



# Chương 2

## Nền tảng ngôn ngữ C#

- ### Nội dung
1. Kiểu dữ liệu
  2. Biến
  3. Hằng
  4. Biểu thức
  5. Khoảng trắng
  6. Câu lệnh
  7. Toán tử
  8. Định dạng
  9. Một số hàm cơ bản
  10. Câu hỏi

- ### 1. Kiểu dữ liệu
- C# là ngôn ngữ phải khai báo kiểu của mỗi đối tượng khi tạo
  - C# chia thành hai tập hợp kiểu dữ liệu:
    - Kiểu xây dựng sẵn (built-in)
    - Kiểu được người dùng định nghĩa (user-defined)
- Kiểu dữ liệu giá trị (value)

Kiểu dữ liệu tham chiếu (reference)

### Kiểu dữ liệu giá trị

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 và giá trị thập phân, được dùng trong tính toán tài chính, kiểu này đòi hỏi phải có hậu tố "m" hay "M" theo sau giá trị.
long	8	Int64	Kiểu số nguyên có dấu có giá trị trong khoảng : -9.223.370.036.854.775.808 đến 9.223.372.036.854.775.807
ulong	8	UInt64	Số nguyên không dấu từ 0 đến 0xFFFFFFFF

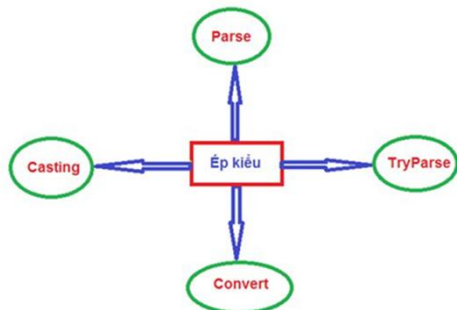
## Kiểu dữ liệu tham chiếu

- **Object:** đây là kiểu dữ liệu cơ sở chứa tất cả các kiểu dữ liệu khác trong C#
- **String:** kiểu dữ liệu chuỗi ký tự
- **Class:** kiểu dữ liệu class
- **Delegate:** kiểu dữ liệu chuyển giao
- **Interface:** kiểu dữ liệu giao tiếp
- **Array:** kiểu dữ liệu mảng

## Cấp phát bộ nhớ

- Các biến của kiểu dữ liệu giá trị
- Các biến của kiểu dữ liệu tham chiếu
- Được lưu trên stack, vùng nhớ này được tham chiếu bởi tên của biến
- Được cấp phát trên heap
- Stack là một cấu trúc dữ liệu lưu trữ thông tin dạng xếp chồng tức là vào sau ra trước (Last In First Out : LIFO)
- Khi một đối tượng được cấp phát trên heap thì địa chỉ của nó được trả về, và địa chỉ này được gán đến một tham chiếu

## Chuyển đổi các kiểu dữ liệu



## Parse

- Phương thức **Parse** là phương thức được sử dụng khá phổ biến khi chúng ta muốn chuyển đổi **một chuỗi** sang **một kiểu dữ liệu tương ứng**
- Mỗi kiểu dữ liệu cơ bản trong C# đều có phương thức **Parse** để chuyển đổi sang kiểu dữ liệu đó



## 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
```

```
byte b = Byte.Parse("1000000000"); //quá giới hạn,  
bool c = Boolean.Parse(null); //tham số là null,  
ArgumentNullException
```

## TryParse

- Giống như Parse, **TryParse** cũng là phương thức được tích hợp sẵn trong các lớp kiểu dữ liệu cơ bản của C#
- Cú pháp của TryParse có phần khác với Parse

```
<kiểu dữ liệu>.TryParse(tham số 1, out tham số 2);
```

```
int a;  
Int32.TryParse("123", out a); //a sẽ mang giá trị 123  
bool b;  
Boolean.TryParse("false", out b); //b sẽ mang giá trị false
```

```
int a;  
Int32.TryParse("hello", out a); //trả về giá trị false, a mang giá trị 0  
bool b;  
Boolean.TryParse("", out b); //trả về giá trị false, b mang giá trị False
```

## Convert

- Lớp **Convert** là một lớp tiện ích trong C# cung cấp cho chúng ta rất nhiều phương thức tính khác nhau để chuyển đổi từ một kiểu dữ liệu này sang kiểu dữ liệu khác
- Tham số mà các phương thức trong **Convert** nhận không nhất thiết phải là chuỗi mà có thể ở nhiều kiểu dữ liệu khác nhau (int, bool, double...)


```
bool a = Convert.ToBoolean("khoaimon");  
  
int b = Convert.ToInt32("123456787654");
```

### Casting (Ép kiểu)

- Ép kiểu là cách chúng ta có thể sử dụng khi muốn chuyển đổi giữa các kiểu dữ liệu có tính chất tương tự nhau (thường là số)
- Có 2 loại:
  - Tường minh
  - Không tường minh

### Casting (Ép kiểu)

```
short x = 10;  
int y = x; // chuyển đổi ngầm định
```



```
short x;  
int y = 100;  
x = y; // Không biên dịch, lỗi !!!
```

short 2 byte  
int 4 byte

```
short x;  
int y = 500;  
x = (short)y; // Ép kiểu tường minh, không báo lỗi
```

### 2. Biến

- Một biến là một vùng lưu trữ với một kiểu dữ liệu
- Biến có thể được gán giá trị và cũng có thể thay đổi giá trị khi thực hiện các lệnh trong chương trình
- Khai báo:  
`<kiểudữliệu> <tênbiến1>, <tênbiến2>, ...;`
- Gán giá trị:  
`<tênbiến> = <giátrị / tênbiếnkhác>;`



## Lưu ý

- Các biến phải được khởi tạo trước khi sử dụng
- Tên biến phân biệt chữ hoa, chữ thường
- Tên biến chỉ có thể bắt đầu với ký tự hoặc dấu \_
- Tên biến không được:
  - Không thể bắt đầu với chữ số
  - Chứa ký tự đặc biệt như \$, #, %, ^, ...
  - Trùng với từ khóa (nhưng nếu muốn đặt trùng tên với từ khóa thì dùng @ ở đằng trước)
  - Tối nghĩa

## Phạm vi của biến

- Toàn cục
- Cục bộ

## 3. Hằng

- Hằng cũng là một biến nhưng giá trị của hằng không thay đổi
- Hằng được phân thành ba loại:
  - Giá trị hằng (literal)
  - Biểu tượng hằng (symbolic constants)
  - Kiểu liệt kê (enumerations)

- Câu lệnh:

`<const> <kiểu dữ liệu> <tên hằng> = <giá trị>;`

```
Const int a = 20;
```

```
Enum <tên liệt kê> : <kiểu dữ liệu>
{
    Danh sách các thành phần liệt kê,
};

enum diem : int
{
    diemmax = 10,
    diemtb = 5,
    diemliet = 0,
};
```

- Mỗi kiểu liệt kê đều có một kiểu dữ liệu cơ sở (int, short, long...)
  - Kiểu liệt kê là một kiểu hình thức do đó bắt buộc phải thực hiện phép chuyển đổi tường minh với các kiểu giá trị nguyên
  - Mỗi thành phần trong kiểu liệt kê tương ứng với một giá trị số nguyên
    - Ta phải khởi tạo
    - Nếu không khởi tạo thì chúng sẽ nhận các giá trị tiếp theo với thành phần đầu tiên là 0
- Kiểu dữ liệu liệt kê không chấp nhận kiểu ký tự và nếu chúng ta bỏ qua phần này thì trình biên dịch sẽ gán giá trị mặc định là kiểu nguyên (int)

## Kiểu chuỗi ký tự

- Kiểu dữ liệu chuỗi khá thân thiện với người lập trình trong bất cứ ngôn ngữ lập trình nào, kiểu dữ liệu **chuỗi** lưu giữ một **mảng** những **ký tự**
- Khai báo một chuỗi : `string chuoi;`
- Hằng chuỗi : `"Xin chao"`

```
string chuoi = "Xin chao"
```

## Định danh

- Định danh là tên do người lập trình chỉ định cho:
  - Các kiểu dữ liệu
  - Các phương thức
  - Biến
  - Hằng
  - Hay đối tượng....
- Cách đặt tên định danh :
  - Bắt đầu với một ký tự chữ cái hay dấu gạch dưới
  - Các ký tự còn lại phải là ký tự chữ cái, chữ số, dấu gạch dưới



## camel notation

Microsoft đề nghị sử dụng cú pháp lạc đà (camel notation) :

- o Tên biến : bắt đầu bởi **ký tự chữ thường**
- o Tên hàm và hầu hết tên các định danh : bắt đầu bởi **ký tự chữ hoa**

## 4. Biểu thức

```
var1 = 24;
```

```
var2 = var1 = 24;
```

```
a = b = c = d = 24;
```

## 5. Khoảng trắng (whitespace)

- C# sẽ bỏ qua tất cả các khoảng trắng trong câu lệnh:

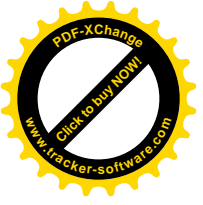
```
var1 = 24; hay var1 = 24 ;
```

- Tuy nhiên lưu ý là khoảng trắng trong một chuỗi sẽ không được bỏ qua :

```
System.WriteLine("Xin chào!");  
System.WriteLine(" Xin chào !");
```

## 6. Cấu trúc câu lệnh (statement)

- Tuần tự
- Phân nhánh
  - o Không điều kiện : goto , break , continue , return , statementthrow
  - o Có điều kiện : if ; switch
- Lặp :
  - o for
  - o while
  - o do ... while



## Statements **break** and **continue**

- goto
- Break
- Continue
- return
- statementthrow

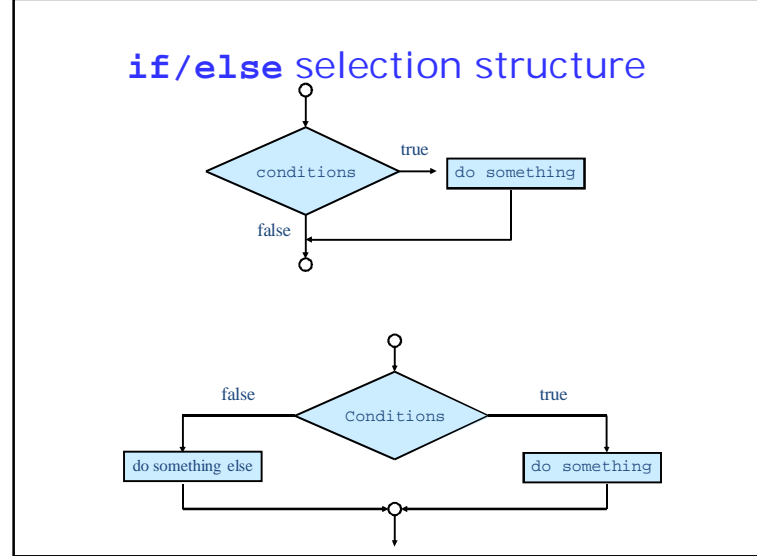
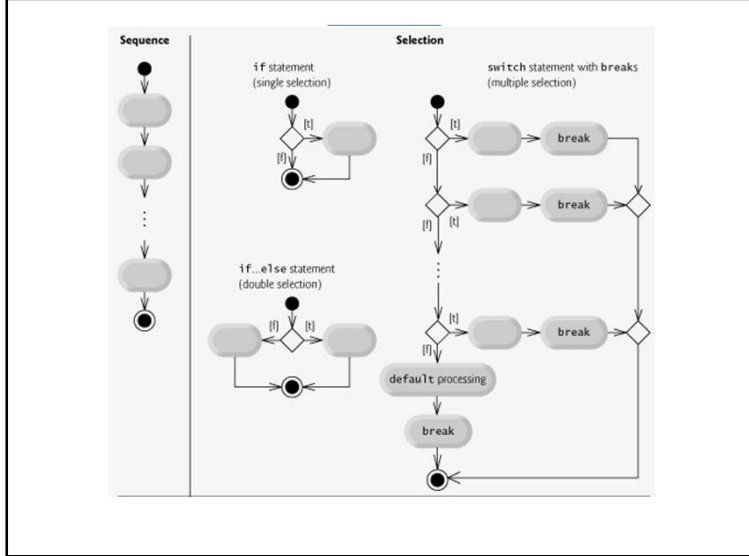
## Goto

- Lệnh nhảy goto là một lệnh nhảy đơn giản, cho phép chương trình nhảy vô điều kiện tới một vị trí trong chương trình thông qua tên nhãn
- Goto giúp chương trình của bạn được linh hoạt hơn nhưng trong nhiều trường hợp nó sẽ làm mất đi cấu trúc thuật toán và gây rối chương trình
- Cách sử dụng lệnh goto:  
Tạo một nhãn  
goto đến nhãn

```
using System;
public class UsingGoto
{
    public static int Main()           i:0
    {                                   i:1
        int i = 0;                     i:2
        lap: // nhãn                   i:3
        Console.WriteLine("i:{0}",i);  i:4
        i++;                           i:5
        if ( i < 10 )                  i:6
            goto lap; // nhảy về nhãn lap i:7
        return 0;                      i:8
    }                                   i:9
}
```

## Continue

- Câu lệnh continue được dùng trong vòng lặp
- Dùng khi bạn muốn khởi động lại một vòng lặp nhưng lại không muốn thi hành phần lệnh còn lại trong vòng lặp, ở một điểm nào đó trong thân vòng lặp



```

if (biểu thức điều kiện)
{
    <lệnh 1>
    <lệnh 2>
    ....
}
[else
{
    <lệnh 1>
    <lệnh 2>
    ...
}]
    
```

```

int s;
s = 3;
s += 1;
if (s > 5)
{
    System.Console.WriteLine(s);
}
else
{
    System.Console.WriteLine(s * 10);
}
System.Console.ReadLine();
    
```



## switch Multiple-Selection Structure

switch (biểu thức điều kiện)

```

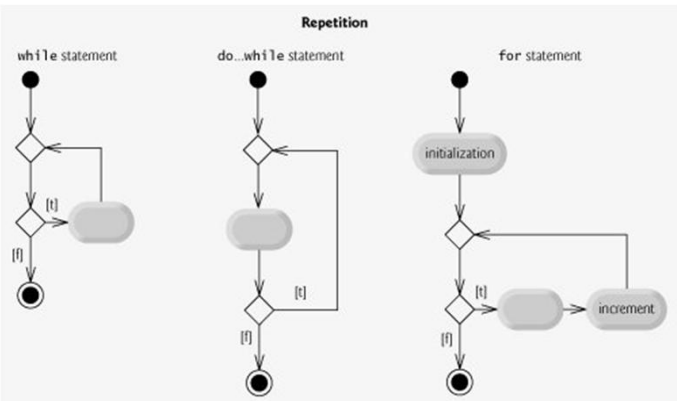
{
    case <giá trị>:
        <Các câu lệnh thực hiện>
        <lệnh nhảy>
    [default:
        <Các câu lệnh thực hiện mặc định>]
}

```

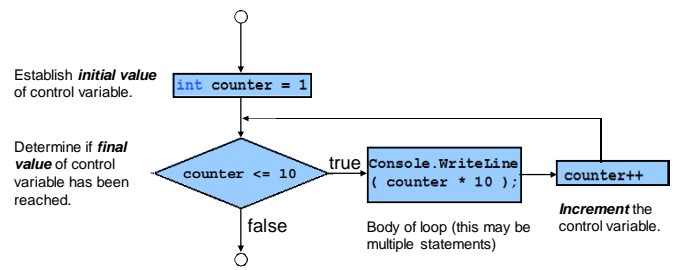
```

int diem;
diem = 7;
switch (diem)
{
    case 3:
    {
        System.Console.WriteLine("Yeu");
        break;
    }
    case 5:
    {
        System.Console.WriteLine("Trung binh");
        break;
    }
    default:
    {
        System.Console.WriteLine("Khong biet");
        break;
    }
}

```



## for Repetition Structure

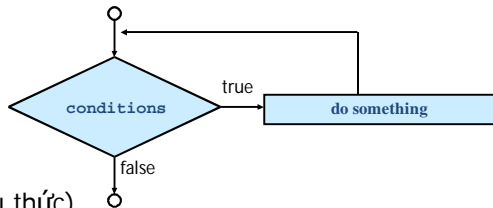


```
for ([ phần khởi tạo ] ; [biểu thức điều kiện]; [bước lặp])  
{  
    <Câu lệnh thực hiện>;  
    <Câu lệnh thực hiện>;  
    <Câu lệnh thực hiện>;  
}
```

```
foreach (<kiểu_tập_hợp> <tên_truy_cập_thành_phần > in <tên_tập_hợp>)  
<Các câu lệnh thực hiện>;
```

```
for (int i = 2; i < 10; i++)  
{  
    for (int j = 1; j < 11; j++)  
    {  
        System.Console.WriteLine(i+"x"+j+"="+i*j+" ");  
    }  
}  
System.Console.ReadLine();
```

### while Repetition Structure



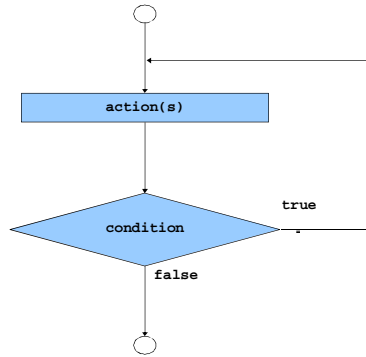
```
while (Biểu thức)  
{  
    <Câu lệnh thực hiện>;  
    <Câu lệnh thực hiện>;  
    <Câu lệnh thực hiện>;  
}
```

```
string pass = "ABCD";  
string chuoi;  
int solan = 0;  
while (solan < 3)  
{  
    System.Console.WriteLine("Nhap pass : ");  
    chuoi = System.Console.ReadLine();  
    if (chuoi == pass)  
    {  
        System.Console.WriteLine("Dung roi");  
        System.Console.ReadLine();  
        solan = 4;  
    }  
    else  
    {  
        System.Console.WriteLine("Sai roi");  
        System.Console.ReadLine();  
        System.Console.Clear();  
        solan += 1;  
    }  
}
```



## do/while Repetition Structure

```
do
{
    <Câu lệnh thực hiện>;
    <Câu lệnh thực hiện>;
    <Câu lệnh thực hiện>;
}
while ( điều kiện )
```



```
using System;
public class UsingDoWhile
{
    public static int Main()
    {
        int i = 11;
        do
        {
            Console.WriteLine("i: {0}",i);
            i++;
        }
        while ( i < 10 )
        return 0;
    }
}
```

## 7. Toán tử

- Toán tử là công cụ dùng để thao tác dữ liệu
- Một toán tử là một ký hiệu dùng để đại diện cho một thao tác cụ thể nào đó được thực hiện trên dữ liệu
- Các loại toán tử:
  - Toán tử gán (=)
  - Toán tử số học
  - Toán tử tăng và giảm (++ ; --)
  - Toán tử quan hệ
  - Toán tử logic
  - Toán tử 3 ngôi

## Toán tử gán

- Toán tử gán hay phép gán làm cho toán hạng bên trái thay đổi giá trị bằng với giá trị của toán hạng bên phải
- Toán tử gán là toán tử hai ngôi
- Đây là toán tử đơn giản nhất thông dụng nhất và cũng dễ sử dụng nhất

```
▪ Ví dụ :
    a = b;
    z = 25;
```

## Toán tử số học

- Phép cộng (+)
- Phép trừ (-)
- Phép nhân (\*)
- Phép chia (/) nguyên
- Phép chia lấy phần dư (%)

## Tăng/giảm (++ / --)

```
var2 = 10;  
var1 = var2++; // Hậu tố
```

var1 10 var2 11

```
var2 = 10;  
var1 = ++var2; // Tiền tố
```

var1 11 var2 11

Toán tử	Ý nghĩa
+=	Cộng thêm giá trị toán hạng bên phải vào giá trị toán hạng bên trái
-=	Toán hạng bên trái được trừ bớt đi một lượng bằng giá trị của toán hạng bên phải
*=	Toán hạng bên trái được nhân với một lượng bằng giá trị của toán hạng bên phải.
/=	Toán hạng bên trái được chia với một lượng bằng giá trị của toán hạng bên phải.
%=	Toán hạng bên trái được chia lấy dư với một lượng bằng giá trị của toán hạng bên phải.

```
luong = 1000000;  
luong = luong + 200000;  
luong = luong * 2;  
luong = luong - 100000;
```

```
luong += 200000;  
luong *= 2;  
luong -= 100000;
```



## Toán tử quan hệ

Tên toán tử	Kí hiệu	Biểu thức so sánh	Kết quả so sánh
So sánh bằng	==	value1 == 100 value1 == 50	true false
Không bằng	!=	value2 != 100 value2 != 90	false true
Lớn hơn	>	value1 > value2 value2 > value1	true false
Lớn hơn hay bằng	>=	value2 >= 50	true
Nhỏ hơn	<	value1 < value2 value2 < value1	false true
Nhỏ hơn hay bằng	<=	value1 <= value2	false

## Toán tử logic

Tên toán tử	Ký hiệu	Biểu thức logic	Giá trị	Logic
and	&&	(x == 3) && (y == 7)	false	Cả hai điều kiện phải đúng
or		(x == 3)    (y == 7)	true	Chỉ cần một điều kiện đúng
not	!	!(x == 3)	true	Biểu thức trong ngoặc phải sai.

## Độ ưu tiên toán tử

STT	Loại toán tử	Toán tử	Thứ tự
1	Phép toán cơ bản	(x) x.y f(x) a[x] x++ x--new typeof sizeof checked unchecked	Trái
2		+ - ! ~ ++x -x (T)x	Trái
3	Phép nhân	* / %	Trái
4	Phép cộng	+ -	Trái
5	Dịch bit	<< >>	Trái
6	Quan hệ	< > <= >= is	Trái
7	So sánh bằng	== !=	Phải
8	Phép toán logic AND	&	Trái
9	Phép toán logic XOR	^	Trái
10	Phép toán logic OR		Trái
11	Điều kiện AND	&&	Trái
12	Điều kiện OR		Trái
13	Điều kiện	?:	Phải
14	Phép gán	= *= /= %= += -= <<= >>= &= ^=  =	Phải

## Toán tử ba ngôi

<Biểu thức điều kiện> ? <Biểu thức thứ 1> : <Biểu thức thứ 2>;

5>6 ? "Sai" : **Dung**

dai = 56;

rong = 45;

rong <= dai ? **Sai roi ban oi** : "Dung roi ban oi";



## 8. 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 học (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

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

## 9. Một số hàm toán học

- Sinh viên tự tìm hiểu

## 10. Câu hỏi

- Sự khác nhau giữa thành phần (Component-Based) và hướng đối tượng (Object-Oriented)?
- Tại sao trong kiểu số không nên khai báo kiểu dữ liệu lớn thay vì dùng kiểu dữ liệu nhỏ hơn?
- Chuyện gì xảy ra nếu ta gán giá trị âm vào biến kiểu không dấu?
- Những ngôn ngữ nào hỗ trợ Common Type System (CTS) trong Common Language Runtime (CLR)?
- Có thể sử dụng chuỗi với câu lệnh switch?
- Những từ theo sau từ nào là từ khóa trong C#: field, cast, as, object, throw, football, do, get, set, basketball.
- Kiểu dữ liệu nào nhỏ nhất có thể lưu trữ được giá trị 45?
- Kết quả của 15%4 là bao nhiêu?
- Số lần tối thiểu các lệnh trong while (do while) được thực hiện?
- Cho biết các lệnh phân nhánh trong C#?