



## **CHƯƠNG 2: NGÔN NGỮ LẬP TRÌNH C#**

**Gv: Đặng Hữu Nghị**

**Bộ môn: CNPM**

**Khoa: CNTT**

# NỘI DUNG

2.1. Biến và Kiểu dữ liệu

2.2. Hàm xuất

2.3. Hàm nhập

2.4. Cấu trúc điều khiển

## 2.1. BIẾN VÀ KIỂU DỮ LIỆU

2.1.1. Biến

2.1.2. Kiểu dữ liệu

2.1.3. Chuyển đổi kiểu dữ liệu

2.1.4. Toán tử số học

2.1.5. Ký hiệu so sánh và phép toán bit

## 2.1.1. BIẾN

- Biến là nơi lưu trữ dữ liệu của chương trình. Dữ liệu của biến nằm trong bộ nhớ vật lý Ram và có thể thay đổi được.
- Muốn sử dụng biến trước tiên lập trình phải chỉ định biến có một kiểu dữ liệu cụ thể nào đó.
- Cú pháp khai báo biến:

**<Kiểu dữ liệu> <Tên biến>;**

## 2.1.2. KIỂU DỮ LIỆU

- 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.
- **Stack**: một vùng bộ nhớ dành lưu trữ dữ liệu với chiều dài cố định
  - ✓ Ví dụ : số nguyên kiểu int luôn chiếm dụng 4 bytes.
  - ✓ Mỗi chương trình khi thực thi đều được cấp riêng 1 stack mà các chương trình khác không truy cập tới được
- **Heap**: một vùng bộ nhớ dùng lưu trữ dữ liệu có dung lượng thay đổi.
  - ✓ Ví dụ: như kiểu string, khi ta tạo đối tượng thuộc lớp string, thì đối tượng sẽ được xác định bởi hai thành phần: Địa chỉ đối tượng lưu ở stack, còn giá trị đối tượng thì sẽ lưu ở heap.

# 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

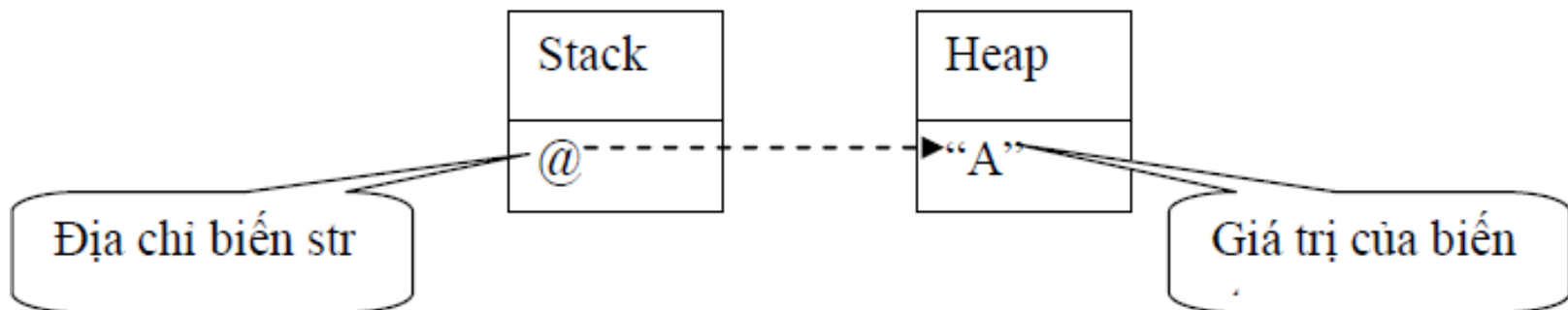
Kiểu C#	Kích thước (byte)	Kiểu .NET	Miền giá trị	Mô tả
float	4	Single	Từ $3,4E-38$ đến $3,4E+38$	Kiểu dấu chấm động, với 7 chữ số có nghĩa
double	8	Double	Từ $1,7E308$ đến $1,7E+308$	Kiểu dấu chấm động có độ chính xác gấp đôi, với 15 chữ số có nghĩa
decimal	8	Decimal		Có độ chính xác đến 28 con số, phải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

```
string str = "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

<Kiểu dữ liệu> [ ]<Tên mảng> = new <Kiểu dữ liệu> [Số phần tử];

Ví Dụ: Khai báo mảng số nguyên arr gồm 5 phần tử

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

# KIỂU MẢNG

## ○ Mảng hai chiều

$\text{<KDL> [,] <Tên mảng> = new <KDL> [\text{Số dòng}, \text{số cột}];$

## ○ VD: Khai báo ma trận số nguyên mt gồm 5 dòng và 3 cột

$\text{long [ , ] mt = new long [5, 3];}$

- Ví dụ: XD c/tr cho phép nhập vào 1 dãy số nguyên. Tính tổng của dãy số đó

```
using System;
```

```
namespace Mang_01
```

```
{  
    class Program  
    {  
        static void Main( )  
        {  
            int n;  
            Console.Write("Nhap so phan tu cua day so: ");  
            do  
                n = int.Parse(Console.ReadLine());  
            while (n <= 0);  
            Int16[] arr = new Int16 [n];  
            for(int i=0;i<n;i++)  
            {  
                Console.Write("A[{0}] = ",i+1);  
                arr[i]= Int16.Parse(Console.ReadLine());  
            }  
            Int16 s = 0;  
            for (int i = 0; i < n; i++)  
                s += arr[i];  
            Console.Write("Tong day so bang: {0}",s);  
            Console.ReadLine();  
        }  
    }  
}
```

## 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)

- Ví Dụ 1:

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

Khi đó giá trị của Hai = 0; Ba = 1; ... ; ChuNhat = 6

- Ví Dụ 2:

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

Khi đó giá trị của Hai = 1; Ba = 2; ... ; ChuNhat = 7

## ENUM (KIỂU LIỆT KÊ)

- Ví Dụ 3:

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

Khi đó giá trị của Hai = 1; Ba = 2; ... ; Sau=10;  
Bay=11;ChuNhat = 12



# ENUM (KIỂU LIỆT KÊ)

- Mặc định, enum có giá trị nguyên cho các hằng số, để khai báo kiểu khác như **byte**, các bạn sử dụng dấu : tên kiểu ngay sau tên tập hợp.
- Ví dụ:

```
enum Ngay : byte
{
    ThuHai,
    ThuBa,
    ThuTu,
    ThuNam,
    ThuSau,
    ThuBay,
    ChuNhat
}
```

Ví dụ:

```
using System;
namespace Kieu_Enum
{
    class Program
    {
        enum Days { Sun, Mon, Tue, Wed, Thu, Fri, Sat };
        static void Main(string[] args)
        {
            Console.WriteLine("Enum trong C#");
            Console.WriteLine("-----\n");

            int dau_tuan = (int)Days.Mon;
            int cuoi_tuan = (int)Days.Fri;
            Console.WriteLine("Thu hai: {0}", dau_tuan);
            Console.WriteLine("Thu sau: {0}", cuoi_tuan);
            Console.ReadKey();
        }
    }
}
```

- Ví dụ 2: Đổi năm âm lịch sang dương lịch

```
using System;
```

```
namespace Am_lich
```

```
{
```

```
    class Program
```

```
    {
```

```
        enum can { canh, tan, nham, quy, giap, at, binh, dinh, mau, ky };
```

```
        enum chi { than, dau, tuat, hoi, ty, suu, dan, mao, thin, ty, ngo, mui};
```

```
        static void Main( )
```

```
        {
```

```
            int nam;
```

```
            Console.Write("Nhap nam duong lich: ");
```

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

```
            Console.Write("Nam am lich la: {0} {1}",(can) (nam % 10), (chi)(nam % 12));
```

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

```
        }
```

```
    }
```

```
}
```

# 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();
}
```

## 2.1.3. CHUYỂN ĐỔI KIỂU DỮ LIỆU

### ○ Sử dụng lớp **Convert**:

- `ToInt32(<Tham số>)`
- `ToInt64(<Tham số>)`
- `ToString(<Tham số>)`
- `toDouble(<Tham số>)`
- `ToBoolean(<Tham số>)`
- `Byte(<Tham số>)`
- ...

<Tham số>: có thể là chuỗi ký tự, hằng số, biến số nguyên, số thực hoặc kiểu bool

## 2.1.3. CHUYỂN ĐỔI KIỂU DỮ LIỆU

### ○ Ví dụ:

```
int a = Convert.ToInt32("10");  
//chuyển chuỗi 10 sang số nguyên  
bool b = Convert.ToBoolean(27);  
//chuyển số 27 sang kiểu boolean  
bool a = Convert.ToBoolean("hello");  
//Sai định dạng, không chuyển được  
int b=Convert.ToInt32("123456787654");  
//Tràn bộ nhớ, không chuyển được  
double d = Convert.ToDouble(null);  
//Trả về giá trị mặc định
```

## 2.1.3. CHUYỂN ĐỔI KIỂU DỮ LIỆU

### ○ Sử dụng phương thức Parse:

- Double.Parse(<chuoi>): Chuyển chuỗi về kiểu Double
- Int32.Parse(<chuoi>): Chuyển chuỗi về kiểu Int32
- Int64.Parse(<chuoi>): Chuyển chuỗi về kiểu Int64
- Boolean.Parse(<chuoi>): Chuyển chuỗi về kiểu Boolean
- Single.Parse(<chuoi>): Chuyển chuỗi về kiểu Single
- ...



## 2.1.3. CHUYỂN ĐỔI KIỂU DỮ LIỆU

### ○ Ví dụ:

```
int a = Int32.Parse("123");  
//a sẽ mang giá trị 123  
float b = Float.Parse("20.7");  
//b sẽ mang giá trị 20.7  
bool c = Boolean.Parse("true");  
//c sẽ mang giá trị true  
int a = Int32.Parse("Hello");  
//sai định dạng, không chuyển được  
byte b = Byte.Parse("100000000000");  
//quá giới hạn, không chuyển được  
bool c = Boolean.Parse(null);  
//tham số là null, không chuyển được
```

## 2.1.3. CHUYỂN ĐỔI KIỂU DỮ LIỆU

### ○ Casting (ép kiểu):

- Đây là cách chuyển kiểu được sử dụng khi muốn chuyển đổi giữa những kiểu dữ liệu gần tương tự nhau. Thường áp dụng với kiểu số.
- Ví dụ:

```
int x= 10;  
float y = x;  
//chuyển đổi ngầm định, y = 10.0  
int z = (int) y;  
//chuyển đổi rõ ràng, z = 10  
int x = 10, y = 4;  
float z = (float)x / y;
```

## 2.1.4. 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	

## 2.1.5. 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

## 2.1.6. CÁC HÀM TOÁN HỌC

Hàm	Chức năng
Abs(x)	Trả về giá trị tuyệt đối của một số thực, nguyên,...
Cos(x)	Trả về giá trị của Cos(x)
Sin(x)	Trả về giá trị của Sin(x)
Exp(x)	Trả về giá trị của $e^x$
Log(x)	Trả về giá trị của Ln(x)
Log10(x)	Trả về giá trị của log10(x)
Pow(x,y)	Trả về giá trị của $x^y$
Round(x,n)	Làm tròn số thực x với độ chính xác là n chữ số thập phân
Sqrt(x)	Trả về căn bậc hai của x

Để sử dụng được các hàm toán học dưới đây thì trước mỗi hàm ta phải đưa thêm Math. trước mỗi tên hàm cần sử dụng

## 2.2. HÀM XUẤT

- Thư viện lớp C# cung cấp sẵn một lớp tên là Console trong đó có chứa một số phương thức tĩnh để nhập / xuất dữ liệu với màn hình dòng lệnh terminal. Một số phương thức bạn có thể sử dụng ngay trong việc in dữ liệu đó là:

**Console.Write(value)**

- In value ra màn hình, nhưng không xuống dòng mới - value có thể là một số, chữ, chuỗi ...

**Console.WriteLine(value)**

- In value ra màn hình, sau đó xuống dòng

## 2.2. HÀM XUẤT

### **Ví dụ:**

```
int a = 5;
```

```
double x = 7.534;
```

```
string s = "ABC";
```

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

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

## 2.2. HÀM XUẤT

- Ta có dùng kỹ thuật **format string** để in theo định dạng

- **Ví dụ 1:**

```
int a = 123; double b = 567.123;  
Console.WriteLine("Biến a = {0}, biến b = {1}", a, b);  
//Xuất ra: Biến a = 123, biến b = 567.123
```

- **Ví dụ 2:**

```
int a = 123; double b = 567.123;  
Console.WriteLine($"Biến a = {a}, biến b = {b} - tích là {a * b}");  
//Xuất ra: Biến a = 123, biến b = 567.123 - tích là 69756.129
```



## 2.2. HÀM XUẤT

○ **Ví dụ 3:** 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 ra: x = 7.5348,; y = 5.65
```

## 2.2. HÀM XUẤT

### 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

## 2.3. HÀM NHẬP

Hàm `Console.ReadLine()` cho phép nhập dữ liệu cho đến khi nhấn Enter. Hàm này trả về chuỗi mà người dùng nhập vào

### **Ví dụ:**

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

### **Hoặc**

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



## 2.4. CẤU TRÚC ĐIỀU KHIỂN

2.4.1. Rẽ nhánh : if...else

2.4.2. Lựa chọn : switch...case

2.4.3. Lặp : for, while, do...while, **foreach**

**2.4.4** Các cấu trúc khác : goto, break, continue

## 2.4.1. CẤU TRÚC RỄ NHÁNH

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

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



## 2.4.1. CẤU TRÚC RỄ NHÁNH

### **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);
}
```



## 2.4.1. CẤU TRÚC RỄ NHÁNH

```
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.



## 2.4.1. CẤU TRÚC RỄ NHÁNH

### ○ Ví Dụ: 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);
}
```



## 2.4.2. 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



## 2.4.2. CẤU TRÚC LỰA CHỌN

**Ví Dụ:** 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 ( $1 \leq n \leq 5$ ): ");
    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

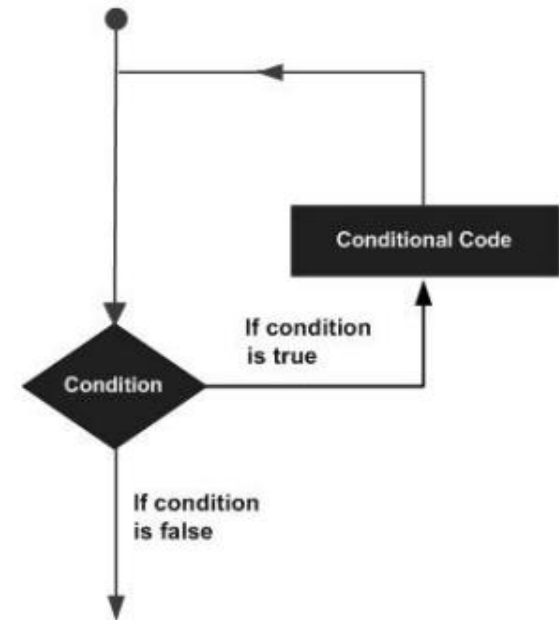
## 2.4.3. CẤU TRÚC LẶP WHILE

while (<biểu thức điều kiện>)

{

lệnh/ khối lệnh;

}



## 2.4.3. CẤU TRÚC LẶP WHILE

### Hoạt động cấu trúc lặp while

- 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. Sau đó quay trở lại bước 2.
  - Ngược lại kết thúc vòng lặp.

## 2.4.3. CẤU TRÚC LẶP WHILE

Ví Dụ: Xuất ra màn hình 10 dòng chữ ABC

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

## Bài tập

- Xây dựng ct cho phép nhập vào 2 số nguyên dương. Tìm và in ra uscln của 2 số đó
- Thuật toán
  1. Nhập vào 2 số a và b
  2. while(a !=b)
    - If (a > b)  
a = a-b;
    - Else  
b=b-a;
  3. In ra Uwsscln là a



## 2.4.4. CẤU TRÚC LẶP FOR

for (<Khởi gán>;<biểu thức ĐK>;<tăng/giảm>)

{

    <khối lệnh>;

}



## 2.4.4. CẤU TRÚC LẶP FOR

Ví Dụ: Xuất ra màn hình 10 dòng chữ ABC

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



## Bài 1

Tính tổng các số từ 1 đến 10

```
Int i, s=0;
```

```
For(i=1;i<=10;i=i+2)
```

```
    s = s + i;
```

```
Console.write(“Tong = {0}”,s);
```

## Bài 2

Xây dựng ct cho phép nhập vào 1 số nguyên N. Tính và in ra N!

## Bài 03

Xây dựng chương trình cho phép nhập vào 1 số nguyên dương N tính;

$$S = \sum_{i=1}^n \left( 1 - \sum_{j=1}^i \frac{1}{j} \right)$$

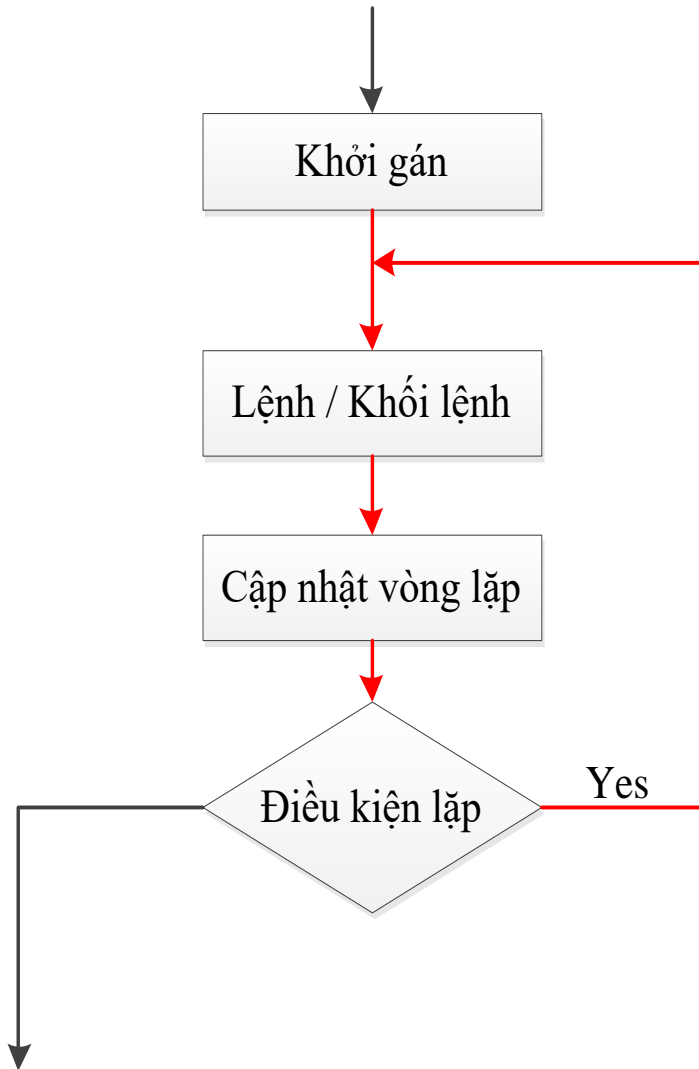
## 2.4.5. CẤU TRÚC LẶP DO...WHILE

**do {**

**< khối lệnh> ;**

**} while (<biểu thức ĐK>);**

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



## 2.4.6. CẤU TRÚC LẶP FOREACH

### ○ Sử dụng cho mảng

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

{

    Khởi lệnh;

}


Xét từng phần tử trong mảng



## 2.4.6. CẤU TRÚC LẶP FOREACH

**Ví Dụ:** 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 US của số nguyên dương.
- Viết chương trình in ra màn hình n dòng dấu \* với n nhập từ bàn phím.

Ví dụ: Nhập  $n=4$

\*

\*   \*

\*   \*   \*

\*   \*   \*   \*





## 2.5. 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.



## 2.5. METHOD - 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.



## 2.5. METHOD - PHƯƠNG THỨC

### Mẫu tổng quát của phương thức

*<phạm vi> <Kiểu dữ liệu> TênPhươngThức([tham số]);*

### Phạm vi

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



## 2.5. METHOD - PHƯƠNG THỨC

Kiểu dữ liệu 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 kiểu dữ liệu tương ứng với kết quả xử lý



## 2.5. METHOD - 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)



## 2.5. METHOD - PHƯƠNG THỨC

**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();  
}
```



## 2.5. METHOD - PHƯƠNG THỨC

### 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

## 2.5. METHOD - PHƯƠNG THỨC

- Ví dụ: Viết chương trình nhập số nguyên dương n và in ra màn hình các ước số của 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();
}
```



## 2.5. METHOD - PHƯƠNG THỨC

### 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*

## 2.5. METHOD - PHƯƠNG THỨC

- Viết chương trình nhập số nguyên dương n và tính. Tính tổng các số từ 1 đến 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();
}
```

## 2.5. METHOD - PHƯƠNG THỨC

### Tham số là tham chiếu

- 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 kiểu dữ liệu của khai báo tham số và trước tên đối số khi gọi phương thức.



## 2.5. METHOD - 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)



## 2.5. METHOD - PHƯƠNG THỨC

**Ví dụ:** 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**

## 2.5. METHOD - PHƯƠNG THỨC

### ○ 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);
}
```

## 2.5. METHOD - PHƯƠNG THỨC

### o 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);
}
```

## 2.5. METHOD - PHƯƠNG THỨC

### ○ 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 của tam giác và in kết quả ra màn hình.



## BÀI TẬP

- 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?



## 2.6. XỬ LÝ NGOẠI LỆ TRONG C#

- Một Exception (ngoại lệ) là một vấn đề xuất hiện trong khi thực thi một chương trình thông thường xảy ra do người dùng nhập liệu không phù hợp
- Xử lý các ngoại lệ này tránh chương trình kết thúc giữa chừng trong quá trình thực thi chương trình

## 2.6. XỬ LÝ NGOẠI LỆ TRONG C#

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);
```



## 2.6. XỬ LÝ NGOẠI LỆ TRONG C#

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.



## 2.6. XỬ LÝ NGOẠI LỆ TRONG C#

- Xử lý ngoại lệ (Exception Handling ) trong C# được xây dựng dựa trên 4 từ khóa là: **try**, **catch**, **finally**, và **throw**.
  - **try**: Một khối try nhận diện một khối lệnh 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.

## 2.6. XỬ LÝ NGOẠI LỆ TRONG C#

- **finally**: Một khối finally được sử dụng để thực thi một tập hợp lệnh đã cho, dù có hay không một exception được phát sinh hoặc không được phát sinh. Ví dụ, nếu bạn mở một file, nó phải được đóng, nếu không sẽ có một exception được tạo ra
- **throw**: phát sinh một exception khi có một vấn đề xuất hiện. Điều này được thực hiện bởi sử dụng từ khóa **throw** trong C#.

## 2.6. XỬ LÝ NGOẠI LỆ TRONG C#

### **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





## 2.6. XỬ LÝ NGOẠI LỆ TRONG C#

### 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



```
try {  
    // các lệnh có thể gây ra ngoại lệ (exception)  
}  
catch ( tên_ngoại_lệ  $e_1$  )  
{  
    // phần code để xử lý lỗi  
}  
catch ( tên_ngoại_lệ  $e_2$  )  
{  
    // phần code để xử lý lỗi  
}  
catch ( tên_ngoại_lệ  $e_n$  )  
{  
    // phần code để xử lý lỗi  
}  
finally  
{  
    // các lệnh được thực thi  
}
```

## 2.6. XỬ LÝ NGOẠI LỆ TRONG C#

### Ví dụ 1:

```
byte k = 0;
```

```
NHAPLAI:
```

```
try {
```

```
    Console.WriteLine("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);
```



## 2.6. XỬ LÝ NGOẠI LỆ TRONG C#

*Ngoại lệ trong catch có thể để trống nếu muốn xử lý lỗi chung chung. Ví dụ:*

```
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;
```

```
}
```

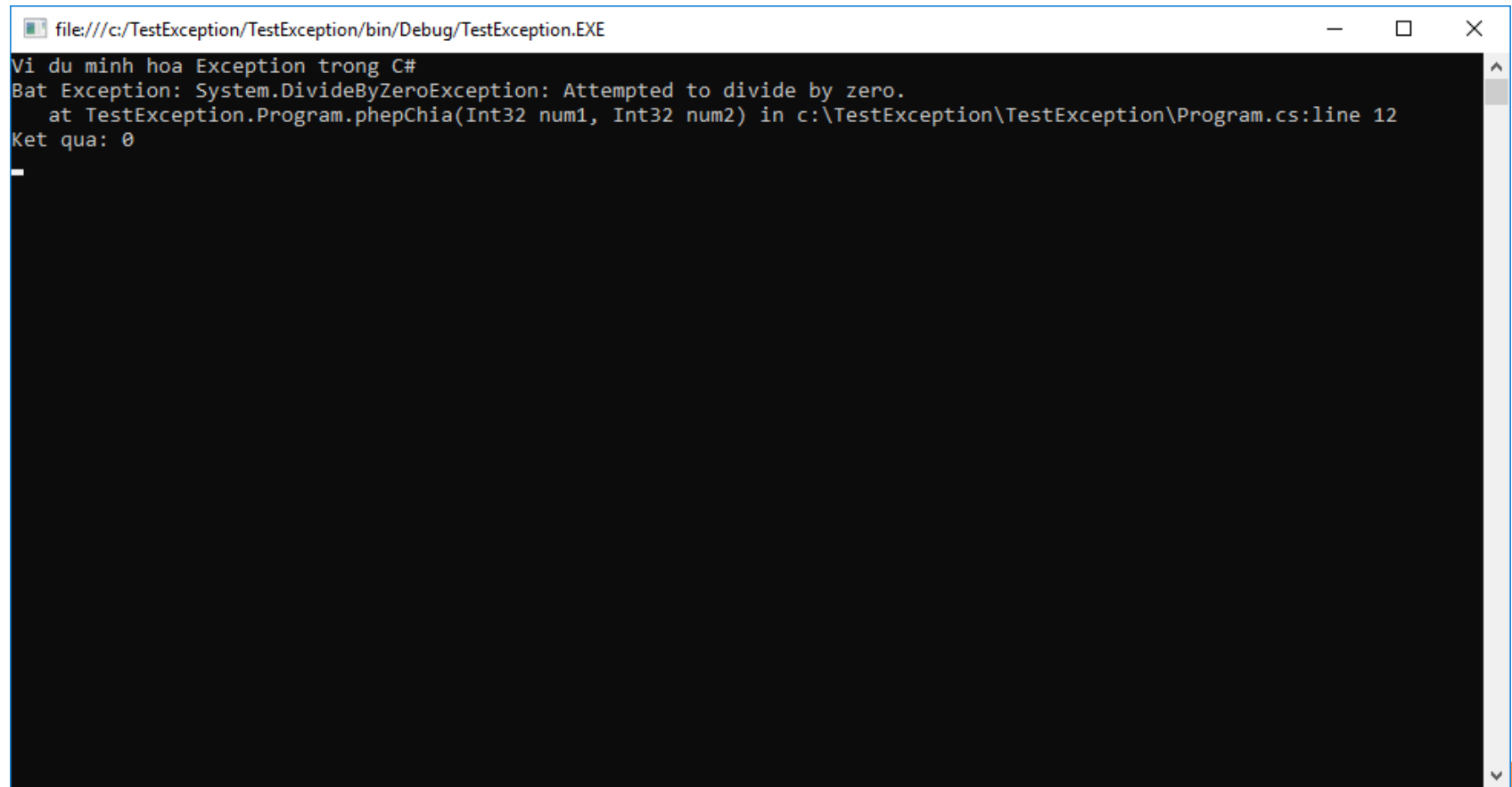
```
Console.WriteLine("Da nhap thanh cong, gia tri k = " + k);
```



## Ví dụ bắt lỗi chia cho 0

```
using System;

namespace TestException
{
    class Program
    {
        float result;
        public void phepChia(int num1, int num2)
        {
            try
            {
                result = (float) num1 / num2;
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine("Bat Exception: {0}", e);
            }
            finally
            {
                Console.WriteLine("Ket qua: {0}", result);
            }
        }
        static void Main(string[] args)
        {
            Program p = new Program();
            Console.WriteLine("Vi du minh hoa Exception trong C#");
            p.phepChia(25, 0);
            Console.ReadKey();
        }
    }
}
```



A screenshot of a Windows command prompt window. The title bar at the top reads "file:///c:/TestException/TestException/bin/Debug/TestException.EXE". The window has standard minimize, maximize, and close buttons. The command prompt area is black with white text. The text displayed is as follows:

```
Vi du minh hoa Exception trong C#  
Bat Exception: System.DivideByZeroException: Attempted to divide by zero.  
    at TestException.Program.phepChia(Int32 num1, Int32 num2) in c:\TestException\TestException\Program.cs:line 12  
Ket qua: 0  
_
```

The text is left-aligned. There is a small white cursor character (an underscore) on the line following "Ket qua: 0". A vertical scrollbar is visible on the right side of the command prompt area.