Chương 1. Cơ Bản về Ngôn ngữ C#

GV: Trần Việt Khánh

Email: tranvietkhanh79@gmail.com

Cập nhật: 10 tháng 09 năm 2017

Nội dung

- 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 - Lập trình là gì?

- 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?

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.

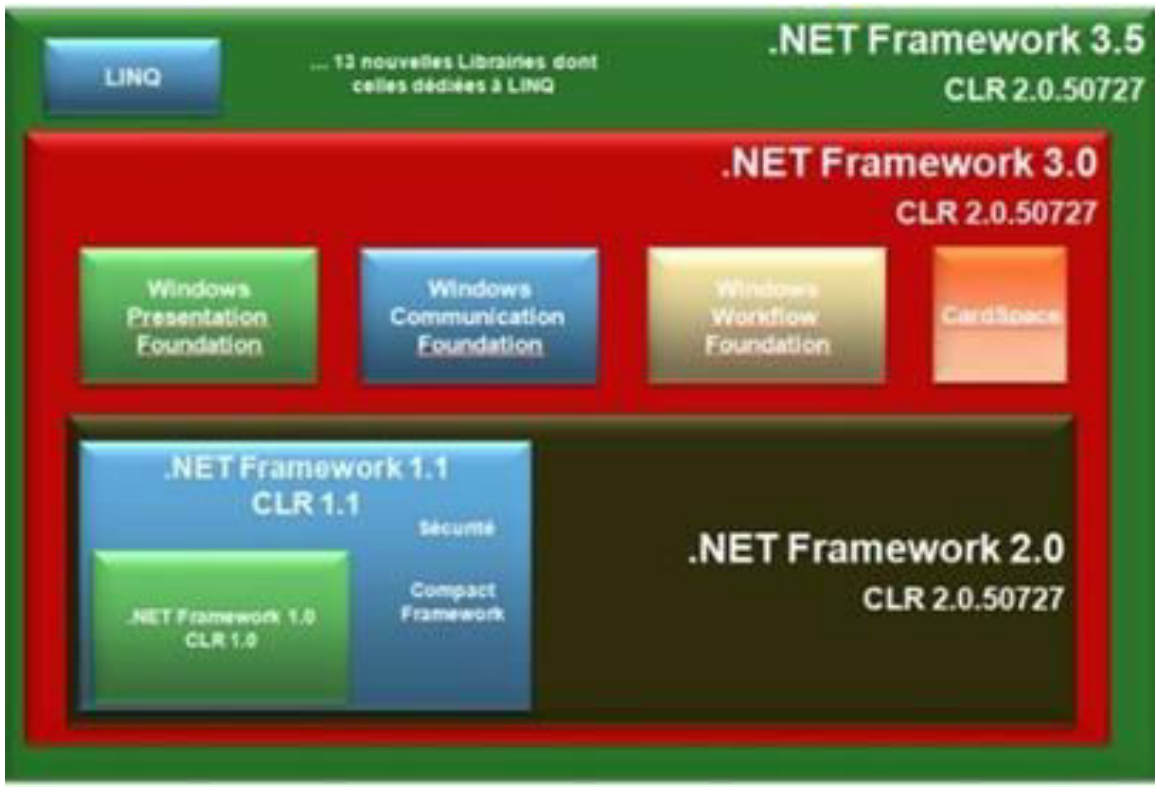
.NET Framework

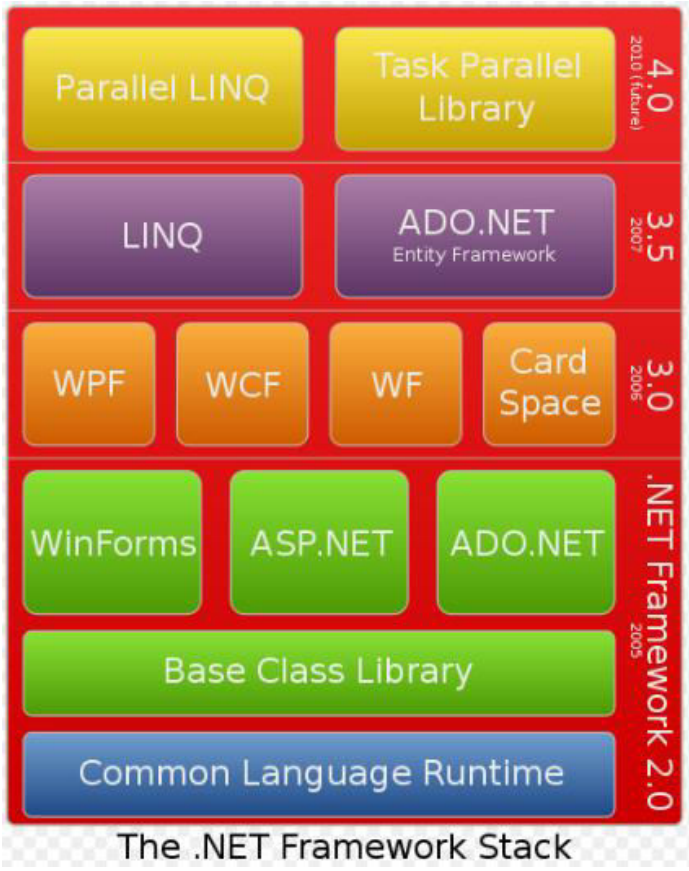
- Framework là một tập hợp các thư viện để hỗ trợ cho người lập trình.
- Mỗi Framework được tạo ra có một kiến trúc khác nhau ⇨ LTV phải tuân theo kiến trúc đó
- .NET Framework là thư viện tài nguyên của Microsoft, hỗ trợ cho các lập trình viên trong nhiều yêu cầu khác nhau.

.NET Framework

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





The .NET Framework Stack

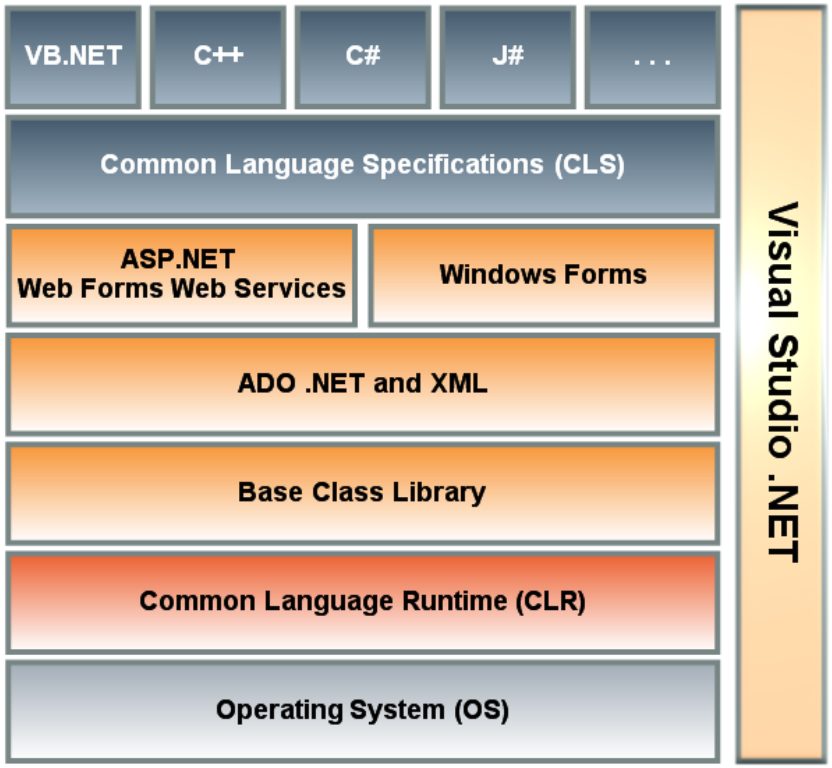
.NET Framework

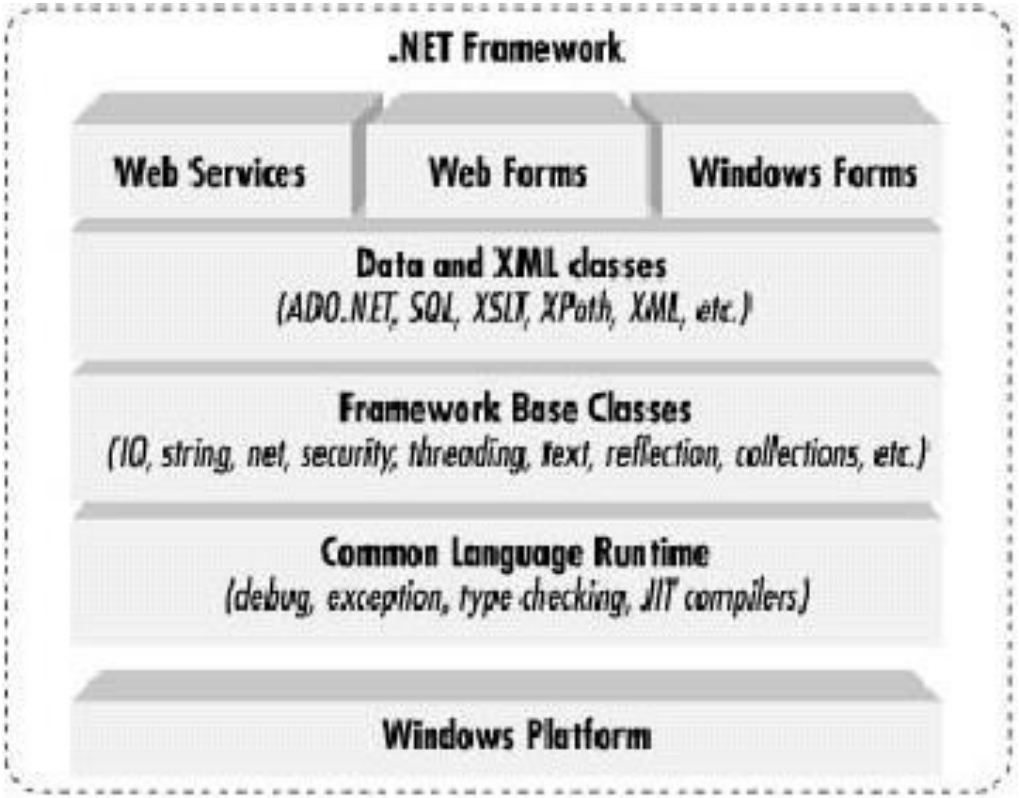
- Các ngôn ngữ : C#, VB.Net, J#, F#, VC++…
- Công cụ phát triển Visual Studio
- Lớp đặc tả ngôn ngữ dùng chung (CLS)
- Các thư viện đê phát triển ứng dụng
- Bộ thực thi ngôn ngữ dùng chung (CLR)

.NET Framework

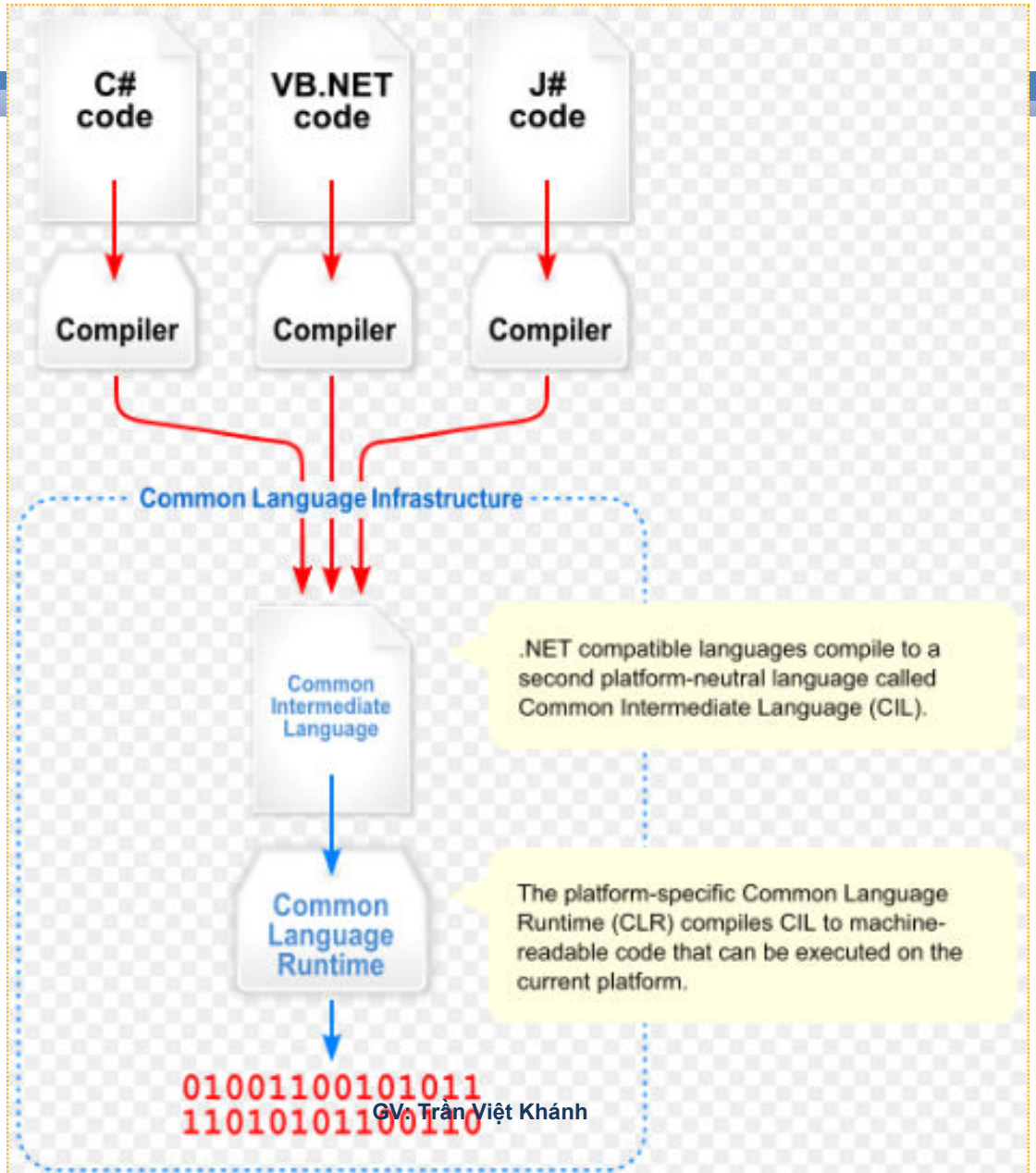
- Chương trình được biên dịch thành ngôn ngữ trung gian (MSIL - Microsoft Intermediate Language), sau đó chúng được CLR thực thi.
- Common Language Runtime - CLR, nền tảng hướng đối tượng cho phát triển ứng dụng Windows và Web mà các ngôn ngữ có thể chia sẻ sử dụng.
- Bộ thư viện Framework Class Library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

.NET Framework





.NET Framework



Ngôn ngữ C#

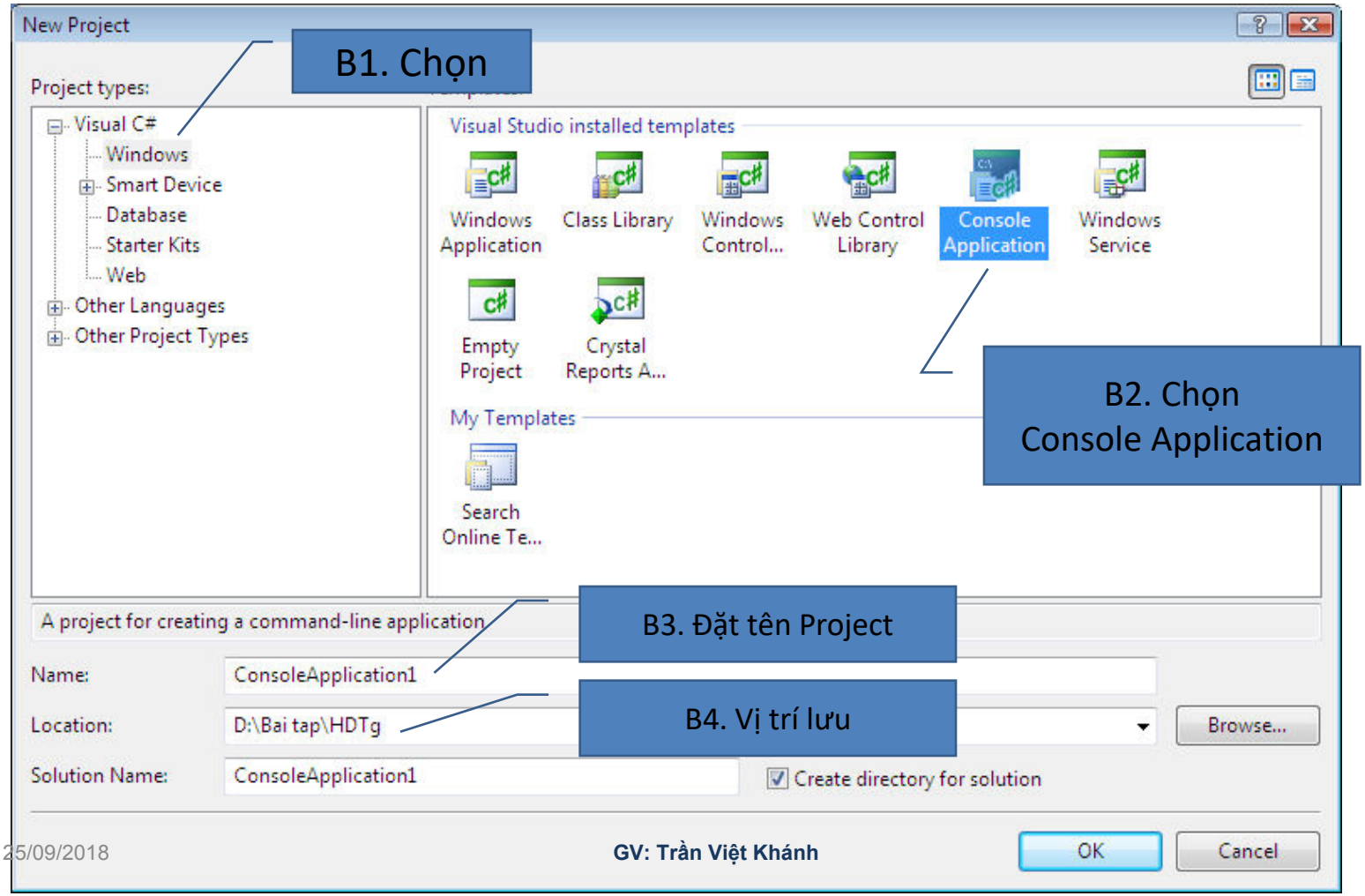
- Ngôn ngữ lập trình được xây dựng dựa trên nền tảng những ngôn ngữ tương tự C (C, C++, Java) nhưng hoạt động trên .Net Framework.
- Hoạt động trên .NET Framework.

Ngôn ngữ C#

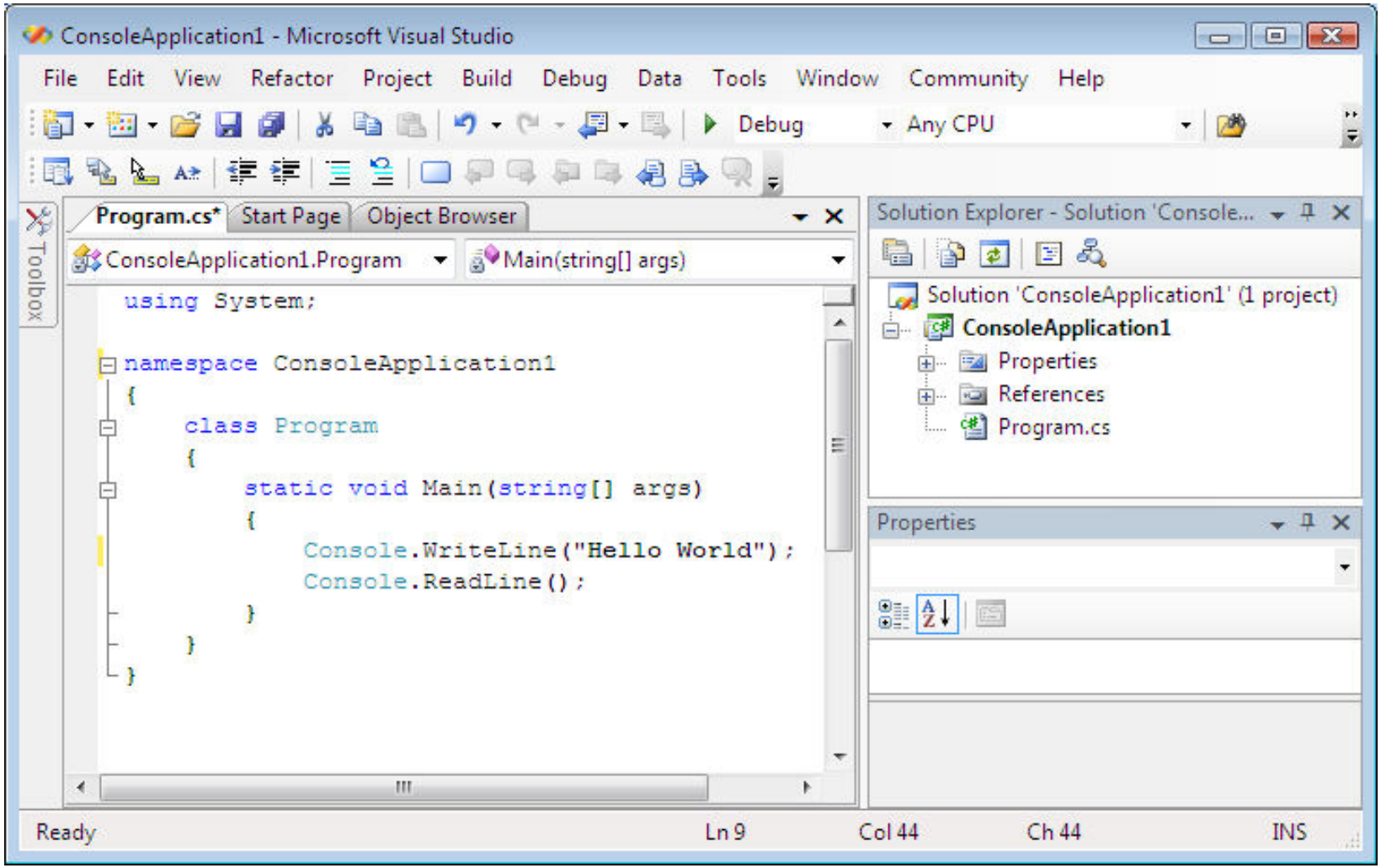
- Dựa trên phương pháp thiết kế hướng đối tượng.
- Ứng dụng : Console, Winform, Webform.
- Có tính diễn đạt ngữ nghĩa cao.
- Phân biệt chữ hoa thường.

Khởi Tạo Project

Tạo project dạng Console: File ⇨ New ⇨ Project



Khởi Tạo Project



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

Khởi Tạo Project

- 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

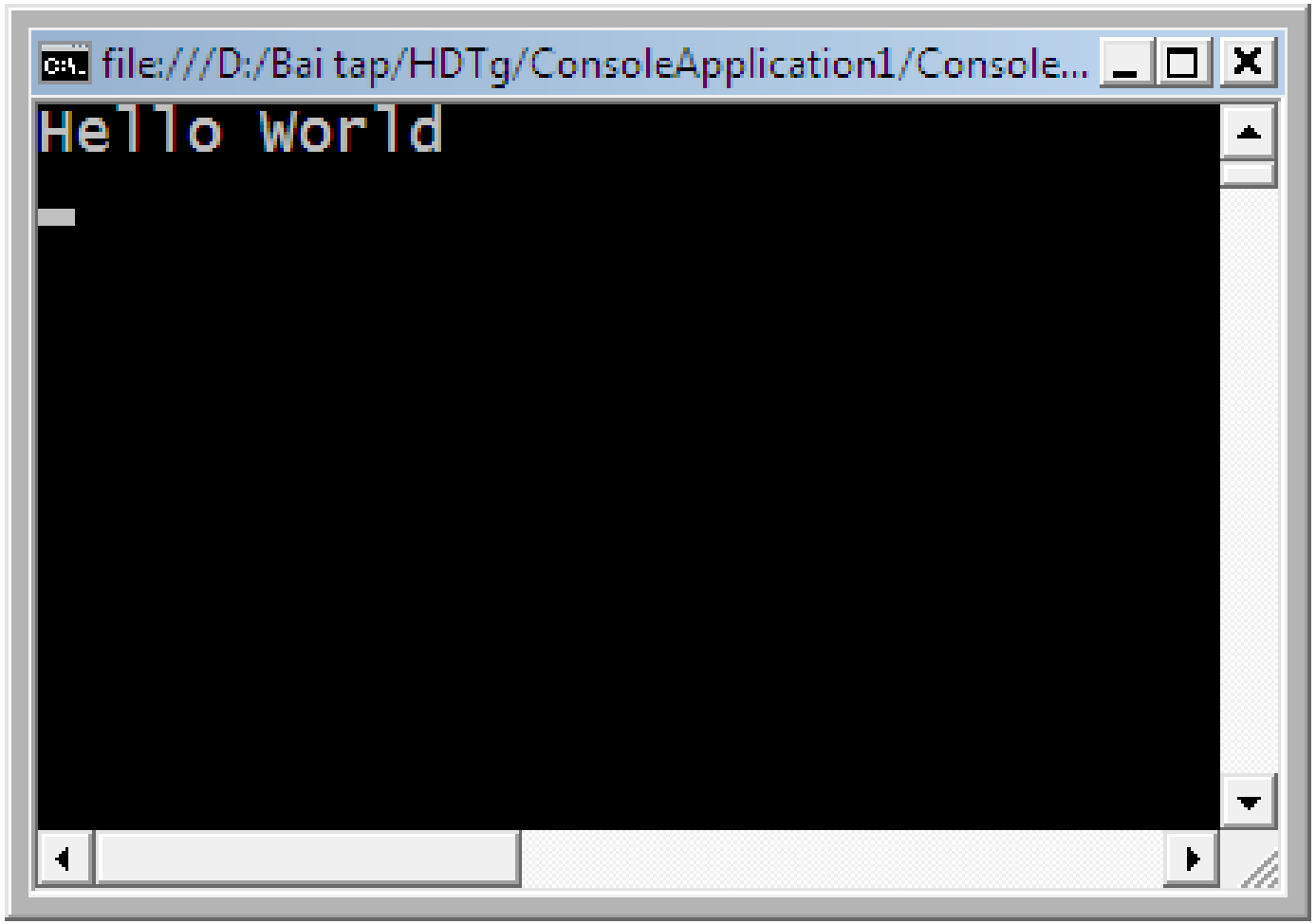
- Trình biên dịch (compiler) sẽ biên dịch các tập tin chứa ngôn ngữ C# thường là các file .cs trong project thành một tập tin chạy chương trình .exe
- Có 2 cách biên dịch :
  - Tại cửa sổ cmd, gõ : csc.exe tenfile.cs
  - Nhấn Build / Compile (hoặc Build / Build Solution) ⇨ Biên dịch cả project.

Compile & chạy chương trình

Chạy chương trình

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

Kết quả


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

Từ khoá – Keywords

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)

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)

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

Khai báo trực tiếp bằng cách ghi đầy đủ namespace.

VD:

System.Media.SoundPlayer spStart

=new
System.Media.SoundPlayer("start.wav");

Sử dụng Namespace

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

- 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)

- 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)

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

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

Số nguyên

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

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

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)

- 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

- 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

- 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

- 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)

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)

- 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)

- 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)

- 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)


```
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)

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

Identifier (định danh)

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

- 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

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

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

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

- 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

- 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

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

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


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

Hoặc

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

Hàm nhập – Console.System

Mẫu chung:

<KDL> Biến;

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

Hoặc

<KDL> Biến;

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

Bài tập

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:

Tổng lương = lcb*hs + pc

1.2 Tham số mặc nhiên

- 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

- 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

- 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 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)

- 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();
```

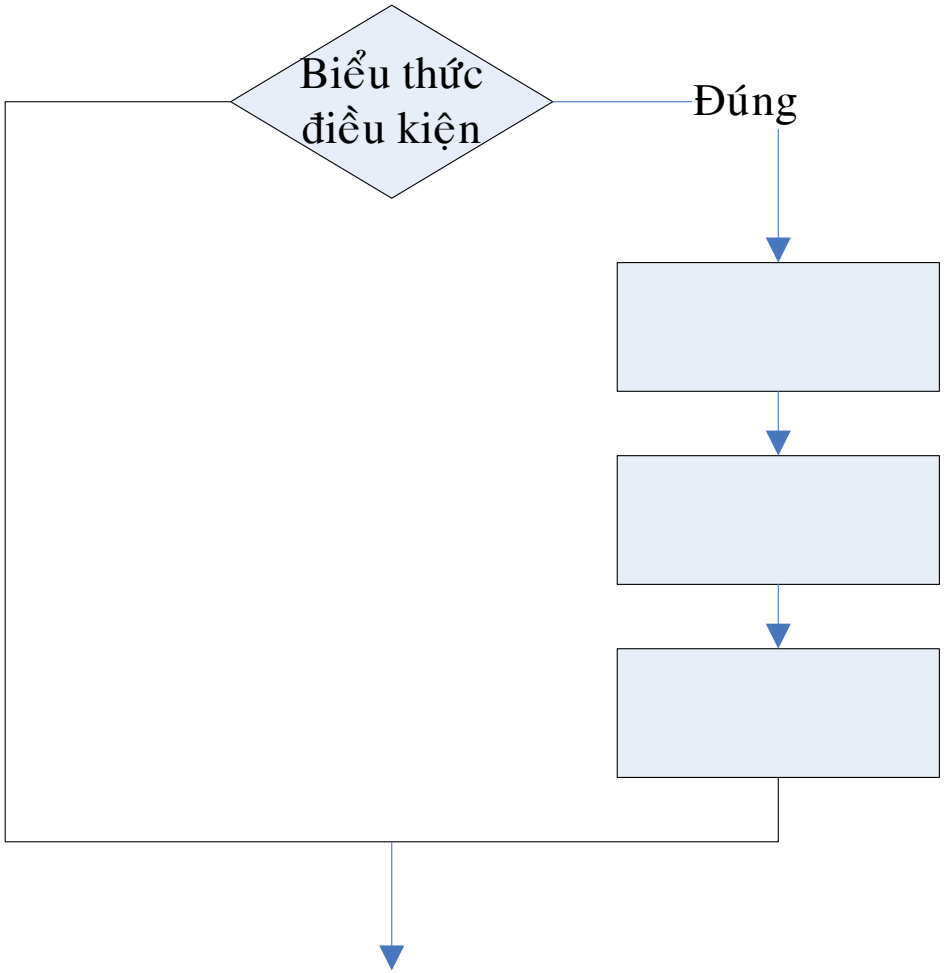
Cấu trúc điều khiển

- 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


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

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



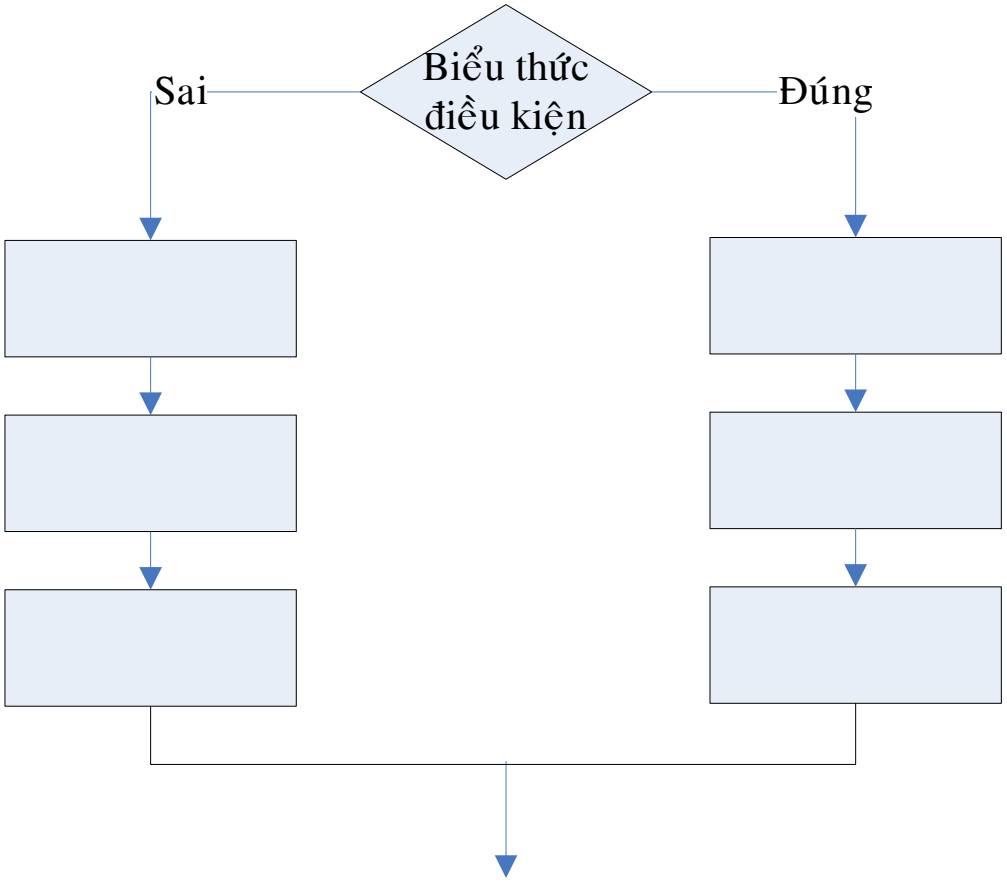
Cấu trúc rẽ nhánh (tt)

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)


```
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


```
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


```
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


```
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;
        }
}
```

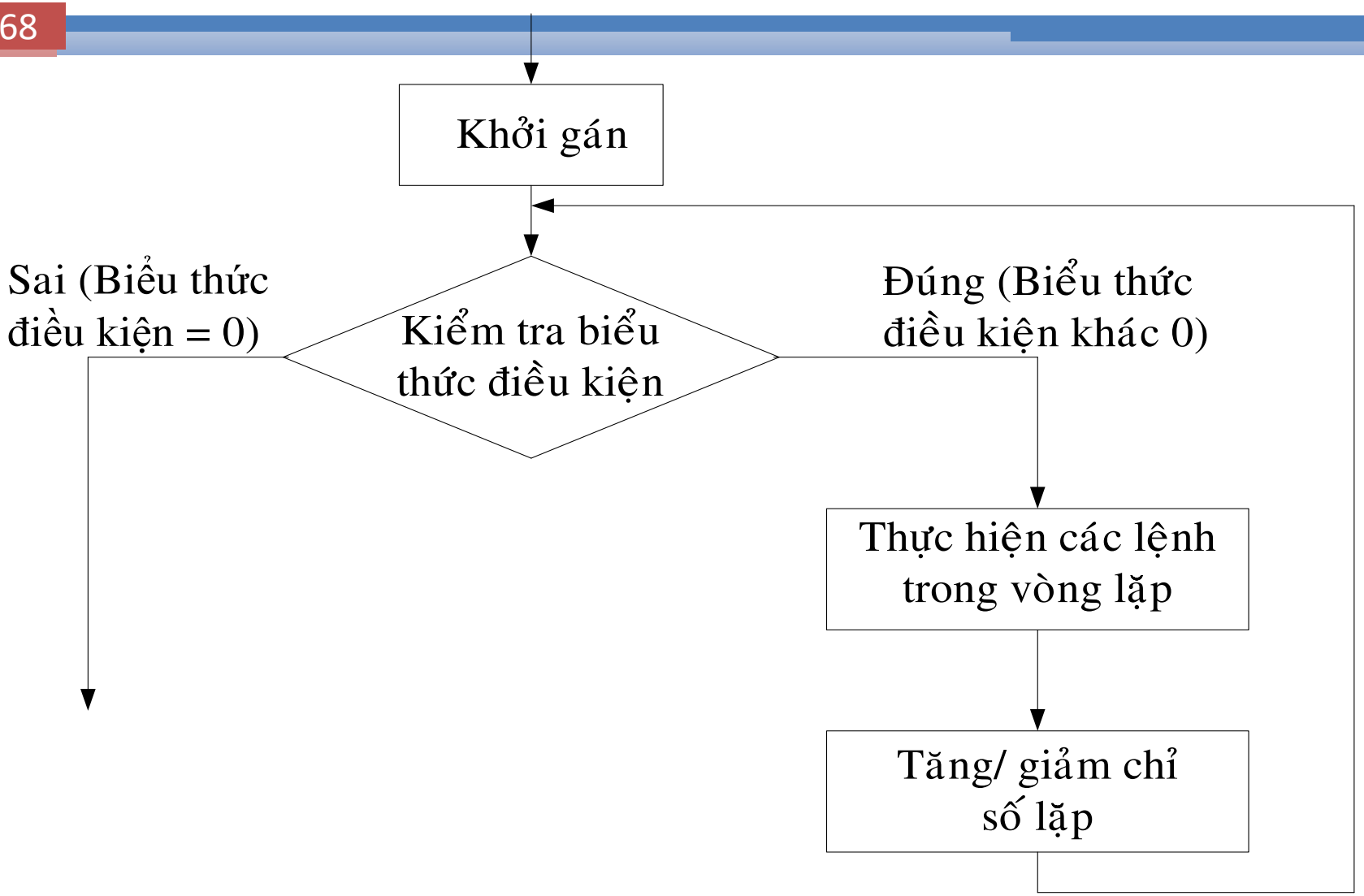
Bài tập

- 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

- while
- for
- do…while
- foreach

Cấu trúc lặp while và for



Hoạt động cấu trúc lặp while và for

- 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

< Khởi gán>;

while (<biểu thức điều kiện>)

{

lệnh/ khối lệnh;

<tăng/giảm chỉ số lặp>;

}

VD: Xuất ra màn hình 10 dòng chữ ABC


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

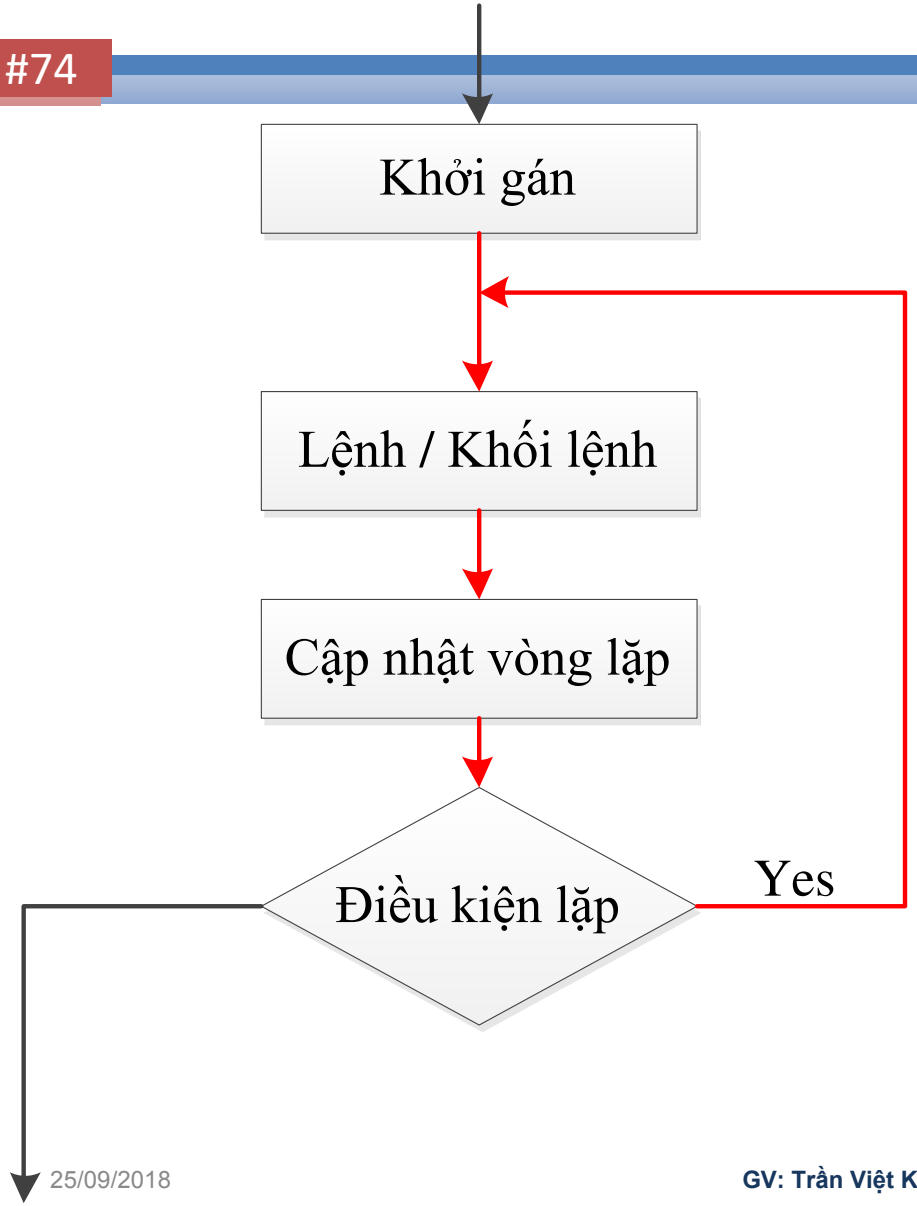
Cấu trúc lặp for


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

VD: Xuất ra màn hình 10 dòng chữ ABC


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

Cấu trúc lặp do…while



do

{

< khối lệnh> ;

} while (biểu thức ĐK);

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

Cấu trúc lặp foreach

- 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


```
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

- 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

- 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 hàmGetTime 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

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

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

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

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

- 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


```
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

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

- 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 → Không trả về giá trị → 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)
{
    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])

{

<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

$$S_n = 1 + 2 + 3 + \cdots + n \qquad ; n > 0$$

#89

- 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 → 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)
{
   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 làm 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();
}
```

Hàm inline

```
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 châm 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

- 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

- 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

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

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)
{
   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)
{
   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)
{
    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

- 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

#100

- 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.

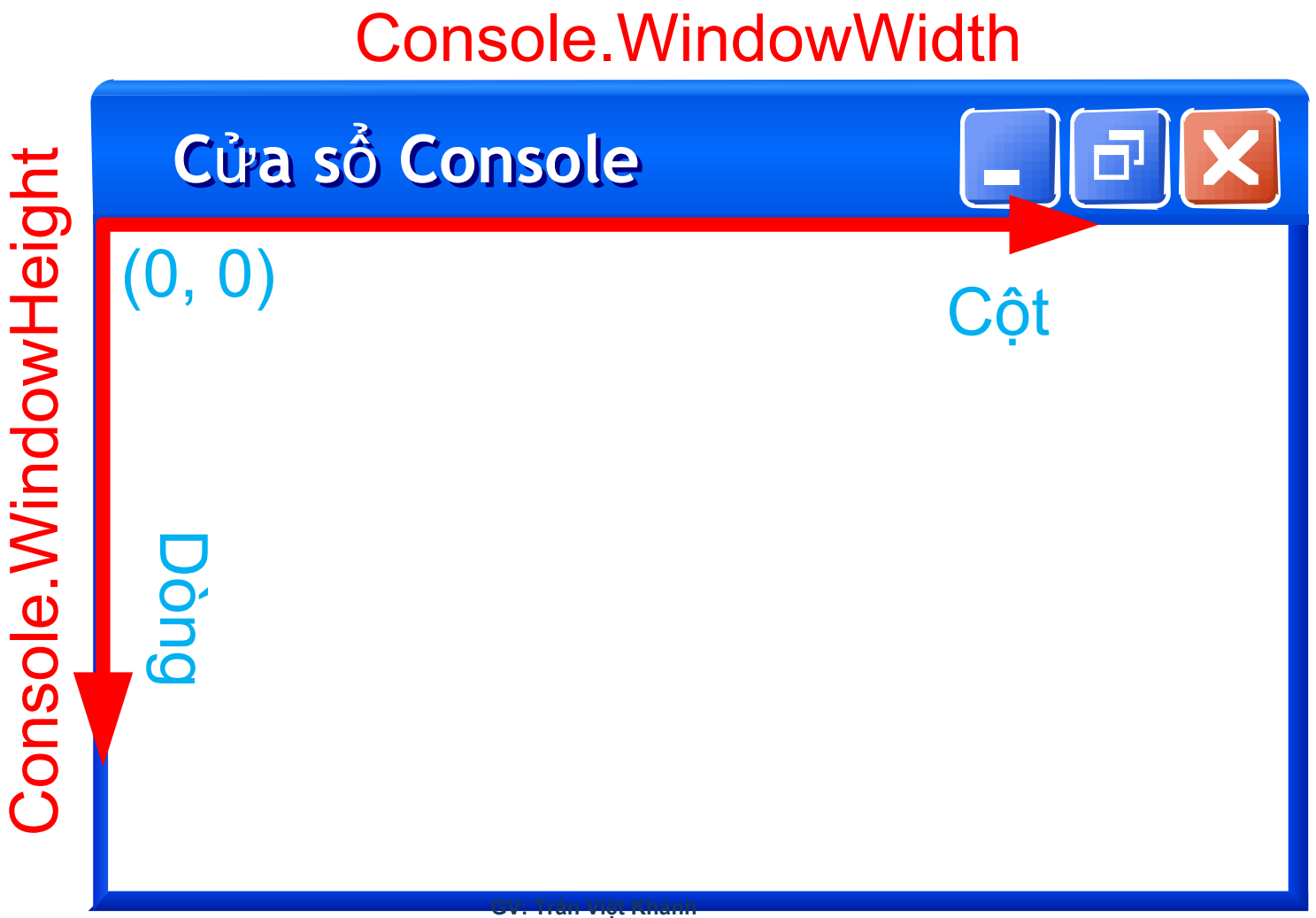
  Ví dụ: Nhập n=6

  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

- 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



Di chuyển dấu nháy

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

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

```
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

- Sử dụng lớp DateTime

DateTime d = DateTime.Now

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

Xác định phím nhấn trong Console

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

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

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

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

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

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

- Có nhấn phím:

  Console.KeyAvailable = true

- Không nhấn phím:

  Console.KeyAvailable = false

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

#112

```
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

- 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 cửa sổ console: Console.Clear();

Hằng số màu

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

- 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

```
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

- 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


```
public static void Main()
{
    int n;
    Console.Write("Nhap kich thuoc mang: ");
    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


```
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


```
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

Hoặc

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

Bài tập

#122

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

```
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


```
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


```
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


```
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

#127

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

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

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

- Thay thế chuỗi con

  Replace(chuỗi cần thay, chuỗi thay thế)

  string s;

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

  → s = "Thay The chuoi"

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

  → s = "THay THe chuoi"

Thao tác cơ bản trên chuỗi ký tự

#131

- Định dạng chuỗi:

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

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

Thao tác cơ bản trên chuỗi ký tự

- Chiều dài chuỗi: Thuộc tính Length

```
string s = "Xin chao";

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

Thao tác cơ bản trên chuỗi ký tự

- 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

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

#135

- 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

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

- 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


```
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

- 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)
{
        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

#141

```
static void DocFile(string tenfile)
{
    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

- Là các lỗi khi chạy chương trình, thông thường xãy 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

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

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

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

- 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


```
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


```
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

>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

- 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


```
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


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

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

BÀI TẬP

#153

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

FAQs

THANKS FOR LISTENING !