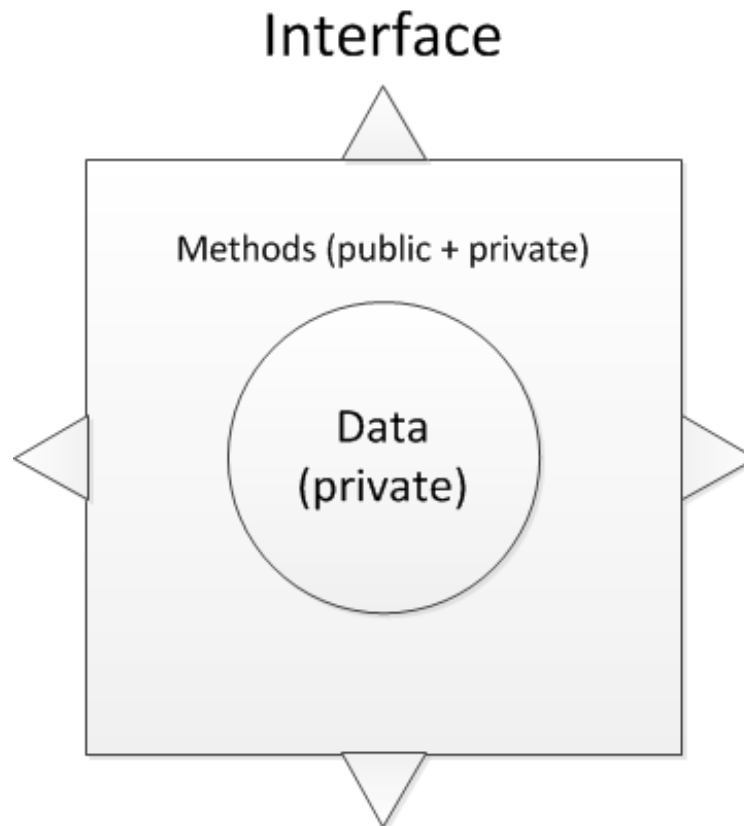
Làm bằng
cách nào?

Implementation

Một cách thể hiện điển hình

#196

Che giấu dữ liệu và các “giải thuật” cụ thể ở bên trong lớp (class)



Cú pháp định nghĩa lớp (class)

#197

```
<từ khóa truy xuất> class <TênLớp>  
{  
    <từ khóa truy xuất> các thuộc tính;  
    <từ khóa truy xuất> phương thức ()  
    {  
        Cài đặt  
    }  
}
```

Từ khóa truy xuất

#198

- `private` (mặc định): Truy xuất trong nội bộ lớp (thường sử dụng cho thuộc tính).
- `protected`: Truy xuất trong nội bộ lớp/ lớp con (được sử dụng cho lớp cơ sở)
- `public`: Truy xuất mọi nơi (thường sử dụng cho phương thức).
- `static`: truy xuất không cần khởi tạo đối tượng của lớp.

VD: định nghĩa lớp CHocSinh

#199

```
public class CHocSinh
{
    private string hoten;
    private int toan, van;
    private float dtb;
    public void Nhap()
    { }
    public void Xuat()
    { }
}
```

Tạo và sử dụng đối tượng

#200

- Tạo đối tượng

<TênLớp> TênĐốiTượng = **new** <TênLớp>();

VD: HOCSINH hsA = new HOCSINH();

- Sử dụng đối tượng

TênĐốiTượng.TênPhươngThức([tham số]);

VD: hsA.Nhap();

hsA.Xuat();

VD: Nhập vào họ tên, điểm văn và điểm toán của 1 học sinh. Tính điểm trung bình và in kết quả

#201

```
public class HOCSINH
{
    private string hoten;
    private int toan, van;
    private float dtb;
    public void Nhap()
    {
        Console.Write("Nhap ho ten: "); hoten = Console.ReadLine();
        Console.Write("Nhap diem van: "); van = int.Parse(Console.ReadLine());
        Console.Write("Nhap diem toan: "); toan = int.Parse(Console.ReadLine());
        dtb = (float)(toan + van) / 2;
    }
    public void Xuat()
    {
        Console.WriteLine("Diem trung binh: {0:0.00}", dtb);
    }
}
```


class Program

{

#202

static void Main(string[] args)

{

HOCSINH hsA = new HOCSINH();

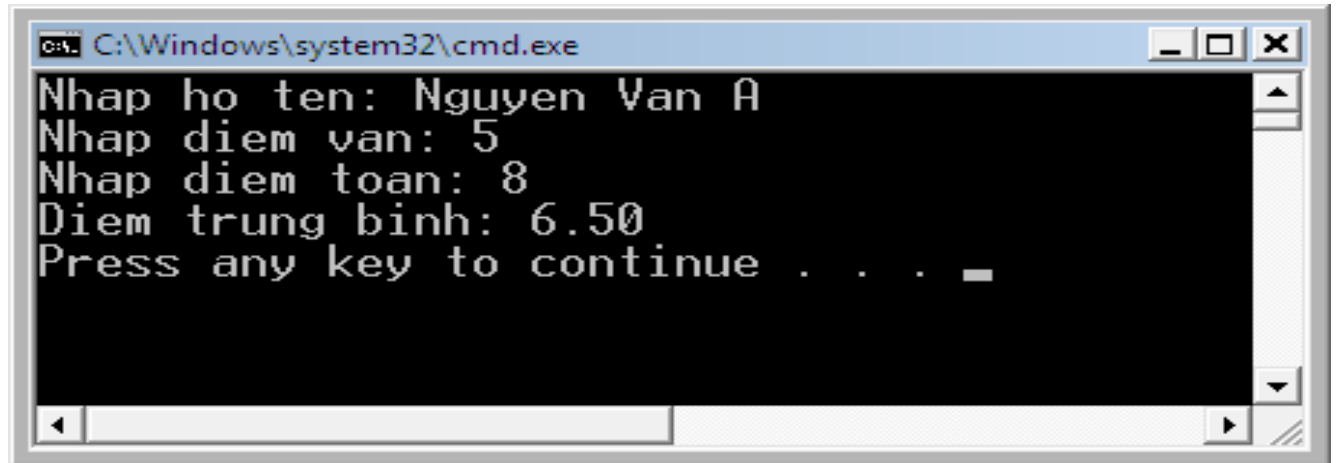
hsA.Nhap();

hsA.Xuat();

}

}

Kết quả:



```
C:\Windows\system32\cmd.exe
Nhap ho ten: Nguyen Van A
Nhap diem van: 5
Nhap diem toan: 8
Diem trung binh: 6.50
Press any key to continue . . . _
```

Chia khai báo lớp thành nhiều file

#203

//File1.cs

namespace PC

{

partial class A

{

int num = 0;

void MethodA()

{

//Cài đặt

}

partial void MethodC();

}

}

//File2.cs

namespace PC

{

partial class A

{

void MethodB()

{

//Cài đặt

}

partial void MethodC()

{

//Cài đặt

}

}

}

Thiết kế thuộc tính

#204

Đối với mỗi đối tượng, xác định các thông tin cần lưu trữ. Sau đó lập bảng mô tả thuộc tính như sau:

Stt	Thuộc tính	Kiểu/ lớp	Ràng buộc	Diễn giải

Nếu có ràng buộc liên thuộc tính thì lập thêm bảng sau:

Stt	Mô tả ràng buộc	Thuộc tính liên quan	Ghi chú

Ràng buộc

#205

Ràng buộc trên lớp là các quy định, quy tắc áp đặt trên các giá trị thuộc tính của đối tượng trong lớp, sao cho đối tượng này thể hiện đúng với thực tế.

Ràng buộc

#206

- Ràng buộc tĩnh: ràng buộc trên giá trị thuộc tính.
- Ràng buộc động: ràng buộc trên biến đổi giá trị thuộc tính.

VD:

- “Lương của nhân viên ít nhất là 1.500.000 đồng” → Ràng buộc tĩnh.
- “Lương của nhân viên chỉ có thể tăng” → Ràng buộc động.

Ràng buộc tĩnh

#207

Gồm 2 loại:

- Ràng buộc trên thuộc tính (Ràng buộc MGT)
- Ràng buộc liên thuộc tính

VD1: Xét lớp điểm ký tự (CDiemKT) trên cửa sổ Console

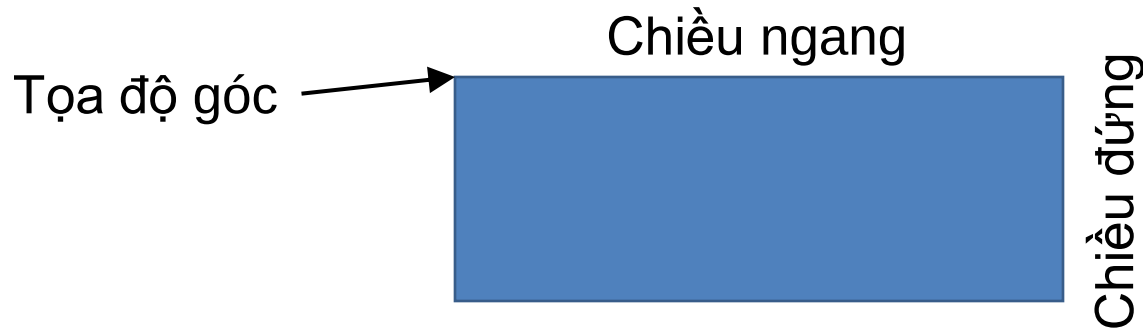
#208

Stt	Thuộc tính	Kiểu/ lớp	Ràng buộc	Diễn giải
1	x	Số nguyên	$0 \leq x < \text{Kích thước ngang}$	Cột
2	y	Số nguyên	$0 \leq y < \text{Kích thước dọc}$	Dòng
3	ch	Ký tự		Ký tự hiển thị

VD2: Xét lớp hình chữ nhật (CHCN) trên cửa sổ Console

Mô tả thuộc tính

#209



Stt	Thuộc tính	Kiểu/ lớp	Ràng buộc	Diễn giải
1	goc	CDiemKT		Toạ độ góc
2	ngang	Số nguyên	$1 < \text{ngang} < \text{Kích thước ngang}$	Chiều ngang
3	dung	Số nguyên	$1 < \text{dung} < \text{Kích thước dọc}$	Chiều đứng

VD2: Xét lớp hình chữ nhật (CHCN) trên cửa sổ Console

Mô tả ràng buộc liên thuộc tính

#210

STT	Mô tả ràng buộc	Thuộc tính liên quan	Ghi chú
1	Tổng của hoành độ góc và m nhỏ hơn kích thước ngang	Goc, m	
2	Tổng của tung độ góc và n nhỏ hơn kích thước dọc	Goc, n	

VD3: Mô tả ràng buộc liên thuộc tính cho lớp CDate

#211

STT	Mô tả ràng buộc	Thuộc tính liên quan	Ghi chú
1	Nếu Th là 4, 6, 9, 11 thì Ng tối đa là 30	Ng, Th	
2	Nếu Th là 2 và Nm nhuận thì Ng tối đa là 29 Nếu Th là 2 và Nm không nhuận thì Ng tối đa là 28	Ng, Th, Nm	

Bài tập

#212

▪ Định nghĩa cấu trúc

```
public struct Sanpham
{
    public string Ma;
    public string Ten;
    public decimal Gia;
    public SanPham(string ma, string ten, decimal gia)
    {
        this.Ma = ma;
        this.Ten = ten;
        this.Gia = gia;
    }
    public string hienThiThongTin()
    {
        return Ma + "\n" + Ten + "\n" + Gia;
    }
}
```

Bài tập

#213

- Khai báo và gán giá trị cho cấu trúc

```
SanPham sp;  
sp.Ma = "sp001";  
sp.Ten = "Dream 100";  
sp.Gia = 20.000.000;
```

```
Console.WriteLine(sp.hienThiThongTin());
```

- Khởi tạo giá trị cho cấu trúc

```
SanPham sp2 = new SanPham("002", "SH", 120.000.000);  
Console.WriteLine(sp2.hienThiThongTin());
```

Bài tập

#214

- Định nghĩa lớp

```
public class Sanpham  
{  
    .....  
}
```

- Tạo đối tượng từ lớp

```
Sanpham dtSanpham;  
dtSanpham = new Sanpham();
```

2. Khái niệm hàm khởi tạo (constructors) – hàm hủy (destructors)

#215

- Khởi tạo không có tham số
public Sanpham()
{
}
- Khởi tạo có tham số
public Sanpham(decimal gia)
{
}

Định nghĩa các thuộc tính và phương thức cho lớp

#216

- Định nghĩa các thuộc tính
private int soluong;
public decimal gia;
- Định nghĩa các phương thức
public void Hienthi()
{
}
public int Hienthi()
{
}

Điều khiển truy cập

#217

Phạm vi	Public	Internal	Protected	private
Cùng lớp	Y	Y	Y	Y
Lớp kế thừa	Y	Y	Y	N
Cùng assembly	Y	Y	N	N
Khác assembly	Y	N	N	N

- Điều khiển truy cập mặc định là private.
- Khi build solution, mọi project trong solution được build thành một assembly(chương trình)

Kế thừa

#218

Lớp cơ sở
(lớp cha)

Sản phẩm	
String	Ma
String	Ten
Decimal	Gia
String	hienThiThongTin()

Lớp dẫn xuất
(lớp con)

Sách	
String	Ma
String	Ten
Decimal	Gia
String	hienThiThongTin()
String	Tacgia

Phần mềm	
String	Ma
String	Ten
Decimal	Gia
String	hienThiThongTin()
String	phienban

Cài đặt lớp cơ sở

#219

- Viết từ khóa virtual trước phương thức có thể sẽ được ghi đè ở các lớp dẫn xuất (lớp con)

```
public class SanPham
{
    public string Ma;
    public string Ten;
    public decimal Gia;
    public SanPham(string ma, string ten, decimal gia)
    {
        this.Ma = ma;
        this.Ten = ten;
        this.Gia = gia;
    }
    public virtual string hienthithongtin()
    {
        return Ma + "\n" + Ten + "\n" + Gia;
    }
}
```

Cài đặt lớp dẫn xuất

#220

- Định nghĩa lớp dẫn xuất

public class TenLopDanXuat : TenLopCoSo

- Định nghĩa phương thức khởi tạo gọi đến phương thức khởi tạo của lớp cơ sở

public class TenLop (danhSachThamSo) : base(danhSachThamSo)

- Gọi đến phương thức hoặc thuộc tính của lớp cơ sở

base.tenPhuongThuc(danhSachThamSo)

base.tenThuocTinh

- Ẩn phương thức non-virtual

public new kieuTraVe tenPhuongThuc

- Ghi đè lên phương thức virtual

public override kieuTraVe tenPhuongThuc

Cài đặt lớp dẫn xuất

#221

```
class Sach : SanPham
{
    public string Tacgia;
    public Sach(string ma, string ten, decimal gia, string tacgia) : base(ma, ten, gia)
    {
        this.Tacgia = tacgia;
    }
    public override string hienThiThongTin()
    {
        return base.hienThiThongTin();
    }
}
```

Cài đặt lớp dẫn xuất

#222

- Cách khác để ghi đè lên một phương thức

```
public override string hienThiThongTin()  
{  
    return base.Ma + "\n" + base.Ten + "\n" + base.Gia + "\n" + Tacgia;  
}
```

```
public override string hienThiThongTin()  
{  
    return base.hienThiThongTin() + "\n" + Tacgia;  
}
```

**Gọi đến phương thức
hienThiThongTin() của
lớp cha**

So sánh LT cấu trúc với LT Hướng Đối Tượng

#223

	Giống nhau	Khác nhau
Lập trình cấu trúc	- Dùng để định nghĩa các biến (kiểu giá trị) - Định nghĩa các hàm/ phương thức.	- Tiết kiệm bộ nhớ hơn. - Không hỗ trợ thuộc tính tự khởi tạo. - Không hỗ trợ phương thức khởi tạo không có tham số. - Các thành viên chỉ được gọi khi được khởi tạo
Lập trình hướng đối tượng	- Dùng để định nghĩa các biến (kiểu đối tượng) - Định nghĩa các hàm/ phương thức.	- Tốn bộ nhớ. - Hỗ trợ thuộc tính tự khởi tạo. - Hỗ trợ phương thức khởi tạo không có tham số. - Hỗ trợ cơ chế ghi đè virtual, override.

Bài tập: thiết kế thuộc tính các lớp

#224

- Lớp thời gian CTime
- Lớp ngày tháng năm CDate
- Lớp phân số CPhanSo
- Lớp CDaThuc (Đa thức 1 ẩn)

$$P_n(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n$$

- Lớp đường thẳng trong mặt phẳng
CDuongThang

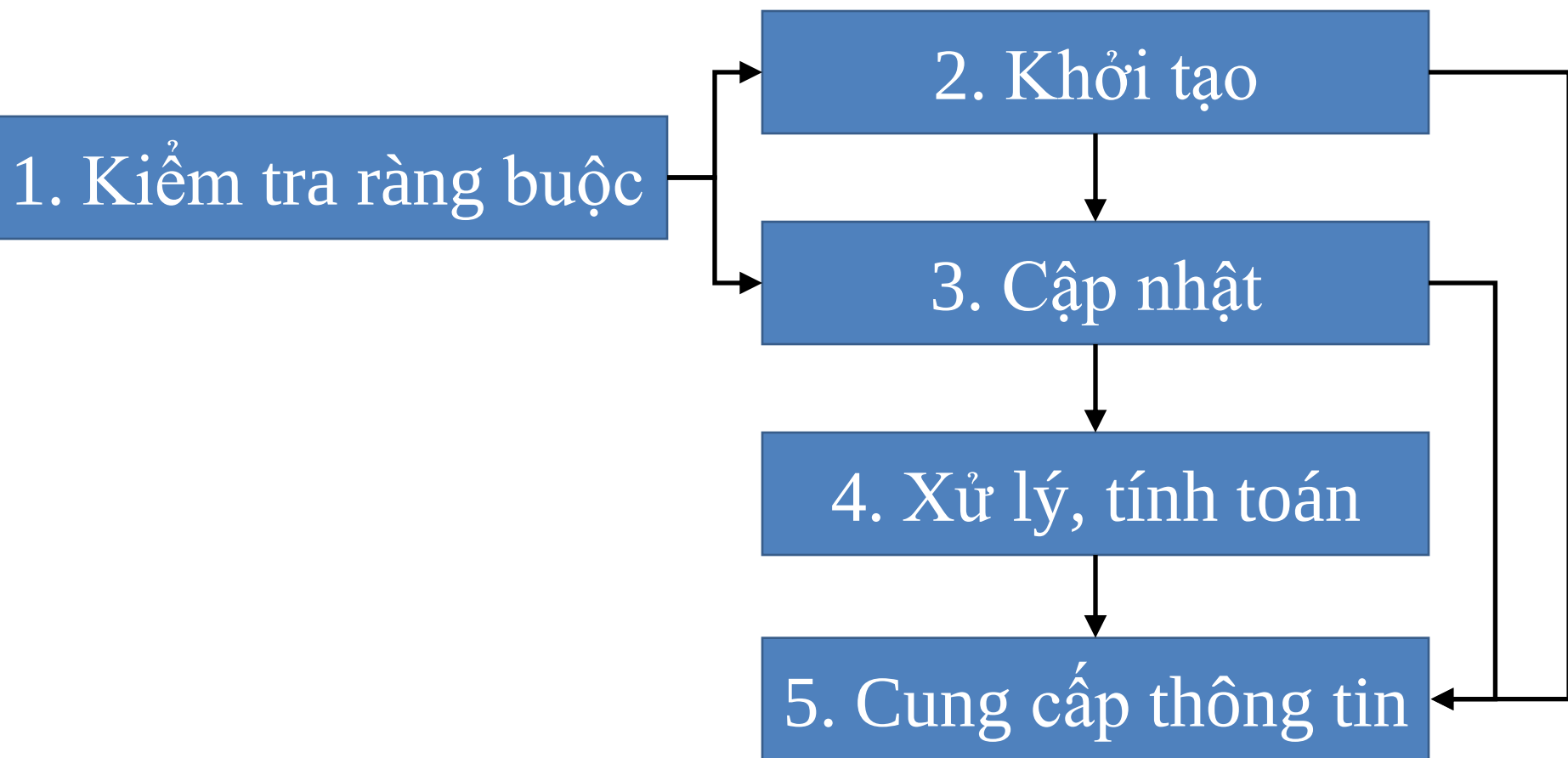
3. Cách khai thác và sử dụng 1 lớp

#225

1. **Nhóm kiểm tra ràng buộc:** Kiểm tra tính hợp lệ giá trị thuộc tính của đối tượng
2. **Nhóm khởi tạo:** Cung cấp giá trị ban đầu cho đối tượng
3. **Nhóm cập nhật:** Thay đổi giá trị thuộc tính của đối tượng
4. **Nhóm xử lý tính toán:** Xử lý tính toán các yêu cầu từ thông tin của đối tượng
5. **Nhóm cung cấp thông tin:** Cung cấp thuộc tính nội bộ của đối tượng

Thiết kế các hành động của lớp

#226



Mẫu thiết kế phương thức ràng buộc

#227

Mẫu: public **bool** KiemTra... (tham số)

- Giá trị trả về
 - **true**: Thoả ràng buộc.
 - **false**: Không thoả ràng buộc.
- Tham số
 - *Ràng buộc miền giá trị*: Chỉ có 1 tham số ứng với tham số cần kiểm tra.
 - *Ràng buộc liên thuộc tính*: Có tham số là các thuộc tính liên quan.

Mẫu thiết kế phương thức ràng buộc

#228

- Tên phương thức

Bắt đầu bằng chữ **KiemTra**

- *Ràng buộc miền giá trị*: Ghép thêm tên thuộc tính
- *Ràng buộc liên thuộc tính*: Ghép thêm số thứ tự ràng buộc

VD thiết kế phương thức kiểm tra ràng buộc cho lớp CHCN

#229

```
class CHCN  
{  
    private CDiem Goc;  
    private int ngang, dung;  
    public bool KiemTraNgang(int ng);  
    public bool KiemTraDung(int d);  
    public bool KiemTra1(int ng, CDiem X);  
    public bool KiemTra2(int d, CDiem Y);  
}
```

Cài đặt phương thức khởi tạo và cập nhật

#230

- Các phương thức thuộc nhóm khởi tạo và cập nhật có liên quan đến ràng buộc phải được bổ sung thêm kiểm tra ràng buộc
- Việc kiểm tra tham số thoả hoặc không thoả ràng buộc bằng cách gọi phương thức kiểm tra ràng buộc tương ứng

public **bool** Tên hàm (Tham số)

{

#231

//Trả về true: thực hiện được

//Trả về false: không thực hiện được

bool kq = false;

if(Tham số **thoả** ràng buộc)

{

 gán giá trị tương ứng cho thuộc tính của lớp

 kq=true;

}

return kq;

}

hoặc

#232

```
public bool Tên hàm ( Tham số )
```

```
{
```

```
//Trả về true: thực hiện được, false: không  
thực hiện được
```

```
if(Tham số không thoả ràng buộc)
```

```
    return false;
```

```
    gán giá trị tương ứng cho thuộc tính của lớp
```

```
    return true;
```

```
}
```

#233

```
public bool CapNhatX(int xx)
{
```

```
    if(!KiemTraX(xx))
```

```
        return false;
```

```
    x=xx;
```

```
    return true;
```

```
}
```

```
public bool CapNhatM(int mm)
```

```
{
```

```
    if(!KiemTra1(mm, Goc))
```

```
        return false;
```

```
    m=mm;
```

```
    return true;
```

```
}
```


VD1: Thiết kế các hành động của lớp CDiemKT

#234

1. Nhóm kiểm tra ràng buộc

```
public bool KiemTraX(int xx);
```

```
public bool KiemTraY(int yy);
```

2. Nhóm khởi tạo

```
public void Nhap();
```

```
public bool KhoiTao (int xx, int yy, char cc);
```

```
public void PhatSinh();
```

VD1: Thiết kế các hành động của lớp CDiemKT

3. Nhóm cập nhật

#235

//Trực tiếp

public bool CapNhatX(int xx);

public bool CapNhatY(int yy);

public void CapNhatCh(char c);

//Gián tiếp

public bool DichPhai(uint k);

public bool DichTrai(uint k);

public bool DichLen(uint k);

public bool DichXuong(uint k);

public bool DichXien1(uint k);

public bool DichXien2(uint k);

VD1: Thiết kế các hành động của lớp CDiemKT

#236

4. Nhóm xử lý tính toán

```
public double KhoangCach(CDiemKT M);  
public int KhoangCachX(CDiemKT M);  
public int KhoangCachY(CDiemKT M);
```

5. Nhóm cung cấp thông tin

```
public void Xuat();  
public void Xoa();  
public int GiaTriX();  
public int GiaTriY();  
public char GiaTriCh();
```

VD2: Thiết kế các hành động của lớp CHCN

#237

1. Nhóm kiểm tra ràng buộc

```
public bool KiemTraM(int mm);
```

```
public bool KiemTraN(int nn);
```

2. Nhóm khởi tạo

```
public void Nhap();
```

```
public bool KhoiTao(CDiemKT M,int cng, int cd);
```

```
public bool KhoiTao(int x, int y, int cng, int cd);
```

```
public void KhoiTao(CDiemKT X, CDiemKT Y);
```

```
public void PhatSinh();
```

VD2: Thiết kế các hành động của lớp CHCN

#238

3. Nhóm cập nhật

//Trực tiếp

```
public bool CapNhatGoc(CDiemKT M);
```

```
public bool CapNhatNgang(int cng);
```

```
public bool CapNhatDung(int cd);
```

VD2: Thiết kế các hành động của lớp CHCN

3. Nhóm cập nhật

#239

//Gián tiếp

public bool DichPhai(int k);

public bool DichTrai(int k);

public bool DichLen(int k);

public bool DichXuong(int k);

public bool TangNgang(int k);

public bool GiamNgang(int k);

public bool TangDung(int k);

public bool GiamDung(int k);

public bool XoayThuan();

public void XoayNghich();

VD2: Thiết kế các hành động của lớp CHCN

#240

4. Nhóm xử lý tính toán

```
public int XetViTri(CDiemKT M);
```

// -1: Bên trong, 0: Trên cạnh, 1: Bên ngoài

```
public int KhoangCachX(CDiemKT M);
```

```
public int KhoangCachY(CDiemKT M);
```

VD2: Thiết kế các hành động của lớp CHCN

#241

5. Nhóm cung cấp thông tin

```
public void Xuat();  
public void Xoa();  
public CDiemKT ToaDoGoc();  
public int ChieuNgang();  
public int ChieuDung();  
public int ChuVi();  
public long DienTich();  
public double DuongCheo();
```


Bài tập

#242

Thiết kế các hành động cho các lớp sau:

- Lớp phân số (CPhanSo)
- Lớp thời gian (CTime)
- Lớp ngày tháng năm (CDate)

Cài đặt phương thức

#243

- Truy xuất và cập nhật dữ liệu (property get-set)
- Trùng tên phương thức (overload)
- Phương thức thiết lập (constructor)
- Cài đặt phép toán (operator)
- Kỹ thuật bổ sung, thay thế chức năng phương thức có sẵn (override)

Truy xuất và cập nhật dữ liệu

#244

```
class CViDu
{
    private int a, b;
    public void Nhap()
    {
    }
    public void Xuat()
    {
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        CViDu vd = new CViDu();
        vd.a = 5; // Lỗi
        vd.b = 4; //Lỗi
        vd.Xuat();
    }
}
```

Mẫu cài đặt property

#245

```
public <kiểu thuộc tính> Tên property
{
    get
    {
        return <tên thuộc tính>;
    }
    set
    {
        if(kiểm tra value thỏa đk)
            <tên thuộc tính> = value;
    }
}
```

Mẫu cài đặt property

#246

- get : Đọc thuộc tính
- set : Gán giá trị cho thuộc tính
- Tên property : Đặt tên bất kỳ theo quy ước nhưng nên đặt dễ nhớ (tốt nhất là trùng tên với tên thuộc tính và ký tự đầu viết hoa)

```
class CViDu
```

```
{
```

```
#247
```

```
private int a, b;
```

```
public int A
```

```
{
```

```
    get{return a;} 
```

```
    set{a=value;} 
```

```
}
```

```
public int B
```

```
{
```

```
    get {return b;} 
```

```
    set { b = value; } 
```

```
}
```

```
}
```

```
class Program
```

```
{
```

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

```
{
```

```
        CViDu vd = new CViDu();
```

```
        vd.A = 5;
```

```
        vd.B = 4;
```

```
}
```

```
}
```

Bài tập

#248

Cài đặt các property cho các lớp sau:

- Lớp phân số (CPhanSo)
- Lớp thời gian (CTime)
- Lớp ngày tháng năm (CDate)

Phương thức trùng tên

#249

Các phương thức có thể có tên trùng nhau ngay cả các phương thức này ở cùng trong 1 lớp nhưng trong số các tham số phải có ít nhất 1 tham số khác

- khác về KDL (nếu cùng số lượng tham số)
- hoặc khác về số lượng tham số

Phương thức trùng tên

#250

```
public void Nhap()
```

```
{ }
```

```
public void Nhap(int aa)
```

```
{ }
```

```
public void Nhap(float aa)
```

```
{ }
```

```
public void Nhap(int aa, float bb)
```

```
{ }
```

```
public int Nhap()
```

Phương thức trùng tên

#251

Phân biệt khi gọi phương thức ?

1. public void PhuongThuc(int a) {}

2. public void PhuongThuc(float a) {}

void Main()

{

int x=7;

float y=6.0;

PhuongThuc(x); //Gọi phương thức số 1

PhuongThuc(y); //Gọi phương thức số 2

}

Phương thức thiết lập

#252

Tự động thực hiện khi đối tượng được sinh ra, các phương thức này có các đặc điểm:

- Không có giá trị trả về.
- Tên phương thức trùng với tên lớp.
- Có thể có hoặc không có tham số.
- Sử dụng phải có từ khóa **new**

Mẫu phương thức thiết lập

#253

```
class <TênLớp>
{
    //Thuộc tính
    //Các phương thức
    public <TênLớp>([tham số])
    {
        Gán giá trị cho thuộc tính
    }
}
```

```
class CViDu
```

```
{
```

```
    private int a, b;
```

```
#254
```

```
    public CViDu()
```

```
    {
```

```
        a = 4;
```

```
        b = 2;
```

```
    }
```

```
    public CViDu(int aa, int bb)
```

```
    {
```

```
        a = aa;
```

```
        b = bb;
```

```
    }
```

```
    public void Xuat()
```

```
    {
```

```
        Console.WriteLine("a = {0}, b = {1}", a, b);
```

```
    }
```

```
}
```

```
class Program
```

```
{
```

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

```
    {
```

```
        CViDu vd = new CViDu();
```

```
        vd.Xuat();
```

```
        vd = new CViDu(1, 23);
```

```
        vd.Xuat();
```

```
    }
```

```
}
```

Sử dụng từ khóa **this**

#255

```
class CViDu
{
    int a, b;
    public void Gan(int a, int b)
    {
        a = a;
        b = b;
    }
    public void Xuat()
    {
        Console.WriteLine("a={0}, b={1}", a, b);
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        CViDu vd = new CViDu();
        vd.Gan(21, 9);
        vd.Xuat();
    }
}
```

Trường hợp tên tham số trùng với tên thuộc tính của đối tượng ta dùng từ khóa **this**. Từ khoá **this** được dùng để tham chiếu đến chính bản thân của đối tượng đó

#256

```
class CViDu
{
    int a, b;
    public void Gan(int a, int b)
    {
        this.a = a;
        this.b = b;
    }
    public void Xuat()
    {
        Console.WriteLine("a={0}, b={1}", a, b);
    }
}
```

Bài tập

#257

Cài đặt các phương thức thiết lập cho các lớp sau:

- Lớp phân số (CPhanSo)
- Lớp thời gian (CTime)
- Lớp ngày tháng năm (CDate)

Cài đặt phép toán

#258

Giả sử lớp phân số (CPhanSo) có phương thức cộng (Cong) và nhân (Nhan) hai phân số. Khi đó, ta muốn cộng 2 phân số a và b lưu vào phân số tổng c, ta phải viết là:

$$c = a.Cong(b);$$

Tương tự cho trường hợp nhân

$$c = a.Nhan(b);$$

Cài đặt phép toán

#259

Nếu muốn viết một cách tự nhiên như sau:

$$\mathbf{c = a + b;}$$

$$\mathbf{c = a * b;}$$

Thì phải cài đặt phương thức thông qua các ký hiệu phép toán (operator)

Mẫu cài đặt phép toán

#260

**public static <KDL> operator ký hiệu
(TênLớp trái, TênLớp phải)**

- Ký hiệu: Gồm các ký hiệu phép toán số học, logic và so sánh
- Trái: Tên tham số sẽ nằm bên trái phép toán
- Phải: Tên tham số sẽ nằm bên phải phép toán
- KDL: bool nếu operator so sánh
TênLớp nếu operator tính toán

VD: Cài đặt phép toán + cho lớp CPhanSo

#261

```
public static CPhanSo operator + (CPhanSo  
ps1, CPhanSo ps2)  
{  
    //Cài đặt  
}
```

Giả sử có 2 phân số a, b và phân số tổng c. Yêu cầu thực hiện như sau:

$$c = a + b;$$

Không dùng operator

class CPhanSo

#262

```
{  
    private int tuso, mauso;  
    public CPhanSo(int t, int m)  
    {  
        tuso = t;  
        mauso = m;  
    }  
    public CPhanSo Cong(CPhanSo ps2)  
    {  
        int tu = tuso*ps2.mauso + ps2.tuso*mauso;  
        int mau = mauso*ps2.mauso;  
        CPhanSo c = new CPhanSo(tu, mau);  
        return c.RutGon();  
    }  
}
```

class Program

#263

```
{  
    static void Main(string[] args)  
    {  
        CPhanSo a = new CPhanSo(3, 5);  
        a.Xuat();  
        CPhanSo b = new CPhanSo(1, 2);  
        b.Xuat();  
        CPhanSo c=new CPhanSo();  
        c = a.Cong(b);  
        Console.WriteLine("Ket qua: ");  
        c.Xuat();  
    }  
}
```

Dùng operator

class CPhanSo

```
{
#264 private int tuso, mauso;
      public CPhanSo(int t, int m)
      {   tuso = t;           mauso = m;
        }
      public static CPhanSo operator +(CPhanSo ps1, CPhanSo ps2)
      {
        int tu = ps1.tuso*ps2.mauso + ps2.tuso*ps1.mauso;
        int mau = ps1.mauso*ps2.mauso;
        CPhanSo c = new CPhanSo(tu, mau);
        return c.RutGon();
      }
      public void Xuat()
      {   Console.WriteLine("{0}/{1}", tuso, mauso);
        }
    }
```

class Program

#265

```
{  
    static void Main(string[] args)  
    {  
        CPhanSo a = new CPhanSo (3, 5);  
        a.Xuat();  
        CPhanSo b = new CPhanSo(1, 2);  
        b.Xuat();  
        CPhanSo c=new CPhanSo();  
        c = a + b;  
        Console.WriteLine("Ket qua: ");  
        c.Xuat();  
    }  
}
```


Bài tập

#266

Cài đặt các operator cho các lớp sau:

- Lớp phân số (CPhanSo)
- Lớp thời gian (CTime)
- Lớp ngày tháng năm (CDate)

Cài đặt bổ sung chức năng phương thức xuất

#267

- Phương thức `Write()` chỉ xuất những dữ liệu có kiểu cơ bản. Ví dụ:

```
int a = 4;
```

```
float b = 7;
```

```
Console.Write("a={0}, b={1}", a, b);
```

- Đối với đối tượng thì phương thức `Write()` không thực hiện được, giả sử có lớp phân số (`CPhanSo`)

```
CPhanSo ps = new CPhanSo(5, 3);
```

```
Console.Write("Phan so: " + ps);
```

Cài đặt bổ sung chức năng phương thức xuất

#268

Để sử dụng được `Console.Write("Phan so: "+ps)` phải cài đặt lại phương thức `ToString()` như sau:

```
public override string ToString()
```

```
{
```

```
    string s=Tạo chuỗi cần xuất;
```

```
    return s;
```

```
}
```

Cài đặt bổ sung chức năng phương thức xuất

#269

```
class CPhanSo
{
    private int tuso, mauso;
    public CPhanSo(int t, int m)
    {
        tuso = t;
        mauso = m;
    }
    public override string ToString()
    {
        string s = tuso.ToString() + "/" + mauso.ToString();
        return s;
    }
}
```

Cài đặt bổ sung chức năng phương thức xuất

#270

```
static void Main(string[] args)
{
    CPhanSo a = new CPhanSo(3, 5);
    Console.Write("Phan so: " + a);
}
```

Bài tập

#271

Cài đặt bổ sung chức năng phương thức xuất
cho các lớp sau:

- CDiemKT
- CHCN
- CDate
- CTime

CHƯƠNG 4: TÁI ĐỊNH NGHĨA (OVERLOADING)

MỤC TIÊU

#273

- ☐ Hiểu được tái định nghĩa và sử dụng lại hàm
- ☐ Hiểu được phương thức toán tử gán là gì?
- ☐ Hiểu được vai trò của toán tử gán trong lập trình hướng đối tượng

4.1 Tái định nghĩa hàm và toán tử

4.2 Trị trả về của hàm là đối tượng, con trỏ
*this

4.3 Cách sử dụng thành phần tĩnh (static)

4.1 Tái định nghĩa hàm và toán tử

#275

1

TOÁN TỬ

2

CHỒNG TOÁN TỬ

3

CÚ PHÁP NẠP CHỒNG TOÁN TỬ

4

TOÁN TỬ MỘT NGÔI

5

TOÁN TỬ HAI NGÔI

6

HỖ TRỢ NGÔN NGỮ .NET KHÁC

7

PHẠM VI SỬ DỤNG CỦA CÁC TOÁN TỬ

8

MỘT SỐ TRƯỜNG HỢP NÊN SỬ DỤNG
TOÁN TỬ NẠP CHỒNG

9

YÊU CẦU KHI SỬ DỤNG CHỒNG TOÁN TỬ

10

ƯU VÀ NHƯỢC ĐIỂM CỦA CHỒNG TOÁN TỬ

TOÁN TỬ

#276

- Toán tử được ký hiệu bằng các biểu tượng đặc biệt và được dùng để thực hiện một hành động nào đó. Trong C# có các toán tử $+$, $-$, $*$, $/$, $\%$, v. v... Thực ra các toán tử này là các hàm.

→ đơn giản là chúng được gọi bằng các cú pháp thay vì gọi hàm.

Ví dụ: $x+7$

- “ $+$ ” là toán tử 2 ngôi với toán hạng là x và 7 .
- Thay vì gọi hàm cộng với 2 toán hạng là x và 7 thì con người thích ký hiệu “ $+$ ” này hơn là gọi hàm cộng.

→ Vì vậy ta hãy nghĩ về nó như là một hàm **$+(x, 7)$**

- “ $+$ ” là tên hàm
- $x, 7$ là 2 đối số truyền vào.
- Hàm “ $+$ ” trả về tổng của các đối số truyền vào.

CHỒNG TOÁN TỬ (Operator Overload)

#277

- Nạp chồng toán tử có nhiều điểm tương tự với nạp chồng hàm, toán tử chính là tên của hàm.
- Việc chồng toán tử thực chất cũng giống như ta xây dựng một hàm thông thường và kết quả thu được như nhau, nhưng chồng toán tử giúp người lập trình có thể sử dụng các ký hiệu toán học thường gặp(+ , - , * , / , v.v...) , dễ nhớ và gần gũi hơn.

CÚ PHÁP NẠP CHỒNG TOÁN TỬ

+ Cú pháp:

#278

Type operator operator_symbol(parameter_list);

Trong đó :

- Type là kiểu giá trị trả về của hàm.
- parameter_list là danh sách các đối số nếu có
- Operator_symbol là các toán tử.

Ví dụ: int operator +(int x,int y);
Phép “+” ở đây đã được nạp chồng.

CÚ PHÁP NẠP CHỒNG TOÁN TỬ

- Trong C# có các hàm toán tử có thể nạp chồng và các phương thức thay thế của nó như sau:

#279

Toán tử	Tên phương thức thay thế	Toán tử	Tên phương thức thay thế
+	Add	>	Compare
-	Subtract	<	Compare
*	Multiply	!=	Compare
/	Divide	>=	Compare
%	Mod	<=	Compare
^	Xor	*=	Multiply
&	BitwiseAnd	- =	Subtract
	Bitwiseor	^=	Xor
&&	Add	<<=	leftshift
	Or	%=	Mod
=	Assign	+=	Add
<<	leftshift	&=	BitwiseAnd
>>	Rightshift	=	Bitwiseor
= =	Equals	/=	Divide

TOÁN TỬ MỘT NGÔI

#280

- Toán tử một ngôi là toán tử chỉ nhận một toán hạng

Ví dụ: $x = -y$; // Đặt x bằng âm y

- Toán tử một ngôi khác:
 $++$, $--$, $+(+a)$, $-(-a)$, $++(++a)$...

TOÁN TỬ HAI NGÔI

#281

- Các toán tử như toán tử (`=`) so sánh giữa hai đối tượng, toán tử (`!=`) so sánh không bằng giữa hai đối tượng, `<` so sánh nhỏ hơn, `>` so sánh lớn hơn... Đây là các toán tử phải có các cặp toán hạng hay ta còn gọi là toán tử hai ngôi.

Ngoài ra, trong C# còn có toán tử ba ngôi.

Ví dụ:

```
int value1,value2, maxValue;  
value1=10; value2=20;  
maxValue=value1>value2 ? value1:value2;
```

- Trong ví dụ trên toán tử ba ngôi được sử dụng để kiểm tra xem giá trị của `value1` có lớn hơn giá trị của `value2` không, nếu đúng thì trả về giá trị của `value1`, tức gán giá trị `value1` cho biến `maxValue`, còn ngược lại thì gán giá trị `value2` cho biến `maxValue`.

CODE MINH HỌA NẠP CHỒNG TOÁN TỬ

#282

```
using System;
namespace VietKhanh
{
    class Box
    {
        private double chieu_dai;
        private double chieu_rong;
        private double chieu_cao;

        public double tinhTheTich()
        {
            return chieu_dai * chieu_rong * chieu_cao;
        }

        public void setChieuDai(double len)
        {
            chieu_dai = len;
        }
    }
}
```

CODE MINH HỌA NẠP CHỒNG TOÁN TỬ

#283

```
    public void setChieuRong(double wid)
    {
        chieu_rong = wid;
    }

    public void setChieuCao(double hei)
    {
        chieu_cao = hei;
    }

    // nạp chồng toán tử "+" để cộng hai đối tượng Box.
    public static Box operator +(Box b, Box c)
    {
        Box box = new Box();
        box.chieu_dai = b.chieu_dai + c.chieu_dai;
        box.chieu_rong = b.chieu_rong + c.chieu_rong;
        box.chieu_cao = b.chieu_cao + c.chieu_cao;
        return box;
    }
}
```

CODE MINH HỌA NẠP CHỒNG TOÁN TỬ

#284

■ Code xử lý trên Form.

//tao cac doi tuong Box1, Box2 va Box3

- Box Box1 = new Box();
- Box Box2 = new Box();
- Box Box3 = new Box();
- double the_tich = 0.0;
-
- // nhap thong tin Box1
- Box1.setChieuDai(6.0);
- Box1.setChieuRong(7.0);
- Box1.setChieuCao(5.0);
-
- // nhap thong tin Box2
- Box2.setChieuDai(12.0);
- Box2.setChieuRong(13.0);
- Box2.setChieuCao(10.0);

CODE MINH HỌA NẠP CHỒNG TOÁN TỬ

#285

-
- // tính và hiện thị the tích Box1
- the_tich = Box1.tinhTheTich();
- lblTheTichHop1.Text = the_tich.ToString();
-
- // tính và hiện thị the tích Box2
- the_tich = Box2.tinhTheTich();
- lblTheTichHop2.Text = the_tich.ToString();
-
- // cộng hai đối tượng Box1 và Box2 thông qua nạp
chồng toán tử operator +(Box b, Box c)
- Box3 = Box1 + Box2;
-
- // tính và hiện thị the tích Box3
- the_tich = Box3.tinhTheTich();

Nạp chồng phương thức (Function Overload)

#286

- C# cho phép nhiều hàm (phương thức) trong cùng một phạm vi (namespace, class) có thể trùng tên, nhưng phải khác nhau về các tham số khi gọi. Điều này được gọi là nạp chồng phương thức hay còn gọi là định nghĩa chồng.
- Ví dụ:

```
class C {  
    public:  
        int compare(int x, int y);  
        int compare(int x, int y) const;  
        int compare(float x, float y);  
};
```

Nạp chồng phương thức (Function Overload)

+ Định nghĩa chồng ở lớp con sẽ che mất các phương thức cùng tên của lớp cha.

#287

```
class D: C { // lớp con D kế thừa lớp cha C (học thêm ở bài sau)
```

```
    public:
```

```
        // Định nghĩa trùng tên với lớp cha
```

```
        int compare(string s1, string s2);
```

```
};
```

```
    // Sử dụng lớp con D
```

```
    D d = new D();
```

```
    d.compare("1234", "abcd"); // OK
```

```
    d.compare(10, 20); // lỗi
```

```
    // Sử dụng lớp cha C
```

```
    d.C::compare(10, 20); // OK
```

Bài tập áp dụng

#288

- Xây dựng class Phân số và thực hiện các phép tính cộng, trừ, nhân, chia, hai phân số, thực hiện rút gọn phân số.
- Các bước thực hiện như sau:

Bước 1: Thiết kế class phân số

class PHANSO

#289

```
{
    int tuso, mauso;
    public PHANSO()
    {
    }
    public PHANSO(int t, int m)
    {
        tuso = t;
        mauso = m;
    }
    int UCLN(int a, int b)
    {
        while(a!=b)
        {
            if (a > b)
                a = a - b;
            else
                b = b - a;
        }
        return a;
    }
}
```


Thiết kế class phân số (tt)

private void RutGon(int tso,int mso)

#290 {

int x = UCLN(tso, mso);

tuso = tso / x;

mauso = mso / x;

}

public void Cong(PHANSO ps1,PHANSO ps2)

{

tuso = ps1.tuso*ps2.mauso + ps2.tuso*ps1.mauso;

mauso = ps1.mauso*ps2.mauso;

RutGon(tuso,mauso);

}

public void Tru(PHANSO ps1,PHANSO ps2)

{

tuso = ps1.tuso * ps2.mauso - ps2.tuso * ps1.mauso;

mauso = ps1.mauso * ps2.mauso;

RutGon(tuso, mauso);

}

Thiết kế class phân số (tt)

#291

```
public void Nhan(PHANSO ps1, PHANSO ps2)
{
    tuso = ps1.tuso * ps2.tuso;
    mauso = ps1.mauso * ps2.mauso;
    RutGon(tuso, mauso);
}
public void Chia(PHANSO ps1, PHANSO ps2)
{
    tuso = ps1.tuso * ps2.mauso ;
    mauso = ps1.mauso * ps2.tuso;
    RutGon(tuso, mauso);
}
public void  NhapPhanSo(string strtuso,string strmauso)
{
    tuso = int.Parse(strtuso);
    mauso = int.Parse(strmauso);
}
public string Xuat()
{
    return tuso + "/" + mauso;
}
}
```

Bước 2: Viết code cho các chức năng cộng, trừ nhân, chia.

#292

```
Void main()
```

```
{
```

```
    PHANSO a = new PHANSO(tuso1, mauso1);
```

```
    PHANSO b = new PHANSO(tuso2, mauso2);
```

```
    PHANSO c = new PHANSO();
```

```
    c.Cong(a,b);
```

```
    lblKetQua.Text = c.Xuat();
```

```
}
```

Bộ test mẫu:

Test	Phân số 1	Phân số 2	Cộng	Trừ	Nhân	Chia
test 1	3/4	2/3	17/12	1/12	1/2	9/8
test 2	7/2	5/4	19/4	9/4	35/8	14/5

DEMO

HỖ TRỢ NGÔN NGỮ .NET KHÁC

#294

C# cung cấp khả năng cho phép nạp chồng toán tử cho các lớp mà chúng ta xây dựng. Trong khi đó các ngôn ngữ .NET khác không cho phép điều này.

+ Vì vậy ta phải xây dựng các hàm (phương thức) thay thế cho các phép toán có sử dụng nạp chồng toán tử để các ngôn ngữ khác có thể gọi và sử dụng được. Để làm được điều này khi sử dụng nạp chồng toán tử nào đó thì phải xây dựng cho nó một phương thức có chức năng tương tự như toán tử đã chồng.

Ví dụ: Khi chúng ta nạp chồng toán tử (+) thì phải cung cấp một phương thức `Add( )` cũng làm chức năng cộng hai đối tượng. Khi sử dụng chồng toán tử (`=`) thì nên cung cấp thêm phương thức `Equals()`, v.v...

PHẠM VI SỬ DỤNG CỦA CÁC TOÁN TỬ

#295

Phạm trù	Toán tử	Hạn chế
Nhị phân toán học	+, *, /, -, %	Không
Thập phân toán học	+, -, ++, --	Không
Nhị phân bit	&, , ^, <<, >>	Không
Thập phân bit	!, ~, true, false	Không
So sánh	==, !=, >=, <, <=, >	Phải nạp chồng theo từng cặp.

MỘT SỐ TRƯỜNG HỢP SỬ DỤNG TOÁN TỬ NẠP CHỒNG

#296

1. Trong toán học, mọi đối tượng toán học như: tọa độ, vector, ma trận, hàm số v...v... Nếu viết chương trình làm những mô hình toán học hay vật lý, bạn nhất định sẽ mô tả những đối tượng này.

2. Những chương trình đồ họa sẽ sử dụng các đối tượng toán học và tọa độ khi tính toán vị trí của nó trên màn hình.

3. Một lớp mô tả số lượng tiền.

4. Việc sử lý từ hay chương trình phân tích văn bản có lớp để mô tả các câu văn, mệnh đề và bạn phải sử dụng các toán hạng để liên kết các câu lại với nhau.

YÊU CẦU KHI SỬ DỤNG CHỒNG TOÁN TỬ

#297

+ Khi định nghĩa những toán tử trong kiểu dữ liệu giá trị, kiểu dữ liệu xây dựng sẵn.

+ Phải cung cấp các phương thức thay thế cho toán tử được nạp chồng. Bởi vì hầu hết các ngôn ngữ khác không cho phép chồng toán tử nên khi ta sử dụng thêm phương thức có chức năng tương tự như toán tử giúp cho chương trình có thể phù hợp với nhiều ngôn ngữ khác nhau.

+ Sử dụng chồng toán tử trong trường hợp kết quả trả về rõ ràng.

Ví dụ: Khi ta sử dụng toán tử “or” hoặc “and” giữa một giá trị thời gian này với một thời gian khác thì kết quả trả về sẽ không rõ ràng vì vậy trong trường hợp này ta không nên sử dụng chồng toán tử.

ƯU VÀ NHƯỢC ĐIỂM CỦA CHỒNG TOÁN TỬ

#298

Ưu điểm :

- + Nạp chồng toán tử là một phương thức ngắn gọn giúp mã nguồn dễ nhìn hơn, dễ quản lý, sáng sủa hơn.
- + Nạp chồng toán tử là đường dẫn cho những đối tượng.

Nhược điểm:

- + Ta phải sử dụng nạp chồng toán tử một cách hạn chế và phù hợp với các toán tử của lớp được xây dựng sẵn, không sử dụng toán tử quá mới hay quá riêng rẽ. Nếu sử dụng toán tử một cách lạm dụng thì sẽ làm chương trình dễ nhầm lẫn.

4.2 Trị trả về của hàm là đối tượng, con trỏ *this

#299

Cú pháp con trỏ:

+ Kiểu dữ liệu: * Tên con trỏ;

Cách dùng con trỏ:

+ * Tên con trỏ = giá trị (nội dung).

Ví dụ:

```
int x=10;  
int *px;  
px = &x;
```

4.2 Trị trả về của hàm là đối tượng, con trỏ *this

#300

Con trỏ `this` trở vào dữ liệu thành viên của chính nó. Điều này có nghĩa là con trỏ `this` chỉ có phạm vi tác dụng trong một lớp. Một điều cực kì quan trọng, là con trỏ `this` chỉ hoạt động với các dữ liệu thành viên và các hàm thành viên được khai báo là không tĩnh (non-static). Các dữ liệu thành viên và hàm thành viên tĩnh (static) không hỗ trợ con trỏ `this`.

Ví dụ

4.2 Trị trả về của hàm là đối tượng, con trỏ *this

#301

```
class Complex{
    float real;
    float img;
public   Complex(float real, float img)
{
    this->real = real;
    this->img = img;
}
bool isMe(const Complex &c)
{
    if(&c==this) return true;
    else return false;
}
public static void main(){
    Complex a = Complex(3, 2);
    Complex b=Complex b(2, 2);
    Complex *c = &a;
    Console.WriteLine( a.isMe(b));// cho giá trị false
    Console.WriteLine( a.isMe(*c));// cho giá trị true
}
```

4.2 Trị trả về của hàm là đối tượng, con trỏ *this

#302

Với việc sử dụng con trỏ this trong hàm tạo, bạn có thể đặt tên các tham số trong hàm tạo trùng với tên các dữ liệu của lớp. Để truy cập đến các thuộc tính của lớp, ta sử dụng con trỏ this. Hàm thành viên isMe sẽ kiểm tra một đối tượng có phải là chính nó hay không (có cùng địa chỉ trên bộ nhớ). Dù là một bản sao của nó (có dữ liệu thành viên giống nhau) thì kết quả nhận được cũng là sai (0). Trong hàm main, ta khởi tạo hai đối tượng a và b. Đối tượng con trỏ c sẽ trỏ vào địa chỉ của đối tượng a. Điều này có nghĩa là c sẽ có cùng vùng địa chỉ với a, còn b thì không. Khi gọi hàm a.isMe(b) sẽ cho kết quả là sai (0) và a.isMe(*c) sẽ cho kết quả là đúng (1).

4.3 Cách sử dụng thành phần tĩnh (static)

#303

Xét ví dụ class TinhToan có phương thức

```
Class TinhToan{  
    public int Cong(int a, int b); }
```

Nếu bạn muốn xài phương thức này thì phải có 1 object TinhToan, ví dụ TinhToan tt = new TinhToan(); rồi gọi tt.Cong(1, 2); mới xài được.

Ở đây tạo object hơi thừa vì với bất kì object nào của class TinhToan thì hàm Cong() cũng như nhau cả, vậy cần tạo 1 object để làm gì. Chỉ cần thêm từ khóa “static” vào public static int Cong(int a, int b); thì không cần tạo object làm gì nữa, gọi TinhToan.Cong(1, 2); là được.

Tóm lại khi sử dụng từ khóa static thì không cần khởi tạo đối tượng (object)

CHƯƠNG 5: HÀM BẠN (FRIEND)

MỤC TIÊU

#305

- Hiểu được khái niệm về hàm friend và cách khai báo nó.
- Sử dụng hàm friend để truy cập các thuộc tính, các phương thức của lớp cơ sở có khai báo là private và protected

NỘI DUNG

#306

1

Hàm thành viên là bạn của 1 lớp

2

Hàm độc lập là bạn của 1 lớp

3

Lớp bạn

5.1 Hàm thành viên là bạn của 1 lớp

#307

Friend của 1 class có thể là thành viên của class khác hoặc tất cả các class trong cùng 1 chương trình. Như là 1 GLOBAL FRIEND

Friend có thể access private hoặc protected của class được khai báo là Friend.

Friends không phải là một thành viên vì vậy không có con trỏ "this".

Friend có thể khai báo ở bất cứ đâu (public, protected or private section) trong một class.

Khai báo và sử dụng

#308

Để khai báo một hàm dạng hàm friend của một lớp, đặt trước nguyên mẫu hàm đó trong định nghĩa lớp với từ khóa friend trong C#, như sau:

```
class Person
{
    private:
        string name;
    public:
        friend void DisplayName( Person person); //hàm hiển thị tên
        void setName( string name );
};
```

Để khai báo tất cả hàm thành viên của lớp Employee là dạng friend của lớp Person, đặt một khai báo sau trong định nghĩa của lớp Person:

```
class Person
{
    friend class Employee; // Employee là friend của person
};
```

5.2 Hàm độc lập là bạn của 1 lớp

#309

Trong lớp dẫn xuất không cho phép truy nhập đến các thuộc tính private của lớp cơ sở.

Vậy nếu bạn muốn tạo quan hệ giữa 2 class A và B. Sao cho B có thể truy cập các private hay protected của A nhưng bạn không muốn xây dựng các phương thức set hoặc get để tránh các class khác ngoài B cũng truy cập được thì làm thế nào?. Đó là truy cập một hàm độc lập có chọn lọc và từ khóa **Friend** sẽ giúp chúng ta làm điều này.

Ví dụ

#310

```
class Person
{
private:
    string name;

public:
    //Hàm DisplayName không phải thành viên của person mà là Friend. Có thể gọi ở bất cứ đâu trong chương trình
    friend void DisplayName( Person person);
    void setName( string name );

    // phân định nghĩa hàm setname
    void Person::setName( string name )
    {
        name = name;
    }
    // Ghi chú: DisplayName() không là hàm thành viên của bất cứ class nào.
    void DisplayName( Person person )
    {
        /* Bởi vì, hàm DisplayName() là hàm friend của Person, do vậy nó có thể
        trực tiếp truy cập bất cứ thành viên nào của class này */
        Console.WriteLine("Tên của bạn là: " +person.name);
    }
};

// hàm main của chương trình
int main( )
{
    Person person;

    // set tên cho person không sử dụng hàm thành viên
    person.setName("Đỗ Trung Quân");

    // sử dụng hàm friend để in ra tên.
    DisplayName( person);

    return 0;
}
```

Định nghĩa lớp dẫn xuất

#311

- Một lớp được xây dựng thừa kế một lớp khác được gọi là lớp dẫn xuất (lớp con); lớp được dùng để xây dựng lớp dẫn xuất được gọi là lớp cơ sở (lớp cha).
- Một lớp dẫn xuất ngoài các thành phần riêng của lớp đó, còn được thừa kế các thành phần của các lớp cơ sở có liên quan

Cách xây dựng lớp dẫn xuất

#312

- Giả sử ta đã xây dựng lớp A, B. Để xây dựng lớp C kế thừa public các lớp A và B, ta viết theo mẫu sau:

```
class C: public A, public B
{
    private:
        //Cac thuoc tinh
    public:
        //Cac phuong thuc
};
```

Tương tự, ta thay từ khóa public bằng private để C kế thừa private từ A và B

Cách xây dựng lớp dẫn xuất

#313

Tính chất:

Kế thừa theo kiểu public thì tất cả các thành phần public của lớp cơ sở sẽ là thành phần public của lớp dẫn xuất. Kế thừa theo kiểu private thì tất cả các thành phần public của lớp cơ sở sẽ là thành phần private của lớp dẫn xuất.

Thừa kế các thuộc tính

Các thuộc tính của lớp cơ sở sẽ được thừa kế trong lớp dẫn xuất. Như vậy thì tập thuộc tính của lớp dẫn xuất sẽ gồm: các thuộc tính mới khai báo trong định nghĩa lớp dẫn xuất và các thuộc tính của lớp cơ sở. Tuy nhiên, trong lớp dẫn xuất không cho phép truy nhập đến các thuộc tính private của lớp cơ sở.

5.3 Lớp Bạn (Friend)

#314

Friend được xây dựng để khắc phục điểm yếu **lớp dẫn xuất không thể truy cập tới các biến private của lớp cơ sở**.

Một friend có thể là một hàm, một mẫu hàm, hoặc hàm thành viên, hoặc một lớp hoặc một mẫu lớp, trong trường hợp này, toàn bộ lớp và tất cả thành viên của nó là friend. Hàm friend trong C++/C# của một lớp được định nghĩa bên ngoài phạm vi lớp đó, nhưng nó có quyền truy cập tất cả thành viên private và protected của lớp đó. Ngay cả khi các nguyên mẫu cho hàm friend xuất hiện trong định nghĩa lớp, thì các hàm friend không là các hàm thành viên.

Chương 6. Kế thừa và đa hình

TRẦN VIỆT KHÁNH

Email: tranvietkhanh79@gmail.com

MỤC TIÊU

#316

- Hiểu được các khái niệm đa hình, kế thừa, cách sử dụng các từ khóa virtual, override, abstract.
- Sử dụng tính kế thừa và tính đa hình trong việc xây dựng một chương trình phần mềm.

Nội dung

#317

- Đóng gói
- Đa hình
- Kế thừa
- Virtual, Override, Abstract trong lập trình hướng đối tượng

6.1 Đóng gói

#318

- **Encapsulation (Tính đóng gói)** được định nghĩa là "tiến trình đóng gói một hoặc nhiều mục bên trong một gói logic hoặc vật lý". Tính đóng gói, trong phương pháp lập trình hướng đối tượng, ngăn cản việc truy cập tới chi tiết của trình trình triển khai (Implementation Detail).
- Tính trừu tượng và tính đóng gói là hai đặc điểm có liên quan với nhau trong lập trình hướng đối tượng. Tính trừu tượng cho phép tạo các thông tin liên quan có thể nhìn thấy và tính đóng gói cho lập trình viên khả năng *triển khai độ trừu tượng đã được kế thừa*.

6.1 Đóng gói

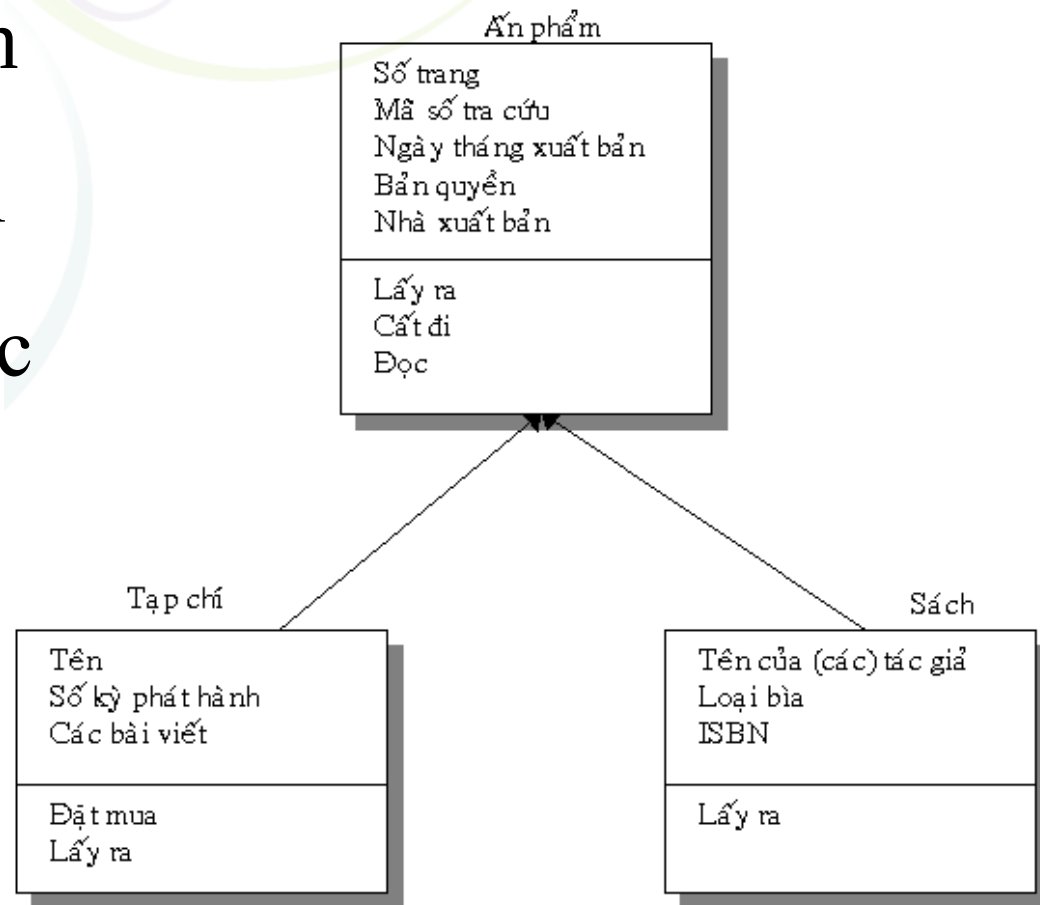
#319

- Tính đóng gói được triển khai bởi sử dụng **Access Specifier**. Một Access Specifier định nghĩa phạm vi và tính nhìn thấy của một thành viên lớp. C++/C# hỗ trợ các Access Specifier sau:
 - Public
 - Private
 - Protected
 - Internal
 - Protected internal

6.2 Tính đa hình

#320

Làm thế nào lưu danh sách (mảng) 2 loại ấn phẩm cùng lúc & thực thi đúng hành động “LayRa” của loại ấn phẩm đó ?



Khái niệm tính đa hình

#321

- Tính đa hình là khả năng để cho một thông điệp có thể thực hiện bằng nhiều cách khác nhau tùy thuộc vào đối tượng cụ thể nhận thông điệp.
- Khi một lớp dẫn xuất được tạo ra, nó có thể thay đổi cách thực hiện các phương thức nào đó mà nó thừa hưởng từ lớp cơ sở.

6.3 Kế thừa

#322

Giả sử đã xây dựng lớp CDate hoàn chỉnh

→ Cần xây dựng ứng dụng tính tiền lãi của một ngân hàng thành lập ngày 14/3/1997

→ Cần xây dựng ứng dụng quản lý sinh viên có thuộc tính ngày tháng năm sinh (sinh viên phải từ 17 tuổi trở lên)

Đặt vấn đề

#323

Cách 1: Sửa lại lớp CDate cho phù hợp với các yêu cầu của lớp CDate trong ứng dụng trên
→ Sửa lại hàm kiểm tra → Ảnh hưởng đến các chương trình khác có sử dụng lớp CDate ở dạng tổng quát.

Đặt vấn đề

#324

Cách 2: Xây dựng lớp CDate mới độc lập với lớp CDate → Tốn nhiều công sức.

Cách 3: Sao chép lớp CDate để tạo lớp CDate mới và sau đó sửa lại theo yêu cầu của chương trình → Khó khăn do thực hiện thủ công khi mở rộng, cập nhật, ...

Đặt vấn đề

#325

➔ Cần có cơ chế cho phép khai báo lớp CDate mới là lớp CDate cũ với 1 số các sửa đổi bổ sung.

Đặt vấn đề

#326

Tương tự cho chương trình đánh caro, cờ tướng trên máy tính. Mỗi quân cờ được xem như 1 điểm ký tự (CDiemKT) nhưng mỗi quân cờ có những đặc điểm khác nhau. Do vậy cần sử dụng lớp CDiemKT bổ sung và sửa đổi một số phần chứ không phải tốn công sức để xây dựng lại từ đầu.

Khái niệm

#327

Kế thừa cho phép khai báo 1 lớp B là 1 lớp dẫn xuất từ lớp A. Khi đó B sẽ có tất cả các thuộc tính và đặc điểm của A, ngoài ra B có thể có thêm những thuộc tính và những hành động mới.

Khái niệm

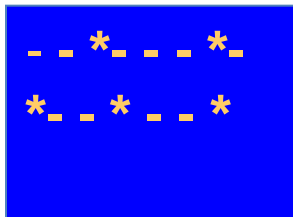
#328

- Kế thừa thể hiện khả năng tái sử dụng các lớp đã được định nghĩa.
- Có thể định nghĩa lớp đối tượng mới dựa trên 1 hay nhiều lớp đối tượng đã có sẵn.
- Lớp có sẵn được gọi là lớp cơ sở (based class) và lớp kế thừa được gọi là lớp dẫn xuất (derived class)

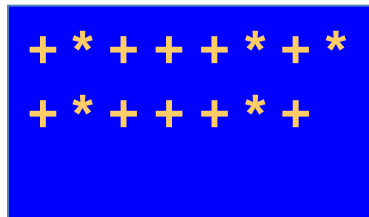
Khái niệm

#329

A

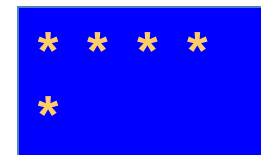


B

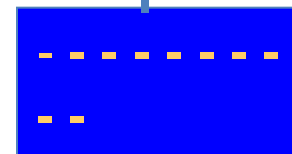


* tính chất chung
- tính chất của A
+ tính chất của B

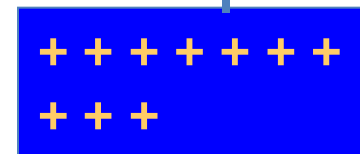
C



A

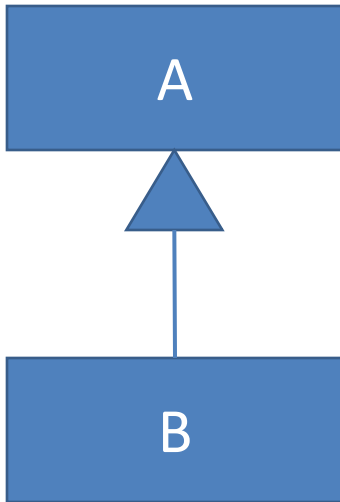


B

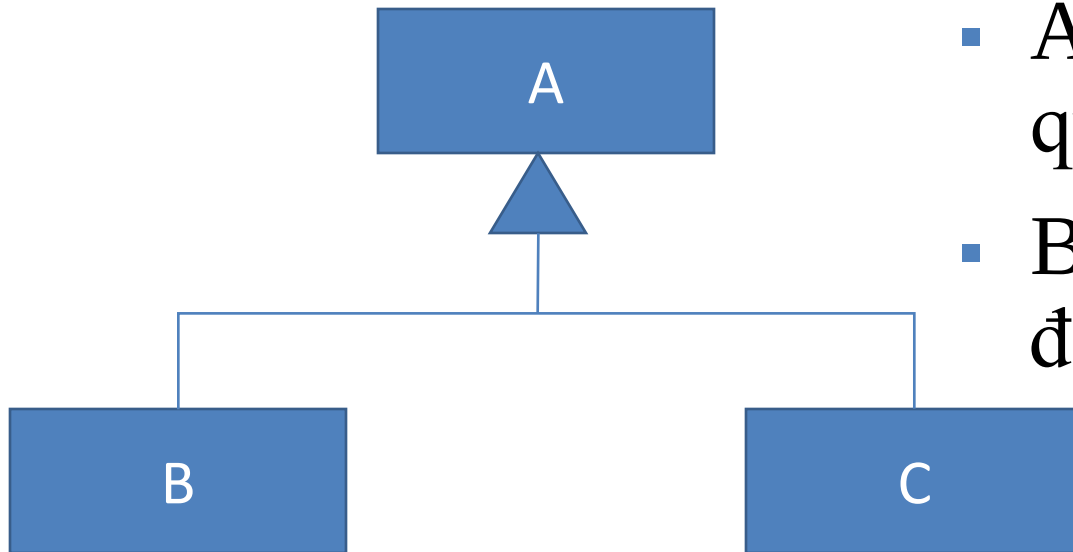


Ký hiệu

#330



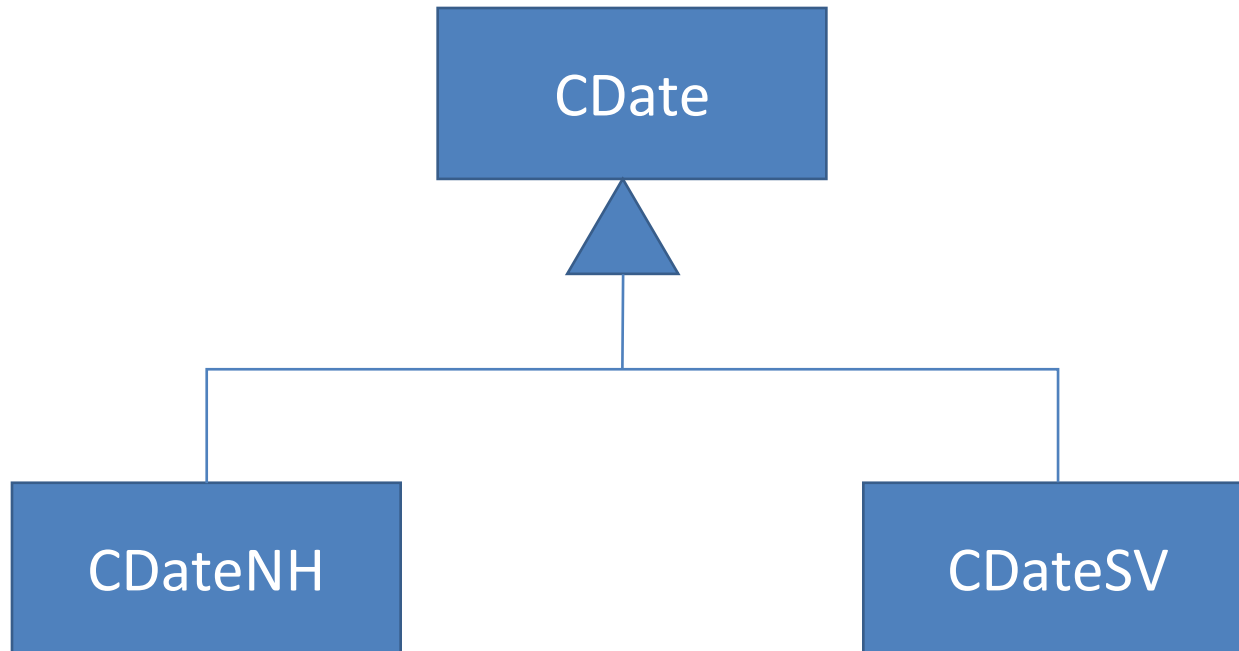
- A: Là trường hợp tổng quát của B
- B: Là trường hợp đặc biệt của A



- A: Là trường hợp tổng quát của B và C
- B, C: Là trường hợp đặc biệt của A

VD: Lớp ngày cho ngân hàng và sinh viên

#331



Khai báo

#332

```
class TênLớpCha
```

```
{
```

Thuộc tính và phương thức của lớp cha

```
}
```

```
class TênLớpDẫnXuất : TênLớpCha
```

```
{
```

Thuộc tính và phương thức bổ sung của
lớp dẫn xuất

```
}
```

Từ khóa **base**

Từ khóa base thường được sử dụng để gọi phương thức khởi tạo (phương thức dựng) của lớp cha

Ví dụ:

Class NhanVien

```
{  
    string strMaSo, strHoTen;  
    public NhanVien()  
    {  
        strMaSo = "";  
        strHoTen = "";  
    }  
}
```

Class NVBienChe:NhanVien

```
{  
    int bacLuong;  
    public NVBienChe():base()  
    {  
        bacLuong = 0;  
    }  
}
```

Khai báo

#334

Có 2 cách để định nghĩa hành động bổ sung cho phương thức đã có sẵn ở lớp cha trong lớp dẫn xuất (phương thức lớp dẫn xuất trùng tên với phương thức lớp cha)

- Dùng từ khóa **new**
- Dùng từ khóa **virtual** và **override**

Khai báo – Dùng từ khóa **new**

#335

```
class COSO
{
    protected kiểu data1;
    protected kiểu data2;
    public void Method1()
    {}
    public void Method2()
    {}
}
```

```
class DANXUAT : COSO
{
    private kiểu data3;
    public new void Method1()
    {}
    public void Method4()
    {}
}
```

Ví dụ

#336

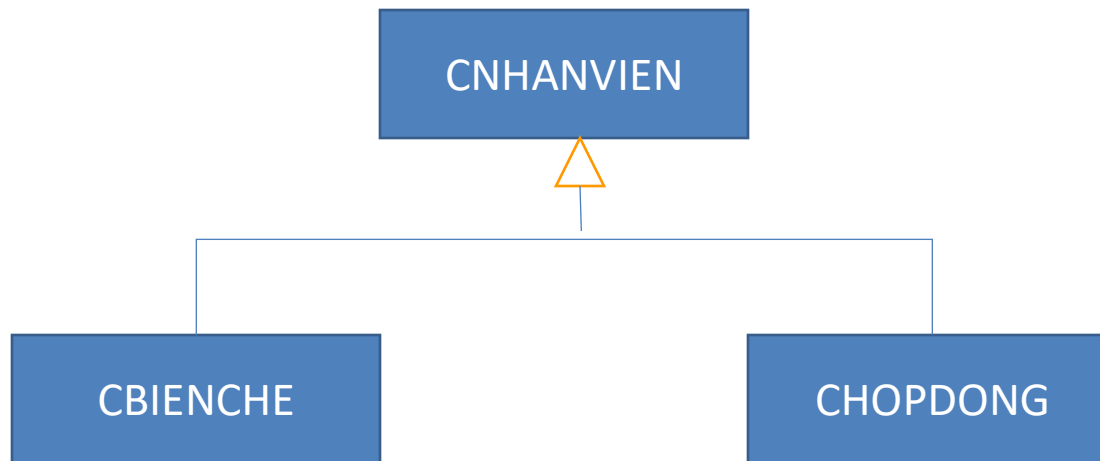
Viết chương trình nhập xuất nhân viên, biết rằng gồm 2 loại nhân viên: Nhân viên biên chế và nhân viên hợp đồng. Thông tin của nhân viên gồm: Mã số, Họ tên.

- Nhân viên biên chế có thông tin riêng là bậc lương.
- Nhân viên hợp đồng có thông tin riêng là số giờ làm.

Ví dụ

#337

Ta có cây kế thừa sau:



VD dùng từ khoá new

#338

```
class CNHANVIEN
```

```
{ protected int maso;
```

```
protected string hoten;
```

```
public void Nhap()
```

```
{ Console.Write("Nhap ma so nhan vien: ");
```

```
maso = int.Parse(Console.ReadLine());
```

```
Console.Write("Nhap ho ten nhan vien: ");
```

```
hoten = Console.ReadLine();
```

```
}
```

```
public void Xuat()
```

```
{ Console.WriteLine("Ma so: {0}\nHo ten: {1}", maso, hoten);
```

```
}
```

```
}
```

Ví dụ – Dùng từ khoá new

#339

```
class CBIENCHE : CNHANVIEN
{
    private float hesoluong;
    public new void Nhap()
    { base.Nhap();
      Console.Write("Nhap he so luong: ");
      hesoluong = float.Parse(Console.ReadLine());
    }
    public new void Xuat()
    { base.Xuat();
      Console.WriteLine("He so luong: " + hesoluong);
    }
}
```

Ví dụ – Dùng từ khoá new

#340

```
class CHOPDONG : CNHANVIEN
{
    private float sogio;
    public new void Nhap()
    {
        base.Nhap();
        Console.Write("Nhap so gio lam viec: ");
        sogio = float.Parse(Console.ReadLine());
    }
    public new void Xuat()
    {
        base.Xuat();
        Console.WriteLine("So gio lam viec: " + sogio);
    }
}
```

6.4 Virtual, Override, Abstract

#341

```
class COSO
{
    protected kiểu data1;
    protected kiểu data2;
    public virtual void Method1()
    {}
    public virtual void Method2()
    {}
}
```

```
class DANXUAT : COSO
{
    private kiểu data3;
    public override void Method1()
    {}
    public void Method4()
    {}
}
```

Dùng Virtual

#342

Ta xét ví dụ sau

```
class CNHANVIEN
```

```
{    protected int maso;    protected string hoten;
```

```
    public virtual void Nhap()
```

```
    {    Console.Write("Nhap ma so nhan vien: ");
```

```
        maso = int.Parse(Console.ReadLine());
```

```
        Console.Write("Nhap ho ten nhan vien: ");
```

```
        hoten = Console.ReadLine();
```

```
    }
```

```
    public virtual void Xuat()
```

```
    {    Console.WriteLine("Ma so: {0}\nHo ten: {1}", maso, hoten);
```

```
    }
```

```
}
```

Ví dụ – Dùng override

#343

```
class CBIENCHE : CNHANVIEN
{
    private float hesoluong;
    public override void Nhap()
    {
        base.Nhap();
        Console.Write("Nhap he so luong: ");
        hesoluong = float.Parse(Console.ReadLine());
    }
    public override void Xuat()
    {
        base.Xuat();
        Console.WriteLine("He so luong: " + hesoluong);
    }
}
```

Ví dụ – Dùng override

#344

```
class CHOPDONG : CNHANVIEN
{
    private float sogio;
    public override void Nhap()
    {
        base.Nhap();
        Console.Write("Nhap so gio lam viec: ");
        sogio = float.Parse(Console.ReadLine());
    }
    public override void Xuat()
    {
        base.Xuat();
        Console.WriteLine("So gio lam viec: " + sogio);
    }
}
```

Ví dụ - Sử dụng phương thức trong Main()

#345

```
static void Main(string[] args)
{
    CBIENCHE nvbc = new CBIENCHE();
    nvbc.Nhap();
    CHOPDONG nvhd = new CHOPDONG();
    nvhd.Nhap();
    Console.WriteLine("\nNhan vien bien che: ");
    nvbc.Xuat();
    Console.WriteLine("\nNhan vien hop dong: ");
    nvhd.Xuat();
}
```


Phạm vi kế thừa

#346

Có 3 phạm vi kế thừa:

- public
- protected
- private

Lưu ý: Nếu không nói rõ là phạm vi kế thừa gì, chúng ta ngầm định đó là kế thừa **public**

Phạm vi kế thừa

#347

- **public**: thành phần public & protected của lớp cơ sở là thành phần public & protected của lớp dẫn xuất.
- **protected**: thành phần public & protected của lớp cơ sở là thành phần protected của lớp dẫn xuất.
- **private**: thành phần public & protected của lớp cơ sở là thành phần private của lớp dẫn xuất.

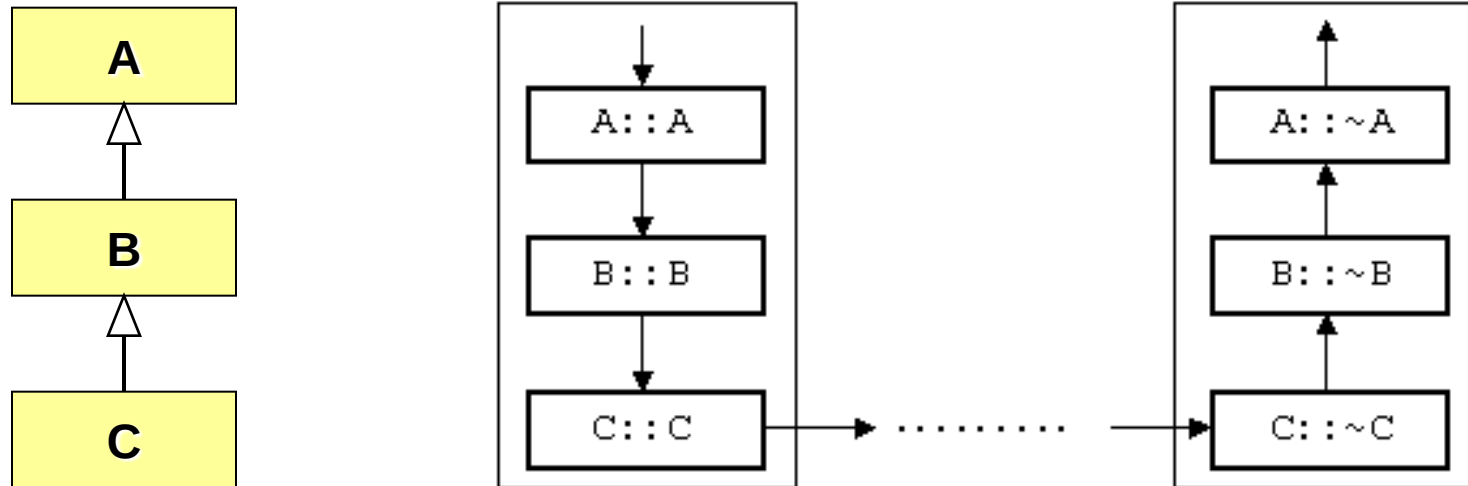
Phương thức thiết lập & huỷ trong kế thừa

#348

- Khi khởi tạo đối tượng:
 - Phương thức thiết lập của **lớp cha** sẽ được gọi trước
 - Sau đó mới là phương thức thiết lập của **lớp con**.
- Khi huỷ đối tượng:
 - Phương thức huỷ của **lớp con** sẽ được gọi trước
 - Sau đó mới là phương thức huỷ của **lớp cha**.

Phương thức thiết lập & huỷ trong kế thừa

#349



Phương thức thiết lập & huỷ trong kế thừa

#350

Trong phương thức thiết lập của lớp dẫn xuất, chúng ta có thể chỉ định phương thức thiết lập nào của lớp cơ sở sẽ được gọi thực hiện. **Nếu không chỉ định, phương thức thiết lập mặc định của lớp cơ sở sẽ được gọi.**

Phương thức thiết lập & huỷ trong kế thừa

#351

```
class A
{
    public A(){}
    public A(int){}
}
class B : public A
{
    public B(int) //Thực hiện A()
    {}
}
```

Phương thức thiết lập & huỷ trong kế thừa

#352

```
class A
{
    public A(){}
    public A(int){}
}
class B : public A
{
    public B(int) : base(int) //Thực hiện A(int)
    {}
}
```

Bài tập

#353

Thiết kế chương trình quản lý các đối tượng sau trong một Viện khoa học: nhà khoa học, nhà quản lý và NV phòng thí nghiệm. Các thành phần dữ liệu của các đối tượng trên:

- Nhà khoa học: họ tên, năm sinh, bằng cấp, chức vụ, số bài báo đã công bố, số ngày công trong tháng, bậc lương
- Nhà quản lý họ tên, năm sinh, bằng cấp, chức vụ, số ngày công trong tháng, bậc lương
- NV phòng thí nghiệm: họ tên, năm sinh, bằng cấp, lương trong tháng.

Thực hiện các yêu cầu sau:

- Các phương thức thiết lập để nhập liệu, biết rằng nhân viên phòng thí nghiệm lãnh lương khoán, còn lương của nhà khoa học và nhà quản lý bằng số ngày công trong tháng * bậc lương.
- Xuất dữ liệu ra màn hình
- In tổng lương đã chi trả cho từng loại đối tượng.

Trừu tượng hóa (Abstract)

#354

- Trừu tượng hóa là khả năng mô tả khái quát các thao tác chung của các lớp đối tượng.
- Đặc tính này giúp cho việc thiết kế lớp mang tính đa hình

Ví dụ

#355

- Nhận xét đoạn code sau

```
static void Main()  
{  
    AnPham a = new AnPham();  
    a.LayRa();  
  
    TapChi t = new TapChi();  
    t.LayRa();  
  
    a = t;  
    a.LayRa();  
}
```

Ví dụ2

#356

- Nhận xét đoạn code sau

```
static void Main()  
{  
    AnPham[] ds = new AnPham[100];  
    for(int i=0;i<100;i++)  
    {  
        if(Nhập tạp chí ?)  
        {  
            ds[i] = new TapChi();  
        }  
        else  
        {  
            ds[i] = new Sach();  
        }  
    }  
}
```

Lớp trừu tượng

#357

Phương thức trừu tượng là phương thức chỉ có tên thôi và nó phải được cài đặt lại ở tất cả các lớp kế thừa. Lớp trừu tượng chỉ thiết lập một cơ sở cho các lớp kế thừa mà nó không thể có bất kỳ một thể hiện nào tồn tại

```
abstract class COSO
{
    protected kiểu data1;
    protected kiểu data2;
    public abstract void Method1();
    public abstract void Method2();
}
```

```
class DANXUAT : COSO
{
    private kiểu data3;
    public override void Method1()
    {}
    public override void Method2()
    {}
}
```

Lớp trừu tượng

#358

```
abstract class Window
```

```
{  
    protected int top, left;  
    public Window(int top, int left)  
    {   this.top = top;  this.left = left; }  
    abstract public void DrawWindow( );  
}
```

```
class ListBox : Window
```

```
{  
    private string listBoxContents;  
    public ListBox(int top, int left, string contents) : base(top, left)  
    {   listBoxContents = contents; }  
    public override void DrawWindow( )  
    {  
        Console.WriteLine("Writing string to the listbox: {0}", listBoxContents);  
    }  
}
```

Lớp trừu tượng

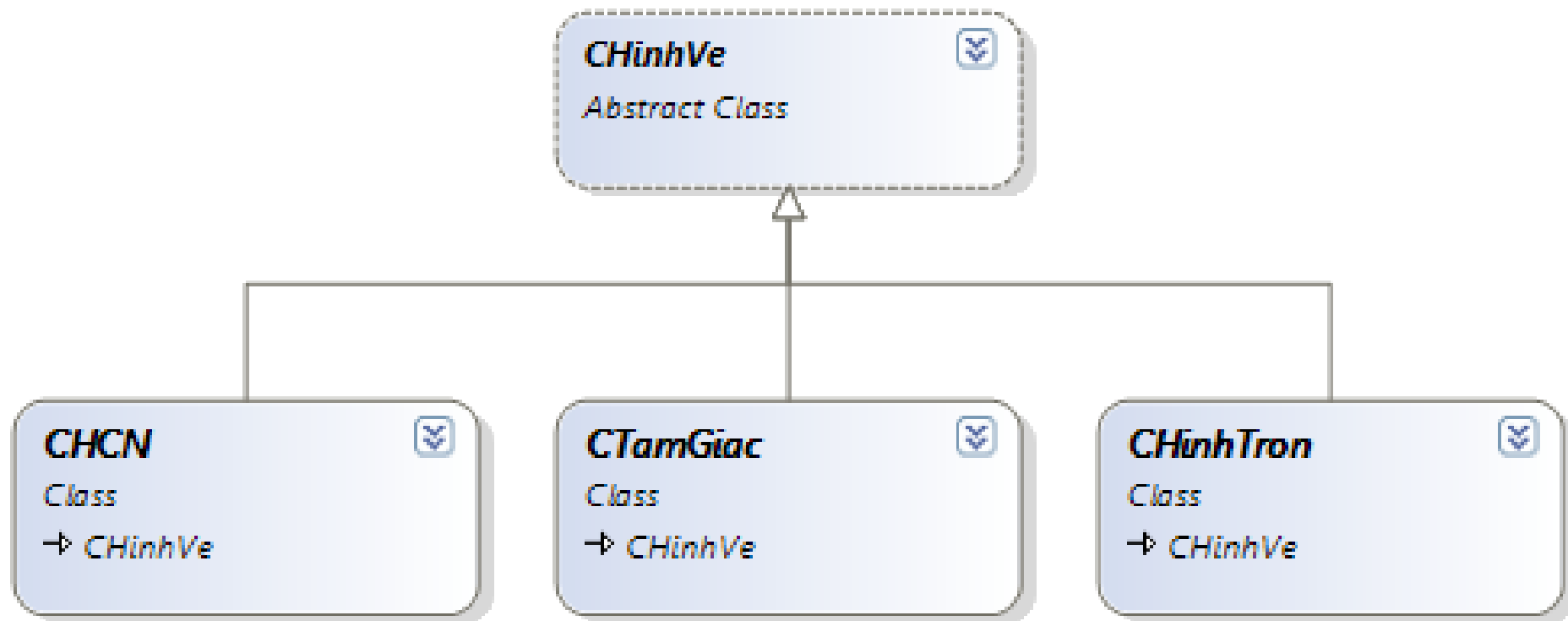
#359

```
public class Button : Window
{
    public Button( int top, int left): base(top, left)
    { }
    public override void DrawWindow( )
    {    Console.WriteLine("Drawing a button at {0}, {1}\n", top, left); }
}

public class Tester
{
    static void Main( )
    {
        Window[] winArray = new Window[3];
        winArray[0] = new ListBox(1,2,"First List Box");
        winArray[1] = new ListBox(3,4,"Second List Box");
        winArray[2] = new Button(5,6);
        for (int i = 0; i < 3; i++)
        {    winArray[i].DrawWindow( ); }
    }
}
```

Ví dụ

#360



Chương 7. Một số kỹ thuật nâng cao

TRẦN VIỆT KHÁNH

Email: tranvietkhanh79@gmail.com

MỤC TIÊU

#362

- Hiểu và sử dụng được các mẫu template hướng đối tượng có sẵn để hỗ trợ xây dựng phần mềm.
- Biết được cách debug và bắt lỗi chương trình.

Nội dung

#363

- Ngoại lệ (Exception)
- Mẫu (Template)

7.1 Ngoại lệ (Exception)

#364

- Một Exception (ngoại lệ) là một vấn đề xuất hiện trong khi thực thi một chương trình. Một Exception trong C# là một phản hồi về một tình huống ngoại lệ mà xuất hiện trong khi một chương trình đang chạy, ví dụ như chia cho số 0.
- Exception cung cấp một cách để truyền điều khiển từ một phần của một chương trình tới phần khác. Exception Handling (Xử lý ngoại lệ) trong C# được xây dựng dựa trên 4 từ khóa là: **try**, **catch**, **finally**, và **throw**.
- **try**: Một khối try nhận diện một khối code mà ở đó các exception cụ thể được kích hoạt. Nó được theo sau bởi một hoặc nhiều khối catch.
- **catch**: Một chương trình bắt một Exception với một Exception Handler tại vị trí trong một chương trình nơi bạn muốn xử lý vấn đề đó. Từ khóa **catch** trong C# chỉ dẫn việc bắt một exception.
- **finally**: Một khối finally được sử dụng để thực thi một tập hợp lệnh đã cho, dù có hay không một exception được ném hoặc không được ném. Ví dụ, nếu bạn mở một file, nó phải được đóng, nếu không sẽ có một exception được tạo ra.
- **throw**: Một chương trình ném một exception khi có một vấn đề xuất hiện. Điều này được thực hiện bởi sử dụng từ khóa **throw** trong C#.

Ngoại lệ (Exception)

#365

■ Cú pháp

- Giả sử một khối tạo một Exception, một phương thức bắt một exception bởi sử dụng kết hợp các từ khóa try và catch. Một khối try/catch được đặt xung quanh code mà có thể tạo một exception. Code bên trong một khối try/catch được xem như là code được bảo vệ, và cú pháp để sử dụng try/catch trong C# như sau:

```
■ try { // các lệnh có thể gây ra ngoại lệ (exception) }  
    catch( tên_ngoại_lệ e1 ) { // phần code để xử lý lỗi }  
    catch( tên_ngoại_lệ e2 ) { // phần code để xử lý lỗi }  
    catch( tên_ngoại_lệ eN ) { // phần code để xử lý lỗi }  
    finally { // các lệnh được thực thi }
```

7.2 Mẫu (Template)

#366

- Template trong C++ nói cho dễ hiểu là "kiểu dữ liệu" được quyết định tại compile time.
 - Compile time: Lúc chương trình được compiler biên dịch
 - Run time: Lúc chương trình đã được dịch ra executable và có thể chạy
- Template có hai loại chính:
 - Hàm template
 - Lớp template

Ví dụ:

```
template <typename T>
```

```
class class_template {
```

```
    T data[10];
```

```
};
```

```
template<typename T>
```

```
void function_template( const vector<T>& vec > {
```

```
}
```

Mẫu (Template)

#367

▪ Cú pháp khi dùng template

Có hai định nghĩa cơ bản mà chúng ta cần phải biết về template là:

1. Tham số template:

```
template<typename T>  
T max( T a, T b ) {  
}
```

2. Đối template:

```
max<int>( 3, 4 );
```

- Tham số và đối template cũng giống như tham số và đối trong hàm, số lượng không bị giới hạn, bạn có thể khai báo tùy ý số lượng đối và tham số,

- Ví dụ:

```
template<typename T1, typename T2, typename T3, typename T4>  
void func( T1 t1, T2 t2, T3 t3, T4 t4 ) {  
  
}
```

Tài liệu tham khảo

#368

- [1] PSG.TS Trần Đan Thư, Lập trình hướng đối tượng, Khoa CNTT Đại học Khoa Học Tự Nhiên TP. HCM, 2010.
- [2] Trương Văn Chí Công, Giáo trình Lập trình hướng đối tượng C++, Khoa CNTT-ĐH Cần Thơ, 2003.
- [3] Ali Bahrami, “Object-oriented Systems Development”, McGraw-Hill, 1999.
- [4] Herbert Schildt, C++ A beginner’s guide, 2nd edition, McGraw-Hill, 2003.
- [5] Herbert Schildt, C++: the complete reference, 3rd edition, McGrawHill, 1998.
- [6] Bruce Eckel, Thinking in C++, Prentice Hall Inc., 2000.
- [7] Ivor Horton, Beginning Visual C++ 2005, Wiley Publising Inc, 2006.
- [8] <http://www.glib.hcmus.edu.vn> (Thư viện Đại học Khoa Học Tự Nhiên TP. HCM)

FAQs

#369



- GV: Trần Việt Khánh
- Email: tranvietkhanh79@gmail.com