

# C# cơ bản

TS. Cao Thị Luyện  
[luyenct@utc.edu.vn](mailto:luyenct@utc.edu.vn)  
0912403345



# Nội dung

- Cấu trúc chương trình C#
- Không gian tên
- Kiểu dữ liệu
- Cấu trúc điều khiển
- Mảng

# 1. Cấu trúc chương trình C#

```
//Sử dụng các không gian tên chuẩn
using System;
using System.Text;
//Khai báo không gian tên của ứng dụng
namespace myConsoleApplication
{
    //Vùng bắt đầu khai báo tên các Class
    class Program
    {
        //Vùng khai báo các phương thức
        static void Main(string[] args)
        {
            //Vùng khai báo lệnh
        }
    }
}
```

# Một số khái niệm trong C#

- C# là ngôn ngữ **phân biệt** chữ hoa/thường
- Chú thích
  - Chú thích trên một dòng //
  - Chú thích trên nhiều dòng /\*..... \*/
  - Trình biên dịch bỏ qua chú thích
- Từ khoá (keyword)
  - Có các chức năng đặc biệt không thể thay đổi trong ngôn ngữ
  - Không được dùng làm tên biến, tên lớp hay bất kỳ thứ gì khác
  - Tất cả các từ khoá đều được viết thường

Ví dụ: **class, double, ref, out,..**

# Danh sách các từ khoá trong C#

abstract	throw	private
event	break	uint
new	finally	char
struct	out	foreach
as	true	protected
explicit	byte	ulong
null	fixed	checked
switch	override	goto
base	try	public
extern	case	unchecked
object	float	
this	params	
bool	typeof	
false	catch	
operator	for	

# Nhập dữ liệu qua Console

- Đọc ký tự văn bản từ cửa sổ console
  - `Console.Read()`
  - `Console.ReadLine()`
- Xuất chuỗi kí tự
  - `Console.Write()`
  - `Console.WriteLine()`

# Xuất dữ liệu qua Console

- Xuất chuỗi kí tự
  - Định dạng số:  
`Console.WriteLine("chuỗi định dạng", số)`
  - Trong đó:
    - Chuỗi định dạng: {số thứ tự, số lượng khoảng trống: kí tự định dạng}
    - Ví dụ: {0,8:C} viết kiểu tiền tệ, dành 8 vị trí
  - Một số kí tự định dạng
    - C: Currency
    - D: Decimal
    - E: Scientific
    - F: Fixed point
    - G: General (mặc định)
    - P: Percent

## 2. Không gian tên (namespace)

- Nhóm các tính năng có liên quan của C# vào một loại
- Cho phép dễ dàng tái sử dụng mã nguồn
- Trong thư viện .NET framework có nhiều không gian tên
- Phải tham chiếu tới để sử dụng



# Các namespace cơ bản

Namespace	Description
System	Chứa lớp toán học, chuyển đổi dữ liệu
System.IO	Các lớp cho thao tác Input và Output
System.Net	Các lớp liên quan đến network protocol
System.Collections	Chức các lớp liên quan đến xử lý tập hợp
System.Data	Các lớp của ADO.NET
System.Drawing	Các lớp thực thi chức năng GUI
System.Threading	Các lớp lập trình MultiThread
System.Web	Các lớp liên quan đến HTTP protocol
System.Xml	Các lớp liên quan XML

# Không gian tên

- Sử dụng:
  - Using <tên namespace>;
- Tạo không gian tên:
  - namespace <Tên namespace>
    - {
    - <Định nghĩa lớp A>
    - <Định nghĩa lớp B>
    - ....
    - }

# 3. Kiểu dữ liệu

- Phân loại kiểu dữ liệu
  - Theo phương thức định nghĩa:
    - Có sẵn (Build-in)
    - Người dùng tự định nghĩa (user-defined)
  - Theo cách thức lưu trữ
    - Giá trị (Value)
    - Tham chiếu (Reference)

# Kiểu dữ liệu

- Kiểu dữ liệu có sẵn
  - C# hỗ trợ một số kiểu dữ liệu có sẵn- tương ứng với một kiểu dữ liệu hỗ trợ bởi hệ thống xác nhận ngôn ngữ chung CLS (Common Language System) trong MS.NET
  - Việc ánh xạ các kiểu dữ liệu nguyên thủy của C# đến các kiểu dữ liệu của .Net sẽ đảm bảo các đối tượng được tạo ra trong C# có thể được sử dụng đồng thời với các đối tượng được tạo ra bởi bất cứ ngôn ngữ khác được biên dịch bởi .Net như VB.Net
  - Mỗi kiểu dữ liệu có kích thước xác định

# Kiểu dữ liệu

Kiểu C#	Số byte	Kiểu .NET	Mô tả
byte	1	Byte	Số nguyên dương không dấu từ 0-255
char	2	Char	Ký tự Unicode
bool	1	Boolean	Giá trị logic true/ false
sbyte	1	Sbyte	Số nguyên có dấu ( từ -128 đến 127)
short	2	Int16	Số nguyên có dấu giá trị từ -32768 đến 32767.
ushort	2	UInt16	Số nguyên không dấu 0 – 65.535
int	4	Int32	Số nguyên có dấu –2.147.483.647 và 2.147.483.647
uint	4	UInt32	Số nguyên không dấu 0 – 4.294.967.295
float	4	Single	Kiểu dấu chấm động, giá trị xấp xỉ từ 3,4E-38 đến 3,4E+38, với 7 chữ số có nghĩa..
double	8	Double	Kiểu dấu chấm động có độ chính xác gấp đôi, giá trị xấp xỉ từ 1,7E-308 đến 1,7E+308, với 15,16 chữ số có nghĩa.
decimal	8	Decimal	Có độ chính xác đến 28 con số

# Kiểu dữ liệu

- Kiểu giá trị (value type)
  - Dữ liệu được lưu trữ trên vùng nhớ ngăn xếp (stack)
  - Ví dụ: int, long, float...
- Kiểu tham chiếu (reference type)
  - Địa chỉ lưu trữ trong ngăn xếp (stack)
  - Dữ liệu thực sự được lưu trữ trong vùng nhớ Heap
  - Ví dụ: class, delegate, interface, object, string, dynamic

# Chuyển đổi các kiểu dữ liệu

- Chuyển đổi ngầm định (implicit)
  - Trình biên dịch tự động thực hiện, đảm bảo không bị mất mát dữ liệu
  - Ví dụ: `short x=5;`

`int y=x;`

- Chuyển đổi tường minh (explicit)
  - Sử dụng toán tử chuyển kiểu
  - Sử dụng các tiện ích



# Hàm chuyển đổi các kiểu dữ liệu

**TryParse**(chuỗi cần chuyển, out biến chứa giá trị)

**TryParse** nếu chuyển thành công thì trả về true ngược lại là false

Ví dụ:

```
int a;
```

```
Int32.TryParse("123", out a) //a = 123 hàm cho kết quả là true
```

- **Convert**: Ví dụ:

```
Double d = Convert.ToInt32("123"); //d mang giá trị 123
```

- **Parse**: phương thức chuyển đổi một chuỗi sang một kiểu dữ liệu.

```
int a = Int32.Parse("123"); // a =123
```

```
float b = Float.Parse("20.7"); //b =20.7
```

```
bool c = Boolean.Parse("true"); //c = true
```



# Biến Variable

- Biến (Variable)
  - Một vùng nhớ có định kiểu
  - Có thể gán và thay đổi giá trị
  - Các biến phải được khởi gán trước khi sử dụng

## Cú pháp:

*[ loại ] kiểu\_dữ\_liệu tên\_biến;*

- *loại*: public, private, protected, static
- *kiểu\_dữ\_liệu*: int , long , float....
- *Tên biến*: theo nguyên tắc đặt tên

int tuoi;

float diem;

double tien;

string ten;

# Hằng (Constant)

- Hằng (Constant)
  - Là biến nhưng giá trị không thể thay đổi sau khi khởi gán
  - Cú pháp: **<const> <kiểu> <tên hằng> = <giá trị>;**
  - Ví dụ: `const int a = 10;`
  - Hằng bắt buộc phải được gán giá trị lúc khai báo
  - Không được thay đổi giá trị của hằng

# Kiểu liệt kê

- Là tập hợp các tên hằng có giá trị không thay đổi (thường được gọi là danh sách liệt kê).
- Cú pháp: [thuộc tính] [bổ sung] **enum** <tên liệt kê> [:kiểu cơ sở] {danh sách các thành phần liệt kê}
- Ví dụ:

```
enum NhietDoNuoc
{
    DoDong = 0,
    DoNguoi = 20,
    DoAm = 40,
    DoNong = 60,
    DoSoi = 100,
}
```

# Kiểu chuỗi kí tự (string)

- Khai báo

- Ví dụ: `string st = "hello";`

- Sử dụng

- Sử dụng các toán tử: `==` (bằng), `!=` (khác), `+` (nối chuỗi)

- Ví dụ:

```
string s1 = "hello ";
```

```
string s2 = "world";
```

```
Console.WriteLine(s1 + s2); //"hello world"
```

```
Console.WriteLine(s1 + s2 == "hello world"); //True
```

# Cách đặt tên

- Dùng 2 cách đặt tên là Camel Case hoặc Pascal Case
  - Camel Case: Chữ cái đầu tiên của từ đầu tiên viết thường, các từ còn lại viết hoa chữ đầu
  - Pascal Case: Viết hoa chữ cái đầu tiên của tất cả các từ
- Không đặt tên các biến khai báo cùng tên nhau mà chỉ khác nhau ở chữ hoa và chữ thường
- Không sử dụng tên bắt đầu với ký tự số
- Không sử dụng tên kết thúc với ký tự số

# Cách đặt tên

- Luôn luôn đặt tên có ý nghĩa cụ thể
- Tránh sử dụng từ viết tắt trừ khi quá dài
- Tránh viết tắt những từ nhỏ hơn 5 ký tự
- Tránh đặt tên các biến hoặc hàm trùng với hàm hoặc biến mặc định của Framework

Ví dụ: `string int`, `public system`;

- Không thêm các tiền tố hoặc hậu tố không có nghĩa
- Sử dụng các tiền tố biến boolean bằng “Is”, “Can”, “Has”

# Toán tử trong C#

- Toán tử số học: +, -, \*, /, %, ^, ++, --
- Toán tử quan hệ: ==, !=, >, >=, <, <=
- Toán tử logic: &&, ||, !
- Toán tử gán: =, +=, -=, /=, \*=, %=
- Toán tử 3 ngôi:  
(biểu thức điều kiện) ? (biểu thức 1): (biểu thức 2)

**Ví dụ:** `a = a > b ? a - b : b - a;`

# Toán tử trong C#

Loại toán tử	Toán tử	Tính kết hợp
Một ngôi	- , ++ , --	phải sang trái
Hai ngôi	^	trái sang phải
	*, /, %	
	+, -	
	=	phải sang trái

Thứ tự	Kiểu toán tử
1	Số học
2	So sánh (quan hệ)
3	logic



## 4. Các cấu trúc điều khiển

### ■ Câu lệnh:

- Chương trình C# là một dãy các câu lệnh (statements)
- Mỗi câu lệnh kết thúc bởi dấu “;”
- Các câu lệnh được xử lý tuần tự theo chiều từ trên xuống dưới (trừ trường hợp các lệnh nhảy, rẽ nhánh, lặp...)

### ■ Lệnh nhảy không điều kiện

- Có lời gọi một phương thức
- Sử dụng các lệnh nhảy không điều kiện: goto, break, continue, return, throw

# Lệnh nhảy có điều kiện (rẽ nhánh)

- Rẽ nhánh chỉ được thực hiện khi điều kiện rẽ nhánh là đúng (true)
- Câu lệnh if...else (có thể lồng nhau)
- Câu lệnh chọn: switch...case

# Câu lệnh if...else

- Cú pháp:

```
if (biểu thức điều kiện) <công việc 1>;  
[else <công việc 2>;]
```

- Thực hiện

Nếu **biểu thức điều kiện** là True thì Công việc 1 được thực hiện, ngược lại **công việc 2** được thực hiện.

- Ví dụ:

Nhập một số, cho biết tính chẵn lẻ của số vừa nhập

# Ví dụ câu lệnh if...else

```
using System;
using System.Collections.Generic;
using System.Text;
namespace IfElse
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;
            Console.Write("nhap so n:");
            n = Convert.ToInt32(Console.ReadLine());
            if ((n % 2) == 0)
                Console.WriteLine(n + " la so chan");
            else Console.WriteLine(n + " la so le");
            Console.ReadLine();
        }
    }
}
```

# Câu lệnh switch case

- Cú pháp

**switch** (biểu thức)

```
{  
    case giá_trị_1: {Các lệnh 1; break; }  
    ...  
    case giá_trị_n: {Các lệnh n; break; }  
    [default: Các lệnh n+1;]  
}
```

- Thực hiện

Biểu thức có giá trị 1, lệnh 1 thực hiện...

Mặc định, lệnh n+1 được thực hiện

# Ví dụ câu lệnh switch case

- Nhập vào số nguyên, viết ra dạng chữ của số đó

```
using System;
using System.Collections.Generic;
using System.Text;
namespace SwitchCase
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;
            Console.Write("nhap so n (0<n<4):");
            n = Convert.ToInt32(Console.ReadLine());
            switch (n)
            {
                case 1: { Console.WriteLine(n + ": mot"); break; }
                case 2: { Console.WriteLine(n + ": hai"); break; }
                case 3: { Console.WriteLine(n + ": ba"); break; }
                default:
                    { Console.WriteLine(n + " Không biết đọc");
                      break; }
            }
            Console.ReadLine();
        }
    }
}
```

21/08/2021

# Câu lệnh lặp

- Câu lệnh lặp for
- Câu lệnh lặp while
- Câu lệnh lặp do...while
- Câu lệnh lặp foreach...in

# Câu lệnh lặp for

- Cú pháp

**for** ([**Khởi tạo**]; [**Biểu thức điều kiện**]; [**Bước lặp**])  
    <**Câu lệnh**>;

- Thực hiện

B1. Thực hiện **Khởi tạo**

B2. **Kiểm tra điều kiện**

- Nếu **đúng** thực hiện **Câu lệnh** rồi **Bước lặp** và quay lại B2.
- Nếu **sai** chuyển sang câu lệnh sau **for**



# Ví dụ câu lệnh for

- Ví dụ: In ra màn hình 10 số nguyên dương đầu tiên

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace forStatement
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("10 so nguyen duong dau tien");
            for (int i = 1; i <= 10; i++)
                Console.Write("{0} ", i);
            Console.ReadLine();
        }
    }
}
```

# Câu lệnh lặp while

- Cú pháp

**while** (biểu thức điều kiện) <Công việc>;

- Ví dụ:

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("10 số nguyên dương đầu tiên");
        int i = 1;
        while (i <= 10)
        {
            Console.Write("{0} ", i);
            i++;
        }
        Console.ReadLine();
    }
}
```

# Câu lệnh lặp do...while

- Cú pháp

**do** <công việc> **while** <biểu thức điều kiện>;

- Ví dụ:

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("10 số nguyên dương đầu tiên");
        int i = 1;
        do
        {
            Console.Write("{0} ", i);
            i++;
        }
        while (i <= 10);
        Console.ReadLine();
    }
}
```

# Câu lệnh lặp foreach...in

- Cho phép tạo vòng lặp thông qua một tập hợp hay một mảng

- Cú pháp

**foreach(<kiểu tập hợp> <tên truy cập thành phần> in <tên tập hợp>)**

**{**

**<Khối lệnh>;**

**}**

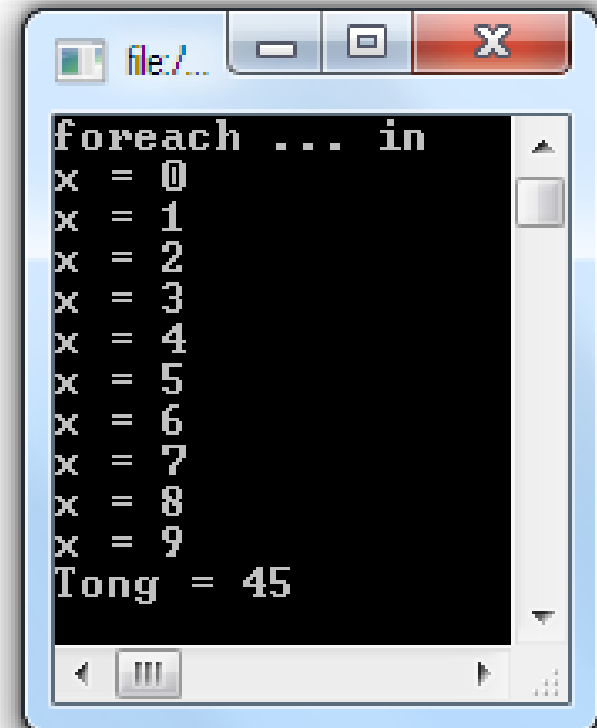
- Thực hiện

Số lần lặp **Khối lệnh** tương ứng bằng số lượng phần tử trong **tập hợp**

# Ví dụ câu lệnh lặp foreach...in

- Tính tổng các phần tử trong mảng

```
static void Main(string[] args)
{
    Console.WriteLine("foreach ... in");
    int[] so = new int[10];
    int tong = 0;
    //gán giá trị cho từng phần tử
    for (int i = 0; i < 10; i++)
    {
        so[i] = i;
    }
    foreach(int x in so)
    {
        Console.WriteLine("x = {0}",x);
        tong+=x;
    }
    Console.WriteLine("Tong = {0}",tong);
}
```



```
file:/...
foreach ... in
x = 0
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
x = 9
Tong = 45
```

## 5. Mảng trong C#

- Mảng là tập hữu hạn các phần tử có cùng kiểu dữ liệu
- Khai báo

**<kiểu dữ liệu>[] <tên mảng>**

int[] so;

float[] diem;

string[] tenlop;

- Tạo thể hiện mảng (dùng **new**)

**[kiểu dữ liệu][ ] [tên mảng] = new [kiểu dữ liệu][tổng số phần tử]**

int [] so = new int[10];

float[] diem = new float[3];

# Mảng trong C#

- Giá trị mặc định: mỗi thành phần sẽ chứa giá trị mặc định của kiểu dữ liệu

**Ví dụ:** `int [] so = new int[5];` tạo một mảng gồm 5 số nguyên, mỗi thành phần giá trị mặc định là 0

- Khởi tạo thành phần của mảng
  - Đặt các giá trị khởi tạo trong cặp dấu { }
  - Ví dụ:

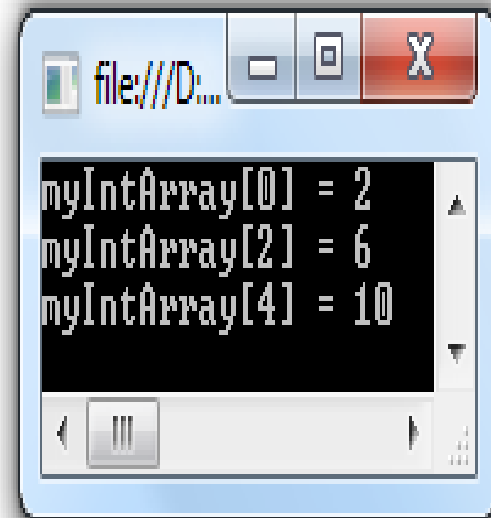
```
int[] myIntArray1 = new int[5]{2,4,6,8,10};
```

```
int[] myIntArray2 = {2,4,6,8,10};
```

# Mảng trong C#

- Truy cập các thành phần trong mảng
  - Dùng toán tử chỉ số [ ]: <tên mảng>[chỉ số]
  - Chỉ số phần tử đầu tiên là 0
  - Ví dụ:

```
int[] myIntArray = new int[5] { 2, 4, 6, 8, 10 };  
int[] myIntArray = { 2, 4, 6, 8, 10 };  
Console.WriteLine("myIntArray[0] = {0}", myIntArray[0]);  
Console.WriteLine("myIntArray[2] = {0}", myIntArray[2]);  
Console.WriteLine("myIntArray[4] = {0}", myIntArray[4]);  
Console.ReadLine();
```





# Mảng trong C#

- Ví dụ: Nhập mảng a gồm N phần tử, in mảng vừa nhập ra màn hình

```
class Program
{
    static void Main(string[] args)
    {
        int n, i;
        int[] a;
        Console.WriteLine("Nhap so luong phan tu: ");
        n = Convert.ToInt32(Console.ReadLine());
        a = new int[n];
        for (i = 0; i < n; i++)
        {
            Console.WriteLine("Nhap phan tu thu {0}: ", i);
            a[i] = convert.Toint32(Console.ReadLine());
        }
        Console.WriteLine("Mang vua nhap la: ");
        for (i = 0; i < n; i++)
        {
            Console.Write("{0} ", a[i]);
        }
        Console.ReadLine();
    }
}
```

# Mảng trong C#

- Ngôn ngữ C# cung cấp cú pháp chuẩn cho việc khai báo những đối tượng Array

Thành viên	Mô tả
BinarySearch()	Phương thức tĩnh public tìm kiếm một mảng một chiều đã sắp thứ tự.
Clear()	Phương thức tĩnh public thiết lập các thành phần của mảng về 0 hay null.
Copy()	Phương thức tĩnh public đã nạp chồng thực hiện sao chép một vùng của mảng vào mảng khác.
CreateInstance()	Phương thức tĩnh public đã nạp chồng tạo một thể hiện mới cho mảng
IndexOf()	Phương thức tĩnh public trả về chỉ mục của thể hiện đầu tiên chứa giá trị trong mảng một chiều
LastIndexOf()	Phương thức tĩnh public trả về chỉ mục của thể hiện cuối cùng của giá trị trong mảng một chiều
Reverse()	Phương thức tĩnh public đảo thứ tự của các thành phần trong mảng một chiều
Sort()	Phương thức tĩnh public sắp xếp giá trị trong mảng một chiều.
IsFixedSize	Thuộc tính public giá trị bool thể hiện mảng có kích thước cố định hay không.

# Mảng trong C#

IsSynchronized	Thuộc tính public giá trị bool thể hiện mảng có hỗ trợ thread-safe
Length	Thuộc tính public chiều dài của mảng
Rank	Thuộc tính public chứa số chiều của mảng
SyncRoot	Thuộc tính public chứa đối tượng dùng để đồng bộ truy cập trong mảng
GetEnumerator()	Phương thức public trả về IEnumerator
GetLength()	Phương thức public trả về kích thước của một chiều cố định trong mảng
GetLowerBound()	Phương thức public trả về cận dưới của chiều xác định trong mảng
GetUpperBound()	Phương thức public trả về cận trên của chiều xác định trong mảng
Initialize()	Khởi tạo tất cả giá trị trong mảng kiểu giá trị bằng cách gọi bộ khởi dụng mặc định của từng giá trị.
SetValue()	Phương thức public thiết lập giá trị cho một thành phần xác định trong mảng.

# Bài tập về nhà

- **Bài 1.** Viết chương trình nhập 2 số thực a và b. Tính tổng, hiệu, tích, thương của 2 số đó.
- **Bài 2.** Viết chương trình giải và biện luận phương trình bậc 2:  $ax^2 + bx + c = 0$ , trong đó các hệ số a, b, c  $\in \mathbb{R}$
- **Bài 3.** Viết chương trình nhập vào từ bàn phím một số nguyên dương N. Kiểm tra xem số đó có phải số nguyên tố hay không? In kết quả ra màn hình.
- **Bài 4.** Viết chương trình nhập vào từ bàn phím số nguyên dương N. Kiểm tra số đó có phải số hoàn hảo hay không? In kết quả ra màn hình. VD: 6 là số hoàn hảo (vì  $6=1+2+3$ )
- **Bài 5.** Viết chương trình nhập vào từ bàn phím dãy gồm N số nguyên. Sắp xếp dãy theo chiều tăng dần và in kết quả ra màn hình.
- **Bài 6.** Viết chương trình nhập vào từ bàn phím dãy điểm. Tính độ dài đường gấp khúc lần lượt đi qua các điểm thứ 1,2,...n..

# Bài tập về nhà

- **Bài 7.** Viết chương trình nhập vào từ bàn phím dãy gồm N số thực
  - - Tính tổng dãy
  - Tính tổng các phần tử nằm trong đoạn  $[0,100]$
  - Tìm giá trị lớn nhất (nhỏ nhất )của dãy
  - Đếm số phần tử nhỏ hơn không hoặc lớn hơn 100

# Thảo luận nhóm

1. Cách nhập xuất trên console của C# ? Cho ví dụ?
  2. Cách chuyển đổi dữ liệu (số sang chuỗi, chuỗi sang số)? Cho ví dụ:
  3. Khai báo mảng trong C#? Cho ví dụ
  4. Tìm hiểu Cấu trúc điều khiển if, for, foreach, do, while để Viết chương trình nhập vào từ bàn phím dãy gồm N số thực
    - Tính tổng dãyTính tổng các phần tử nhỏ hơn không hoặc lớn hơn 100
- Tìm giá trị lớn nhất của dãy
- Đếm số phần tử nằm trong đoạn  $[-10, 10]$
- Sắp xếp dãy theo thứ tự tăng dần