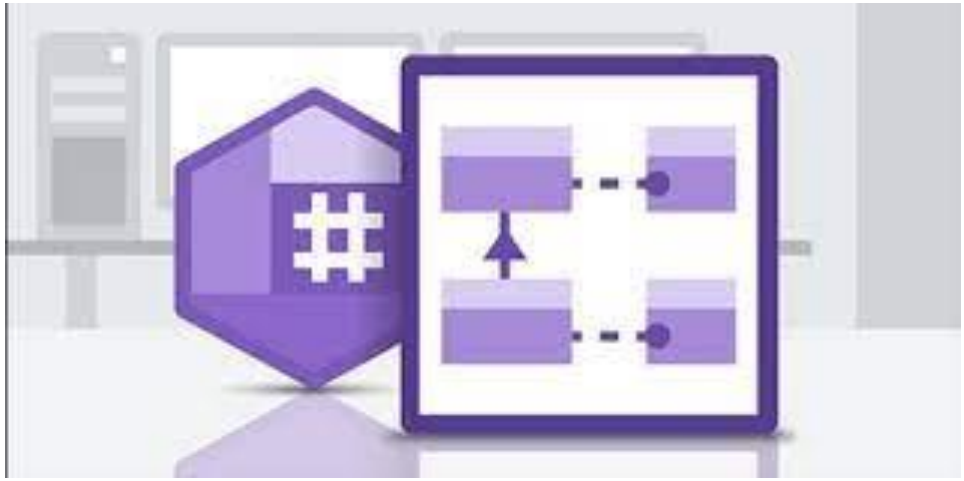
C# 3 – Câu lệnh và Toán tử

Giảng viên: **ThS. Lê Thiện Nhật Quang**

Email: quangltn.dotnet.vn@gmail.com

Website: http://dotnet.edu.vn

Điện thoại: **0868.917.786**



MỤC TIÊU

- Tìm hiểu các câu lệnh và biểu thức
- Giải thích các loại toán tử
- Giải thích cách thức thực thi chuyển đổi dữ liệu trong C#

1.1. CÂU LỆNH - STATEMENT

Là một đơn vị thực thi cơ bản của một chương trình.

Một chương trình sẽ bao gồm nhiều câu lệnh.

Ví dụ:

int age = 21;

int marks = 90;

Trong ví dụ trên, cả hai dòng đều là câu lệnh.

Có nhiều loại câu lệnh khác nhau. Tập trung vào hai loại:

1. Declaration Statement/ Câu lệnh khai báo

2. Expression Statement/ Câu lệnh biểu thức

1.1. CÂU LỆNH (2)

```
internal class Circle
{
    0 references
    static void Main(string[] args)
    {
        const float _pi = 3.14F;
        float radius = 5;
        float area = _pi * radius * radius;
        Console.WriteLine("Area of the circle is " + area);
    }
}
```

block

1.2. SỬ DỤNG CÁC CÂU LỆNH

Các câu lệnh được sử dụng để chỉ định đầu vào, xử lý và các tác vụ đầu ra của chương trình.
Các câu lệnh gồm:

- Kiểu dữ liệu – Data type
- Biến - Variable
- Toán tử - Operators
- Hằng - Contants
- Literals
- Từ khóa – Keywords
- Các ký tự lệnh – Escape sequence characters

Các câu lệnh giúp xây dựng luồng logic trong chương trình.:

- Khởi tạo biến và đối tượng
- Cho đầu vào
- Gọi phương thức của class
- Hiển thị kết quả đầu ra

```
double area = 3.1452 * radius * radius;
```

```
{
    int side = 10;
    int height = 5;
    double area = 0.5 * side * height;
    Console.WriteLine("Area: " , area);
}
```

```
{
    int side = 5;
    int height = 10;
    double area;
    {
        area = 0.5 * side * height;
    }
    Console.WriteLine(area);
}
```

1.3. CÁC LOẠI CÂU LỆNH

Tương tự như các câu lệnh của C, C++; Các câu lệnh C# phân thành 7 loại:

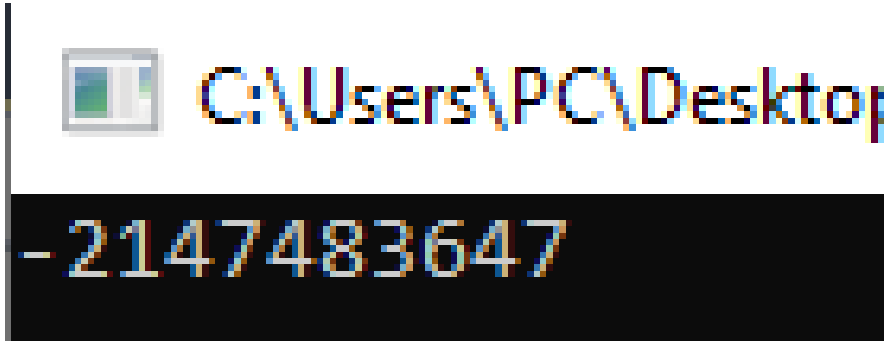
- Câu lệnh lựa chọn - Selection Statements
- Câu lệnh lặp - Iteration Statements
- Câu lệnh nhảy - Jump Statements
- Câu lệnh xử lý ngoại lệ - Exception Handling Statements
- Checked and Unchecked Statements
- Câu lệnh cố định - Fixed Statement
- Câu lệnh khóa - Lock Statement

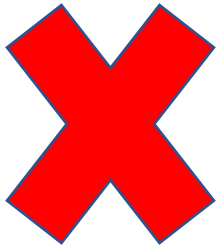
1.4. CÂU LỆNH CHECKED VÀ UNCHECKED

Trong quá trình thực thi, thông thường ứng với các biến thuộc các kiểu dữ liệu được định nghĩa trong chương trình chỉ có thể chứa được giá trị trong khoảng cho phép tương ứng với kiểu dữ liệu mà nó thuộc về.

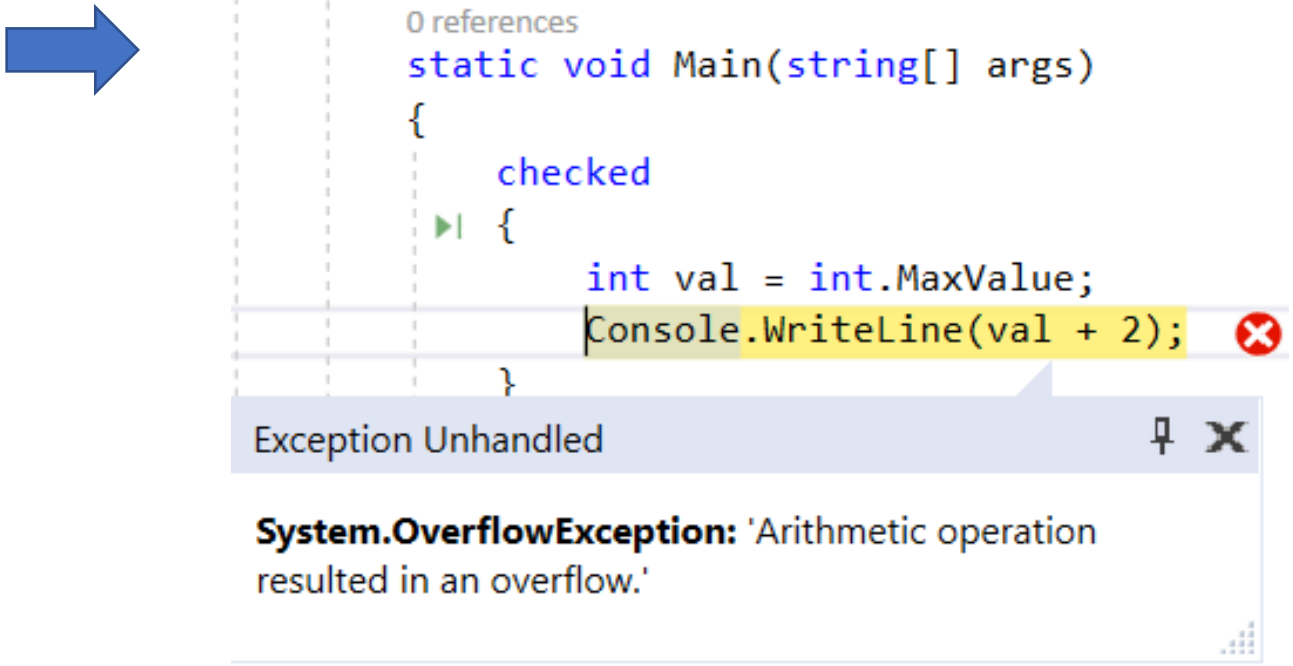
Ví dụ 1 biến kiểu byte chỉ có thể chứa được giá trị số nguyên dương lớn nhất là 255, tuy nhiên trong những hoàn cảnh đặc biệt nào đó, giá trị gán cho biến của kiểu này có nguy cơ lớn hơn 255 dẫn đến hiện tượng tràn số trong bộ nhớ (overflow). Để giải quyết tình trạng này, C# cung cấp cấu trúc checked … unchecked


```
/// <summary>
/// Quang Lê
/// </summary>
/// <param name="args"></param>
0 references
static void Main(string[] args)
{
    int val = int.MaxValue;
    Console.WriteLine(val + 2);
    Console.ReadLine();
}
```





1.4. CÂU LỆNH CHECKED VÀ UNCHECKED (2)



Từ khóa Unchecked bỏ qua các ngoại lệ số học. Nó không kiểm tra rõ ràng và tạo ra kết quả có thể bị cắt ngắn hoặc sai.

1.5. BIỂU THỨC

Những câu lệnh mà thực hiện việc đánh giá một giá trị gọi là biểu thức. Một phép gán một giá trị cho một biến cũng là một biểu thức:

var1 = 24;

Trong câu lệnh trên phép đánh giá hay định lượng chính là phép gán có giá trị là 24 cho biến var1. Lưu ý là toán tử gán ('=') không phải là toán tử so sánh. Do vậy khi sử dụng toán tử này thì biến bên trái sẽ nhận giá trị của phần bên phải. Các toán tử của ngôn ngữ C# như phép so sánh hay phép gán sẽ được trình bày chi tiết trong mục toán tử của chương này.

Do var1 = 24 là một biểu thức được định giá trị là 24 nên biểu thức này có thể được xem như phần bên phải của một biểu thức gán khác:

var2 = var1 = 24;

Lệnh này sẽ được thực hiện từ bên phải sang khi đó biến var1 sẽ nhận được giá trị là 24 và tiếp sau đó thì var2 cũng được nhận giá trị là 24. Do vậy cả hai biến đều cùng nhận một giá trị là 24. Có thể dùng lệnh trên để khởi tạo nhiều biến có cùng một giá trị như:

a = b = c = d = 24;

1.6. SỰ KHÁC NHAU CÂU LỆNH VÀ BIỂU THỨC

Câu lệnh	Biểu thức
Không nhất thiết phải trả về giá trị. Ví dụ, hãy xem xét câu lệnh sau: int oddNum = 5; Câu lệnh chỉ lưu giá trị 5 trong biến oddNum.	Luôn đánh giá một giá trị. Ví dụ, hãy xem xét biểu thức sau: 100 * (25 * 10) Biểu thức ước tính giá trị 2500.
Các câu lệnh được thực thi bởi trình biên dịch.	Biểu thức là một phần của câu lệnh và được đánh giá bởi trình biên dịch.

2.1. TOÁN TỬ (OPERATOR)

Là một công cụ để thao tác với dữ liệu.
Một toán tử là một ký hiệu dùng để đại diện cho một thao tác cụ thể được thực hiện trên dữ liệu.
Toán tử dùng để thực hiện các hoạt động trên biến. Có các loại toán tử sau:

- Toán tử toán học (Arithmetic Operators): +, -, *, /, %, ++, --
- Toán tử quan hệ (Relational Operators): ==, !=, >, <, <=, >=
- Toán tử luận lý (Logical Operators): &, |, ^, >>, <<
- Toán tử điều kiện: &&, ||, !
- Toán tử tăng, giảm
- Toán tử gán bằng (Assignment Operators): =, +=, -=, *=, /=

2.2. TOÁN TỬ TOÁN HỌC

Giả sử biến a có giá trị là 10 biến b có giá trị là 9

Toán tử	Mô tả	Ví dụ
+	Thực hiện **cộng** hai toán hạng	**a + b** kết quả bằng 19
-	Thực hiện **trừ** hai toán hạng	**a - b** kết quả bằng 1
*	Thực hiện **nhân** hai toán hạng	**a * b** kết quả bằng 90
/	Thực hiện **chia lấy phần nguyên** hai toán hạng nếu 2 toán hạng là số nguyên. Ngược lại thì thực hiện chia bình thường	**a / b** kết quả bằng 1
%	Thực hiện **chia lấy dư**	**a%b** kết quả bằng 1

2.2. TOÁN TỬ TOÁN HỌC (2)


```
int valueOne = 10;
int valueTwo = 2;
int add = valueOne + valueTwo;
int sub = valueOne - valueTwo;
int mult = ValueOne * valueTwo;
int div = valueOne / valueTwo;
int modu = valueOne % valueTwo;
Console.WriteLine("Addition " + add );
Console.WriteLine("Subtraction " + sub);
Console.WriteLine("Multiplication " + mult);
Console.WriteLine("Division " + div);
Console.WriteLine("Remainder " + modu);
```

2.3. TOÁN TỬ QUAN HỆ

Giả sử biến a có giá trị bằng 10 và biến b có giá trị bằng 9:

Toán tử	Mô tả	Ví dụ
==	So sánh 2 toán hạng có bằng nhau hay không. Nếu **bằng** thì trả về true nếu **không bằng** thì trả về false	a == b sẽ trả về false
!=	So sánh 2 toán hạng có bằng nhau hay không. Nếu **không bằng** thì trả về true nếu **bằng** thì trả về false	a != b sẽ trả về true
>	So sánh 2 toán hạng bên trái có **lớn hơn** toán hạng bên phải hay không. Nếu **lớn hơn** thì trả về true nếu **không lớn hơn** thì trả về false	a > b sẽ trả về true
<	So sánh 2 toán hạng bên trái có **nhỏ hơn** toán hạng bên phải hay không. Nếu **nhỏ hơn** thì trả về true nếu **không nhỏ hơn** thì trả về false	a < b sẽ trả về false
>=	So sánh 2 toán hạng bên trái **có lớn hơn hoặc bằng** toán hạng bên phải hay không. Nếu **lớn hơn hoặc bằng** thì trả về true nếu **nhỏ hơn** thì trả về false	a >= b sẽ trả về true
<=	So sánh 2 toán hạng có **nhỏ hơn hoặc bằng** hay không. Nếu **nhỏ hơn hoặc bằng** thì trả về true nếu **lớn hơn** thì trả về false	a <= b sẽ trả về false

2.3. TOÁN TỬ QUAN HỆ (2)

Lưu ý:

1.Các toán tử quan hệ này chỉ áp dụng cho số hoặc ký tự.

2.Hai toán hạng hai bên phải cùng loại (cùng là số hoặc cùng là ký tự).

3.Bản chất của việc so sánh 2 ký tự với nhau là so sánh mã ASCII của các ký tự đó.

•**Ví dụ:** so sánh '**A**' và '**B**' bản chất là so sánh số **65** với **66**.

4.Không nên sử dụng các toán tử trên để so sánh các chuỗi với nhau vì bản chất việc so sánh chuỗi là so sánh từng ký tự tương ứng với nhau mà so sánh ký tự là so sánh mã ASCII của ký tự đó như vậy ký tự '**K**' sẽ khác ký tự '**k**'. Để so sánh hai chuỗi người ta thường dùng hàm so sánh chuỗi đã được hỗ trợ sẵn (sẽ tìm hiểu ở những bài tiếp theo).

2.3. TOÁN TỬ QUAN HỆ (3)

```
int leftVal = 50;
int rightVal = 100;
Console.WriteLine("Equal: " + (leftVal == rightVal));
Console.WriteLine("Not Equal: " + (leftVal != rightVal));
Console.WriteLine("Greater: " + (leftVal > rightVal));
Console.WriteLine("Lesser: " + (leftVal < rightVal));
Console.WriteLine("Greater or Equal: " + (leftVal >= rightVal));
Console.WriteLine("Lesser or Equal: " + (leftVal <= rightVal));
```

2.4. TOÁN TỬ LUẬN LÝ – BOOLEAN LOGIC

Toán tử	Mô tả	Ví dụ
&	Trả về là true nếu cả 2 biểu thức có kết quả là true	(percent >= 75) & (percent <=100)
|	Trả về là true nếu 1 trong các biểu thức có kết quả là true	(choice == 'Y') | (choice =='y')
^	Trả về là true nếu chỉ có 1 biểu thức có kết qủa là true. Nếu cả 2 biểu thức có kết quả là true thì toán hạng trả về là false	(choice == 'Q') ^ (choice =='q')

2.4. TOÁN TỬ LUẬN LÝ – BOOLEAN LOGIC (2)

```
if ((quantity > 2000) | (price < 10.5))
{
Console.WriteLine ("You can buy more goods at a lower price");
}

if ((quantity == 2000) & (price == 10.5))
{
Console.WriteLine ("The goods are correctly priced");
}

if ((quantity == 2000) ^ (price == 10.5))
{
Console.WriteLine ("You have to compromise between quantity and
price");
}
```

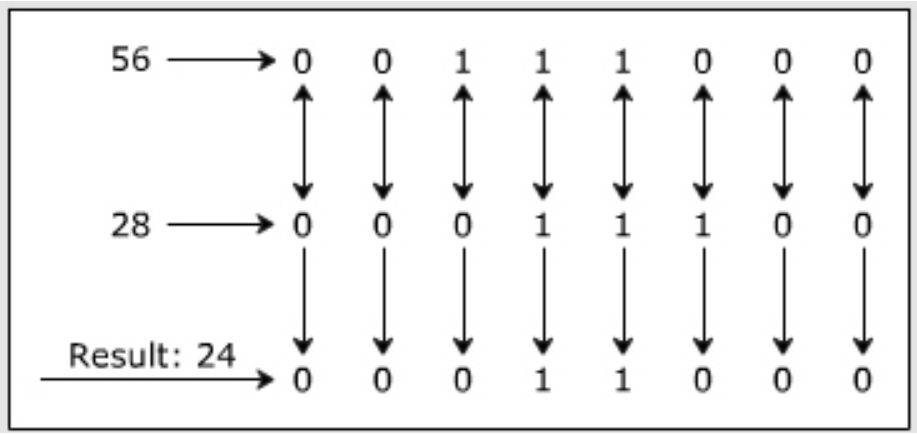
2.5. TOÁN TỬ LUẬN LÝ - BITWISE

Giả sử a có giá trị bằng 10 và b có giá trị bằng 9. Giá trị biến a đổi ra nhị phân là 1010 và giá trị biến b đổi ra nhị phân là 1001

Toán tử	Mô tả	Ví dụ
&	Sao chép bit 1 tới kết quả nếu nó **tồn tại trong cả hai** toán hạng tại vị trí tương ứng, ngược lại thì bit kết quả bằng 0	**a&b** sẽ cho kết quả là 1000 tương đương với số 8 trong hệ thập phân
|	Sao chép bit 1 tới kết quả nếu nó **tồn tại ở một trong hai** toán hạng tại vị trí tương ứng, ngược lại thì bit kết quả bằng 0	**a|b** sẽ cho kết quả 1011 tương đương với số 11 trong hệ thập phân
^	Sao chép bit 1 tới kết quả nếu **nó chỉ tồn tại ở một** toán hạng tại vị trí tương ứng, ngược lại thì bit kết quả bằng 0	**a^b** sẽ cho kết quả 0011 tương đương với số 3 trong hệ thập phân
~	Dùng để **đảo bit** 0 thành 1 và ngược lại 1 thành 0	**~a** sẽ cho kết quả 0101
<<	Dịch trái n bit. Giá trị toán hạng bên trái sẽ được **dịch trái** n bit với n được xác định bởi toán hạng bên phải	**a<<2** sẽ cho kết quả 101000
>>	Dịch phải n bit. Giá trị toán hạng bên trái sẽ được **dịch phải** n bit với n được xác định bởi toán hạng bên phải	**a>>2** sẽ cho kết quả 0010

2.5. TOÁN TỬ LUẬN LÝ – BITWISE (2)

```
result = 56 & 28; //(56 = 00111000 and 28 = 00011100)
Console.WriteLine(result);
```




```
result = 56 | 28; //(56 = 00111000 or 28 = 00011100)
Console.WriteLine(result);
```

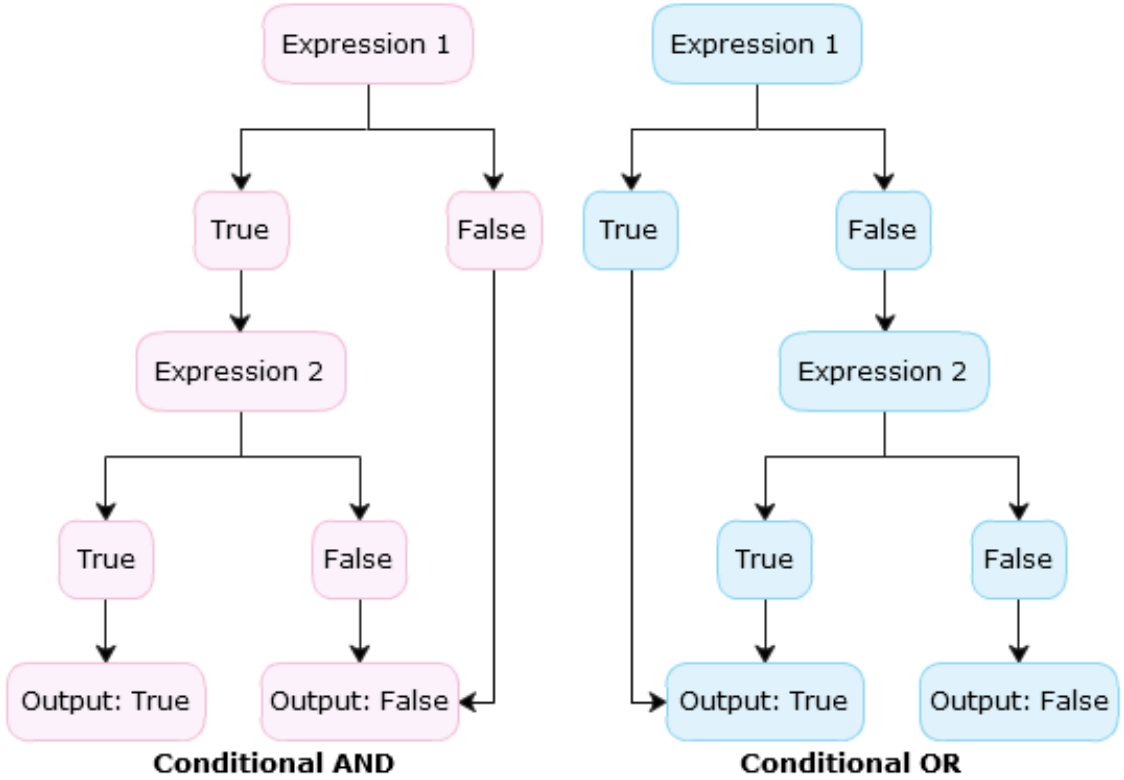
```
result = 56 ^ 28; //(56 = 00111000 xor 28 = 00011100)
Console.WriteLine(result);
```

2.6. TOÁN TỬ ĐIỀU KIỆN

Có 2 loại toán tử điều kiện: AND(&&) và OR(||)

Trong đó, toán tử điều kiện tương tự như toán tử Boolean logic nhưng:

- Toán tử điều kiện AND chỉ đánh giá biểu thức thứ 2 nếu biểu thức đầu tiên trả về true vì toán tử này chỉ trả về true nếu cả 2 biểu thức đều là true
- Toán tử điều kiện OR chỉ đánh giá biểu thức thứ 2 nếu biểu thức đầu tiên trả về false vì toán tử này trả về true nếu một trong hai biểu thức đều là true



2.6. TOÁN TỬ ĐIỀU KIỆN (2)

```
int num = 0;
if (num >= 1 && num <= 10)
{
    Console.WriteLine("The number exists between 1 and 10");
}
else
{
    Console.WriteLine("The number does not exist between 1 and 10");
}

int num = -5;
if (num < 0 || num > 10)
{
    Console.WriteLine("The number does not exist between 1 and 10");
}
else
{
    Console.WriteLine("The number exists between 1 and 10");
}
```

2.7. TOÁN TỬ TĂNG, GIẢM

Toán tử ++ thêm vào biến 1 đơn vị, còn -- bớt đi một đơn vị, toán tử này có thể viết bên trái hoặc bên phải biến.

```
int a = 10;
a++;            // a là 11 (thêm 1)
++a;            // a là 12 (thêm 1)
a--;            // a là 11 (bớt 1)
--a;            // a là 10 (bớt 1)
```

Nếu biến độc lập, thì việc viết ++ hoặc -- bên trái hay bên phải không có khác biệt. Tuy nhiên trong biểu thức thì có sự khác biệt.

Nếu viết trước ++x thì toán tử ++ thi hành trước rồi mới áp dụng vào biểu thức, nếu viết sau dạng x++ thì biểu thức thi hành xong mới đến ++ (tương tự với --)

Ví dụ ++ thi hành sau do viết sau biến

```
int a = 5;
int b = 2 * a++;          // b = 10; a = 6
```

Ví dụ ++ sẽ thi hành trước

```
int a = 5;
int b = 2 * ++a;          // b = 12; a = 6
```

2.8. TOÁN TỬ GÁN

Toán tử gán, dùng để gán giá trị (biểu thức giá trị) vào biến. Các toán tử gán gồm:

Toán tử	Diễn tả	Ví dụ
=	Toán tử gán: Gán biểu thức bên phải của = vào biến bên trái	int x = 10 + 12; // x bằng 22
+=	Toán tử công thêm: Công thêm vào biến bên trái += giá trị bên phải	int x = 10; x += 2; // x bằng 12
-=	Toán tử trừ bớt: Bớt đi giá trị biến bên trái của -= một lượng bằng biểu thức bên phải	int c = 10; c -= 3; // c = 7
*=	Toán tử nhân với: a *= b tương đương a = a * b;	int x = 2; x *= 3; //x = 6
/=	Toán tử chia cho: a / b tương đương a = a / b	int a = 6; a /= 2; // a = 3
%=	Toán tử gán module: a %=b tương đương a = a % b;	int a = 10; a %= 3; // a = 1

2.8. TOÁN TỬ GÁN (2)

```
int valueOne = 5;
int valueTwo = 10;
Console.WriteLine(“Value1 =” + valueOne);
valueOne += 4;
Console.WriteLine(“Value1 += 4= “ + valueOne);
valueOne -= 8;
Console.WriteLine(“Value1 -= 8= “ + valueOne);
valueOne *= 7;
Console.WriteLine(“Value1 *= 7= “ + valueOne);
valueOne /= 2;
Console.WriteLine(“Value1 /= 2= “ + valueOne);
Console.WriteLine(“Value1 == Value2: {0}”, (valueOne == valueTwo));
```

2.9. ĐỘ ƯU TIÊN CỦA TOÁN TỬ

Độ ưu tiên của các toán tử biểu thị cho việc toán tử nào được ưu tiên thực hiện trước trong câu lệnh. Độ ưu tiên được tóm tắt ở bảng sau:

Mức	Toán tử	Thứ tự
Cao nhất	() [] .	Trái sang phải
	+ - ++ -- ! ~ new sizeof() typeof()	Phải sang trái
	* / %	Trái sang phải
	<< >>	Trái sang phải
	< <= > >=	Trái sang phải
	== !=	Trái sang phải
	& ^ |	Trái sang phải
	&& ||	Trái sang phải
	? :	
	= += -= *= /= %=	Phải sang trái
Thấp nhất	,	

2.9. ĐỘ ƯU TIÊN CỦA TOÁN TỬ (2)

```
int valueOne = 10;
Console.WriteLine((4 * 5 - 3 ) / 6 + 7 - 8 % 5);
Console.WriteLine((32 < 4) || (8 == 8));
Console.WriteLine(((valueOne *= 6) > (valueOne += 5)) &&
((valueOne /= 2) != (valueOne -= 5)));
```

2.10. TOÁN TỬ SHIFT

```
using System;
class ShiftOperator {
        static void Main(string[] args) {
                uint num = 100; // 01100100 = 100
                uint result = num << 1; // 11001000 = 200
                Console.WriteLine("Value before left shift : " + num);
                Console.WriteLine("Value after left shift " + result);
                num = 80; // 10100000 = 80
                result = num >> 1; // 01010000 = 40
                Console.WriteLine("\nValue before right shift : " + num);
                Console.WriteLine("Value after right shift : " + result);
        }
}
```

2.11. TOÁN TỬ NỐI CHUỖI

```
using System;
class Concatenation {
	static void Main(string[] args) {
		int num = 6;
		string msg = "";
		if (num < 0) {
			msg = "The number " + num + " is negative";
		}
		else if ((num % 2) == 0) {
			msg = "The number " + num + " is even";
		}
		else {
			msg = "The number " + num + " is odd";
		}
		if(msg != "")
			Console.WriteLine(msg);
	}
}
```

2.12. TOÁN TỬ BA NGÔI

Toán tử 3 ngôi (ternary operator) **? :** trong C# hoạt động trên 3 toán hạng. Nó là dạng viết tắt của câu lệnh **if-then-else.** Toán tử 3 ngôi có thể được sử dụng như dưới đây:

Variable = Condition? Expression1 : Expression2;

Toán tử 3 ngôi hoạt động như sau:

- Nếu biểu thức được đưa ra bởi điều kiện (condition) là True, kết quả biểu thức 1 (Expression1) được gán cho biến.
- Nếu là False, kết quả biểu thức 2 (Expression2) được gán cho biến.

2.12. TOÁN TỬ BA NGÔI (2)

```
using System;
class LargestNumber {
        public static void Main()
        {
                int numOne = 5;
                int numTwo = 25;
                int numThree = 15;
                int result = 0;
                if (numOne > numTwo)
                {
                        result = (numOne > numThree) ? result = numOne : result = numThree;
                }
                else
                {
                        result = (numTwo > numThree) ? result = numTwo : result = numThree;
                }
                if(result != 0)
                        Console.WriteLine("{0} is the largest number", result);
        }
}
```

3.1. CHUYỂN ĐỔI DỮ LIỆU

Chuyển đổi dữ liệu được thực hiện thông qua truyền, một cơ chế để chuyển đổi kiểu dữ liệu này sang kiểu dữ liệu khác.

Ví dụ:

- Khảo sát hệ thống trả lương của một tổ chức.
- Tổng lương của một nhân viên được tính toán và lưu trữ trong một biến thuộc kiểu float.
- Bộ phận tính lương muốn số tiền lương là một số nguyên và do đó, muốn bỏ qua bất kỳ chữ số nào sau dấu thập phân của tiền lương đã tính.
- Lập trình viên có thể đạt được điều này bằng cách sử dụng tính năng ép kiểu, cho phép bạn thay đổi kiểu dữ liệu của một biến.
- C# hỗ trợ hai kiểu ép kiểu, cụ thể là Implicit(kiểu ngầm định) và Explicit(kiểu tường minh).
- Ép kiểu chủ yếu được sử dụng để:
  - Chuyển đổi một kiểu dữ liệu sang một kiểu dữ liệu khác thuộc cùng một hệ thống phân cấp hoặc khác nhau.
  - Hiển thị đầu ra số chính xác. Ví dụ: bạn có thể hiển thị thương số chính xác trong các phép chia toán học.
  - Ngăn mất dữ liệu số nếu giá trị kết quả vượt quá phạm vi của kiểu dữ liệu của biến.

3.2. KIỂU NGẦM ĐỊNH - IMPLICIT

❖Việc chuyển đổi được thực hiện bởi trình biên dịch theo một phương thức an toàn kiểu (type-safe)

❖Từ miền giá trị nhỏ sang miền giá trị lớn

❖Từ lớp dẫn xuất sang lớp cơ sở

```
// 1. Ép kiểu từ kiểu có miền giá trị nhỏ qua kiểu có miền giá trị lớn
byte b = 7;
int i = b;          /*  Kiểu int có miền giá trị lớn hơn kiểu byte      */

float flt = 13.3f;
double dbl = flt;   /*  Kiểu double có miền giá trị lớn hơn kiểu float  */

// 2. Ép kiểu từ lớp dẫn xuất(string) qua lớp cơ sở(object)
string s = "Hello World";
object o = s;
```

3.3. KIỂU TƯỜNG MINH - EXPLICIT

❖Sử dụng phương thức Parse()

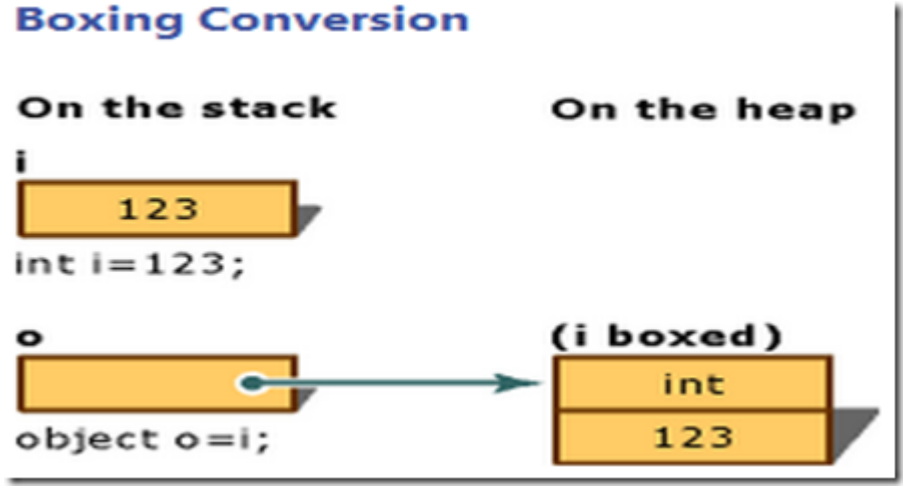
```
string stringValue = "123";
int a = int.Parse( stringValue );    // a sẽ mang giá trị 123

float b = float.Parse( "20.7" );     // b sẽ mang giá trị 20.7
bool c = bool.Parse( "true" );       // c sẽ mang giá trị True
```

❖Sử dụng lớp hỗ trợ Convert

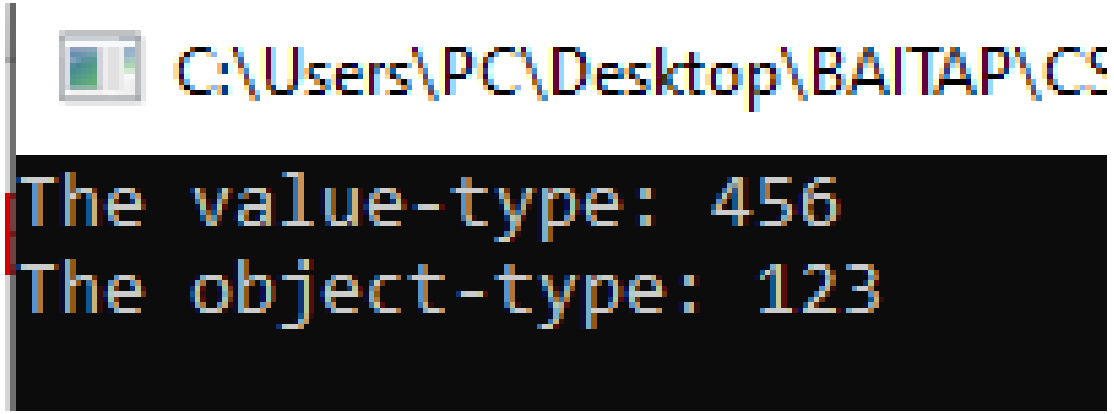
```
int a = Convert.ToInt32( "123" );
Console.WriteLine(a);                    // a = 123

double b = Convert.ToInt32( 789 );
Console.WriteLine(b);                    // b = 789

bool c = Convert.ToBoolean( 123 );
Console.WriteLine(c);                    // c = True

bool d = Convert.ToBoolean(null);        // Tham số truyền vào là null -> sẽ output giá trị
Console.WriteLine(d);                    // d = False
```

3.4. PHÂN BIỆT BOXING VÀ UNBOXING

Boxing là quá trình chuyển dữ liệu từ kiểu tham trị sang kiểu tham chiếu. Quá trình boxing một biến kiểu tham trị sẽ khởi tạo một đối tượng trong vùng nhớ Heap và copy gía trị của biến tham trị vào đối tượng mới này. Và quá trình boxing được thực hiện nhờ quá trình chuyển đổi ngầm định.




```
static void Main(string[] args)
{
    int i = 123;
    object o = i; //implicit boxing
    i = 456;
    Console.WriteLine("The value-type: {0}", i);
    Console.WriteLine("The object-type: {0}",o);
    Console.ReadLine();
}
```

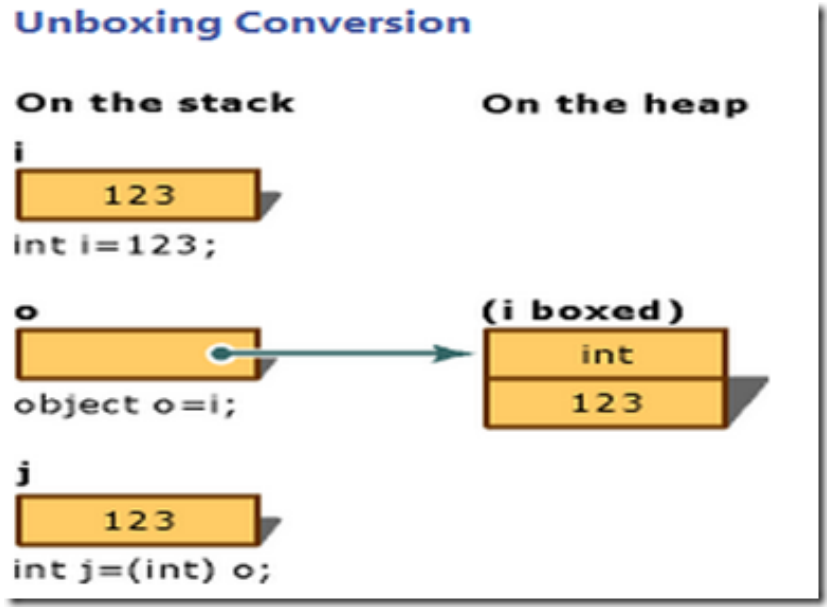


3.4. PHÂN BIỆT BOXING VÀ UNBOXING (2)

Unboxing là quá trình ngược lại với Boxing, tức là đưa từ kiểu tham chiếu ra kiểu tham trị. Quá trình này sẽ được thực hiện một cách tường minh. Gồm có 2 bước :

– Bước 1 : Kiểm tra chắc chắn rằng đối tượng đã được boxing đúng kiểu giá trị đưa ra.

– Bước 2 : Copy giá trị sang biến dữ liệu kiểu tham trị.



Khi một **kiểu giá trị được chuyển đổi thành kiểu object**, nó được gọi là boxing và ngược lại, khi một **kiểu object được chuyển đổi thành kiểu giá trị**, nó được gọi là unboxing.

```
object b = a; //boxing, b là kiểu tham chiếu chứa giá trị 100
int c = (int)b; //unboxing, c mang giá trị 100
```