



Bài 2

Nhập xuất dữ liệu
Các cấu trúc điều khiển, lặp

- Nhập xuất dữ liệu trong ứng dụng **Console**
- Chuyển đổi kiểu dữ liệu
- Các cấu trúc lựa chọn
- Các câu trúc lặp
- Các lệnh nhảy

- Ứng dụng Console là loại ứng dụng cho phép người dùng thực hiện các hành động tại dòng lệnh.
- Trong ứng dụng Console có 3 luồng cơ bản để xử lý vào/ ra dữ liệu:
 - Nhập chuẩn (Standard in): dùng để nhận các tham số đầu vào
 - Xuất chuẩn (Standard out): dùng để hiển thị kết quả đầu ra
 - Thông báo lỗi (Standard err): dùng để thông báo lỗi

- Trong C#, lớp Console trong **namespace System** dùng để thực hiện các hoạt động trên console. Để hiển thị ra màn hình ta sử dụng 2 phương thức đầu ra :
 - *Console.Write() : Hiển thị kết quả ra màn hình.*
 - *Console.WriteLine() : Hiển thị kết quả ra màn hình và xuống dòng.*

Ví dụ

```
Console.WriteLine("Xin chào mọi người");  
Console.WriteLine("Toi ten la Nguyen Van A");  
Console.Write("Xin chào mọi người");  
Console.Write("Toi ten la Nguyen Van A");  
// Nhấn Ctrl + F5 và kiểm tra sự khác biệt
```

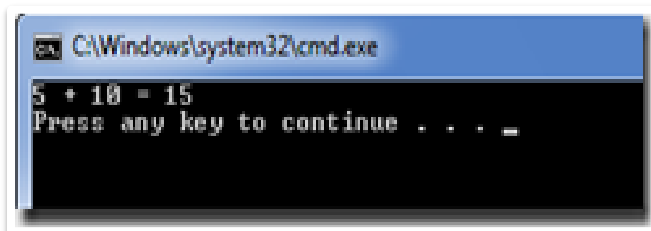
- **Xuất dữ liệu theo vị trí quy định**

`Console.Write("{0},{1},...", biến1, biến2,...);`

Ví dụ

```
int a = 5;  
int b = 10;  
int Tong = a + b;  
Console.WriteLine("{0} + {1} = {2}", a, b, Tong);  
//Ctrl+F5 để xem kết quả:
```

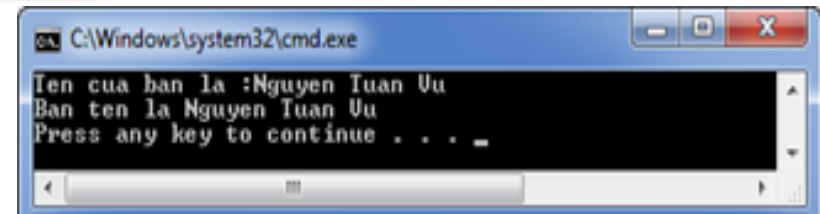
Ta thấy các **vị trí** có cấu trúc như sau
: **{index}**, index bắt đầu từ 0. Như vd trên thì
***a* → {0}; *b* → {1}; *Tong* → {2}**



- Trong C#, để đọc dữ liệu từ bàn phím ta sử dụng nhập chuẩn **Standard in** và thông qua 2 phương thức của lớp **Console**.
 - *Console.Read()* : Đọc một ký tự từ bàn phím.
 - *Console.ReadLine()* : Đọc một chuỗi ký tự từ bàn phím.

Ví dụ

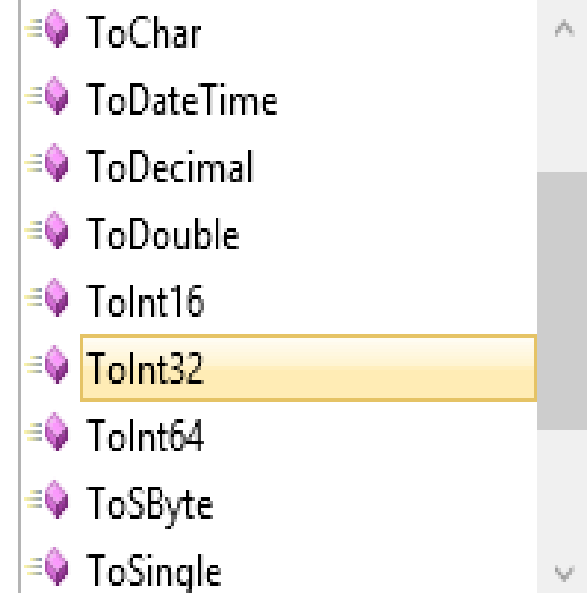
```
string name = "";  
Console.Write("Ten cua ban la :");  
name = Console.ReadLine();  
Console.WriteLine("Ban ten la {0}", name);  
//Ctrl+F5 để nhập và xem kết quả
```



- Trong ứng dụng console, lệnh `Console.ReadLine()` chỉ cho phép nhập vào 1 chuỗi, tuy nhiên bạn có thể chuyển đổi về các kiểu khác nhau như `int`, `double`..., bằng cách sử dụng các phương thức của lớp `Convert`

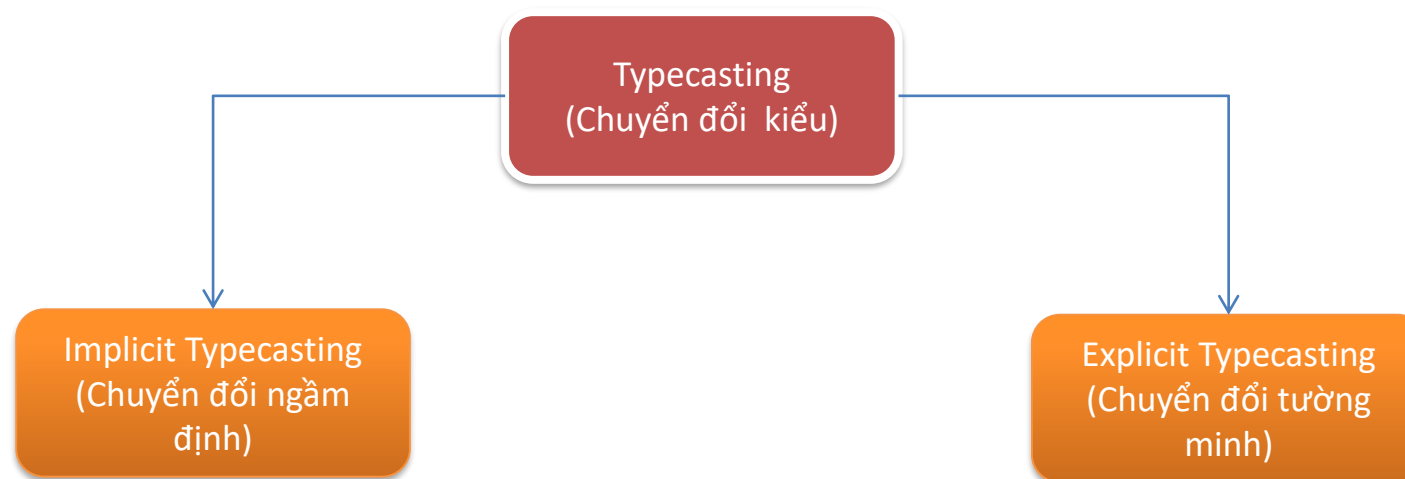
Ví dụ

```
string a = "39";  
string b = "21";  
int a1 = Convert.ToInt32(a);  
int b1 = Convert.ToInt32(b);  
Console.WriteLine(a + b);  
Console.WriteLine(a1 + b1);  
//Ctrl+F5 để xem kết quả:
```



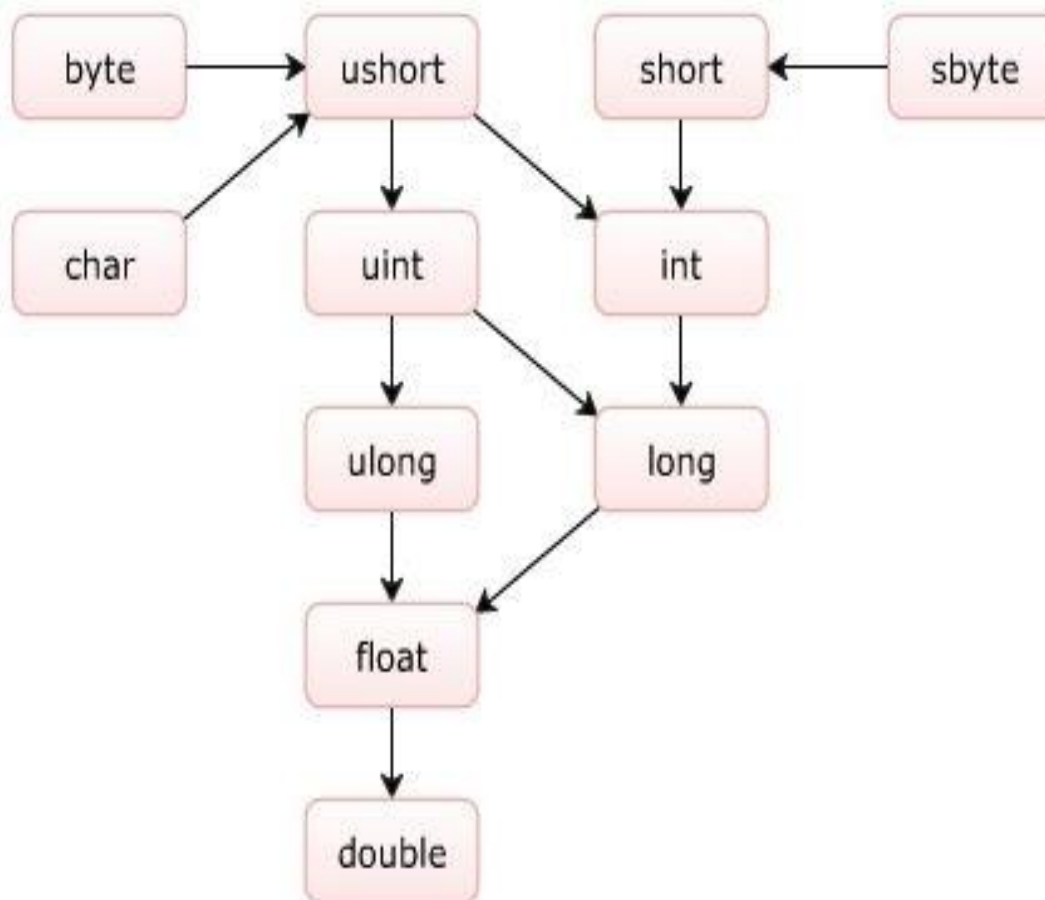
- ToChar
- ToDateTime
- ToDecimal
- ToDouble
- ToInt16
- ToInt32**
- ToInt64
- ToSByte
- ToSingle

- Chuyển đổi kiểu dữ liệu trong C# có 2 loại

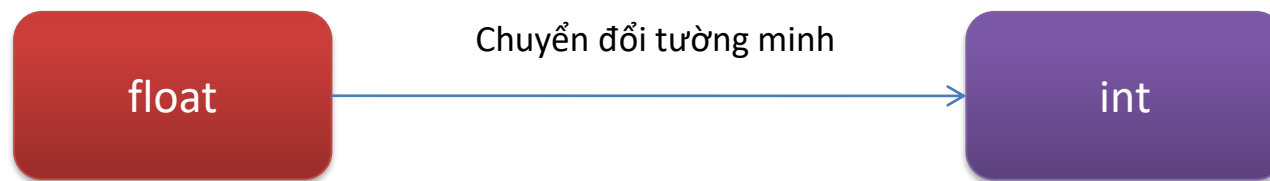


Chuyển đổi ngầm định

- Là cơ chế chuyển đổi tự động bởi trình biên dịch C#, các kiểu dữ liệu có miền giá trị thấp sẽ tự động chuyển về các kiểu có miền giá trị cao hơn



- Là cơ chế chuyển đổi từ các kiểu dữ liệu có miền giá trị cao về các kiểu có miền giá trị thấp hơn



- Phương pháp
 - Dùng các phương thức của lớp Convert
 - Ép kiểu theo cú pháp: **(kiểu_dữ_liệu) giá_trị;**

Ví dụ

```
double a = 10.56;  
int b = (int)a;  
Console.WriteLine(a);  
//Ctrl+F5 để xem kết quả:
```

- Các cấu trúc lựa chọn để thực thi lệnh theo tình huống (Selection constructs): với mục đích làm cho khả năng thực thi của chương trình được linh hoạt chúng ta sẽ sử dụng cấu trúc dạng này.
- Ứng với mỗi tình huống khác nhau sẽ có những cách giải quyết khác nhau, cấu trúc lựa chọn gồm một số dạng như:
 - **if ...; if ... else ...; if...else...if; ...**
 - **switch ...**

(Các cấu trúc thuộc dạng này còn được gọi với cụm từ : “Cấu trúc rẽ nhánh”).

Cấu trúc “if”

if(điều_kiện)

{

Các câu lệnh

}

Ví dụ

```
int a = 10;  
if (a % 2 == 0)  
    Console.WriteLine("{0} là số chẵn", a);
```

Cấu trúc “if...else”

■ Cú pháp

```
if(điều_kiện)
{
    các câu lệnh
}
else
{
    các câu lệnh
}
```

Ví dụ

```
int a = 10;
if (a % 2 == 0)
    Console.WriteLine("{0} là số chẵn", a);
else
    Console.WriteLine("{0} là số lẻ", a);
```

Cấu trúc “if...else...if”

- **Cú pháp**

```
if(điều_kiện_1)
{   các câu lệnh;   }
else if(điều_kiện_2)
{   các câu lệnh;   }
else if(điều_kiện_3)
{   các câu lệnh;   }
...
else
{   các câu lệnh;   }
```

Ví dụ

```
double diem = 5.6;
if (diem < 5)
    Console.WriteLine("Xep loai yeu");
else if (diem < 7)
    Console.WriteLine("Xep loai trung binh");
else if (diem < 9)
    Console.WriteLine("Xep loai kha");
else
    Console.WriteLine("Xep loai gioi");
```

Ví dụ

```
double diem = 5.6;
if (diem >= 0 && diem <= 10)
{
    if (diem < 5)
        Console.WriteLine("Xep loai yeu");
    else if (diem < 7)
        Console.WriteLine("Xep loai trung binh");
    else if (diem < 9)
        Console.WriteLine("Xep loai kha");
    else
        Console.WriteLine("Xep loai gioi");
}
else
    Console.WriteLine("Diem nhap sai");
```

Cấu trúc “switch..case”

■ Cú pháp

```
switch(biểu_thức)
{
    case giá_trị_1:
        các_lệnh;
        break;
    case giá_trị_2:
        các_lệnh;
        break;
    ...
    case giá_trị_n :
        các_lệnh;
        break;
    default:
        các_lệnh;
}
```

Ví dụ

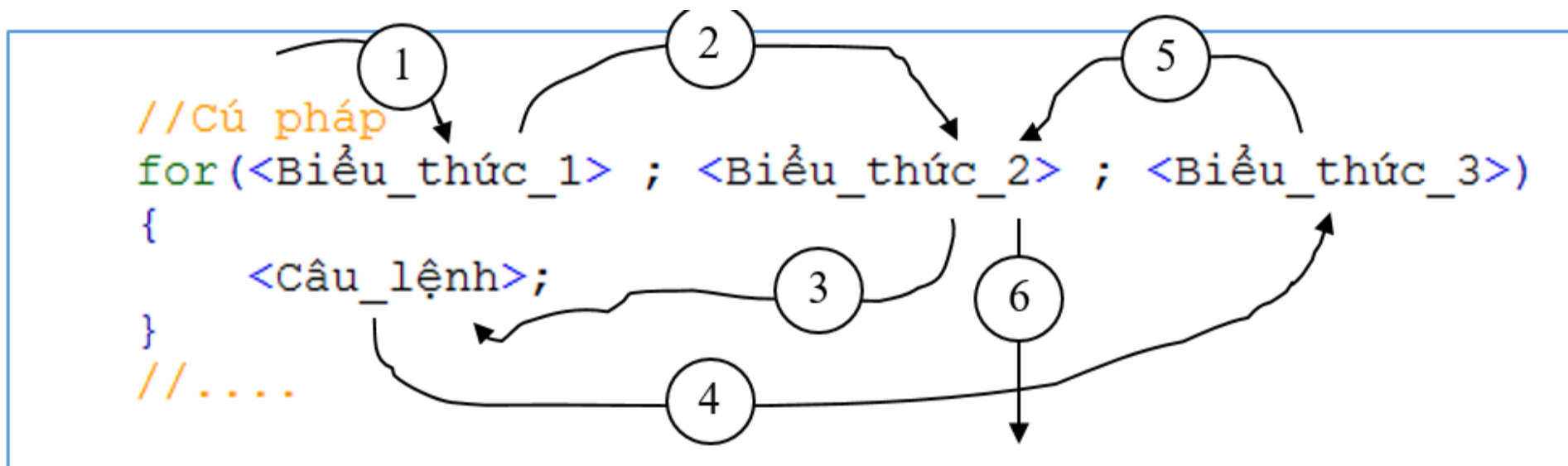
```
int thu = 5;
switch (thu)
{
    case 2:
        Console.WriteLine("Thu 2"); break;
    case 3:
        Console.WriteLine("Thu 3"); break;
    case 4:
        Console.WriteLine("Thu 4"); break;
    case 5:
        Console.WriteLine("Thu 5"); break;
    case 6:
        Console.WriteLine("Thu 6"); break;
    case 7:
        Console.WriteLine("Thu 7"); break;
    case 8:
        Console.WriteLine("Chu nhật"); break;
    default:
        Console.WriteLine("Sai thu"); break;
}
```


- Các cấu trúc lặp để thực thi các lệnh theo chu kỳ.
- Khi có 1 hoặc nhiều lệnh cần phải thực thi nhiều lần, chúng ta sẽ sử dụng đến cấu trúc lặp trong chương trình của mình.
- Thông thường cấu trúc lặp trong các ngôn ngữ lập trình hiện đại chia làm 4 dạng:



Cấu trúc “for”

■ Cú pháp



Ví dụ

```
for (int i = 1; i <= 10; i++)  
{  
    Console.WriteLine(i);  
}
```

Cấu trúc “do...while”

■ Cú pháp

do

{

một hoặc nhiều câu lệnh;

} while(điều_kiện);

Ví dụ

```
int a = 0;
do
{
    Console.WriteLine("Nhập 1 số trong khoảng 10-100:");
    a = Convert.ToInt32(Console.ReadLine());
} while (a < 10 || a > 100);
Console.WriteLine("\n Số vừa nhập là a={0}", a);
```

Cấu trúc “while”

- **Cú pháp**

`while(điều_kiện)`

`{`

một hoặc nhiều câu lệnh

`}`

Ví dụ

```
int i = 10;
while (i > 0)
{
    Console.WriteLine(i);
    i--;
}
```

Cấu trúc “foreach”

- **Cú pháp**

```
foreach (<kiểu_dl> <biến> in <tập hợp>)  
{  
  
    một hoặc nhiều câu lệnh  
  
}
```

Ví dụ

```
string[] names = { "Lan", "Dung", "Diep", "Hang", "Anh", "Tu" };  
foreach (string name in names)  
{  
    Console.WriteLine(name);  
}
```

Vòng lặp lồng nhau

- Các vòng lặp được đặt lồng vào nhau theo từng mục đích sử dụng

Ví dụ

```
for (int i = 2; i < 10; i++)  
{  
    Console.WriteLine("Bang cuu chuong so {0}", i);  
    for (int j = 1; j <= 10; j++)  
    {  
        Console.WriteLine("{0}*{1}={2}", i, j, i * j);  
    }  
}
```

- Các câu lệnh nhảy được sử dụng để chuyển điều khiển chương trình từ một vị trí này tới một vị trí khác
- C# hỗ trợ 4 câu lệnh nhảy



- Câu lệnh break được sử dụng trong các cấu trúc lặp và cấu trúc switch, khi gặp lệnh **break**, điều khiển chương trình sẽ thoát ra khỏi cấu trúc đó và thực hiện các lệnh tiếp theo

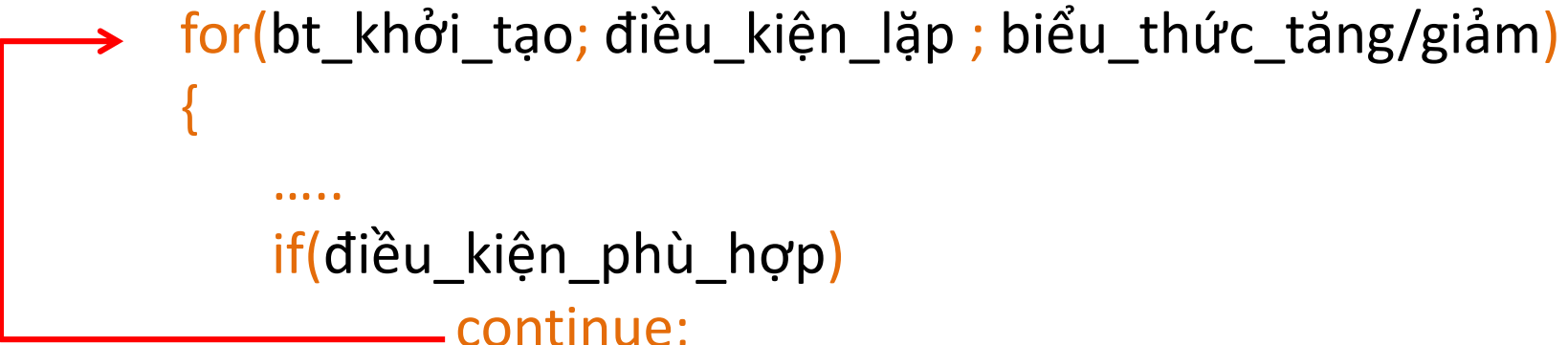
```
for(bt_khởi_tạo; điều_kiện_lặp ; biểu_thức_tăng/giảm)
{
    ....
    if(điều_kiện_thỏa_mãn)
        break;
    ....
}
```



Thoát khỏi vòng lặp

Câu lệnh “continue”

- Câu lệnh continue được sử dụng trong các cấu trúc lặp, khi gặp lệnh này máy sẽ kết thúc lần lặp hiện tại và chuyển điều khiển chương trình tới vị trí bắt đầu của vòng lặp để thực hiện lần tiếp theo



```
for(bt_khởi_tạo; điều_kiện_lặp ; biểu_thức_tăng/giảm)
{
    .....
    if(điều_kiện_phù_hợp)
        continue;
    ....
}
```

Câu lệnh “go to”

- Câu lệnh này chuyển điều khiển thực thi chương trình tới vị trí được gán nhãn:

```
if(điều_kiện_đúng)
```

```
{
```

```
    go to HienThi;
```

```
}
```

```
....
```

```
HienThi:
```

```
    Các lệnh
```



Câu lệnh “return”

- Câu lệnh return sẽ trả về giá trị của một biểu thức, nó thường sử dụng trong các phương thức để trả về giá trị khi phương thức được thực thi
- Cú pháp:
return (expression);

Ví dụ

```
static int Add(int a, int b)
{
    return (a + b);
}
```

Hỏi Đáp

