**A: Tổng quan về C# và lập trình hướng đối tượng**

C# là một ngôn ngữ lập trình hướng đối tượng, trong đó chương trình được xây dựng từ các đối tượng tương tác với nhau thông qua các phương thức (action). Các đối tượng cùng loại được xem như cùng kiểu hoặc trong cùng một lớp (class). Một chương trình C# bao gồm các phần chính như sau:

* + Khai báo Namespace : Sử dụng using để nhập các namespace có sẵn (thư viện), chẳng hạn System chứa các lớp cơ bản của C# như Console, String, v.v.
  + Định nghĩa namespace: Đặt tên namespace giúp tổ chức mã nguồn, tránh xung đột tên giữa các lớp.
  + Định nghĩa Class : Lớp chính chứa phương thức Main bắt đầu quá trình thực thi chương trình. Lớp có thể chứa các biến và phương thức khác phục vụ cho chương trình.
  + Phương thức của Class : Phương thức bổ sung có thể được gọi từ Main hoặc từ các phương thức khác trong lớp.
  + Thuộc tính của Class
  + Phương thức Main (Điểm bắt đầu thực thi chương trình): Điểm bắt đầu thực thi chương trình, nơi các lệnh đầu tiên của chương trình được thực hiện.
  + Lệnh và biểu thức: Phương thức bổ sung có thể được gọi từ Main hoặc từ các phương thức khác trong lớp.
  + Comment

Lưu ý: C# phân biệt chữ hoa và chữ thường (case-sensitive), và các lệnh kết thúc bằng dấu chấm phẩy (;)

**B: Quy tắc đặt tên trong C#**

1. **Quy tắc Camel Case**: Chữ cái đầu tiên viết thường, các từ tiếp theo viết hoa chữ cái đầu.
   * Dùng cho biến có phạm vi truy cập là private hoặc protected và các tham số hàm.
   * Ví dụ: educationFree, chiPi, ...
2. **Quy tắc Pascal Case**: Viết hoa chữ cái đầu của mỗi từ.
   * Dùng cho các thành phần khác như tên hàm, interface, enum, sự kiện, ...
   * Ví dụ: CalculateInterest, EmployeeList.

**Lưu ý khi đặt tên biến:**

* Đặt tên ngắn gọn, dễ hiểu, thể hiện rõ mục đích của biến (ví dụ: Name, Tuoi, GioiTinh).
* Tránh sử dụng tên biến là một ký tự đơn lẻ như i, k, m ngoại trừ trong vòng lặp.
* C# phân biệt chữ hoa chữ thường (ví dụ: a khác A và Console.WriteLine() khác Console.WRITELINE()).

**C: Các kiểu dữ liệu cơ bản trong C#**

* **int**: số nguyên có dấu, dùng 32 bit, giá trị từ -2,147,483,648 đến 2,147,483,647.
* **sbyte, byte, short, ushort, long, ulong**: các kiểu số nguyên có hoặc không dấu với kích thước và phạm vi khác nhau.
* **float, double, decimal**: kiểu số thực dấu chấm động với kích thước khác nhau.
* **char**: ký tự Unicode 16 bit.
* **bool**: kiểu logic chỉ có giá trị true hoặc false.
* **string**: chuỗi ký tự.
* **object**: kiểu cơ sở, mọi đối tượng trong C# đều kế thừa từ kiểu này.

**D: Nhập, xuất và chuyển đổi kiểu dữ liệu trong C#**

* **Xuất:**

+ Console.WriteLine(); 🡺 In ra màn hình và xuống dòng

+ Console.Write(); 🡺 In ra màn hình nhưng không xuống dòng

* **Nhập:**

+ Console.ReadLine(); 🡺 Cho phép nhập dữ liệu và trả về chuỗi

+ Console.ReadKey(); 🡺 trả về thông tin phím bấm, lấy ký tự bằng Console.ReadKey().KeyChar

* **Ép kiểu dữ liệu:**

Trong C#, ép kiểu có 4 loại:

- Chuyển đổi kiểu ngầm định (implicit): Là việc chuyển đổi được thực hiện bởi trình biên dịch và chúng ta không cần tác động gì.

Ví dụ:

int intValue = 10;

long longValue = intValue; /\* Chuyển kiểu ngầm định vì kiểu long có miền giá trị lớn hơn kiểu int nên có thể chuyển từ int sang long.\*/

float floatValue = 10.9f;

double doubleValue = floatValue; /\* Tương tự vì kiểu double có miền giá trị lớn hơn kiểu float nên có thể chuyển từ float sang double.\*/

- Chuyển đổi kiểu tường minh (explicit): Là việc chuyển kiểu một cách rõ ràng và dùng từ khóa chỉ định chứ không dùng phương thức.

- Cú pháp:

(<kiểu dữ liệu>) <biến cần ép kiểu>

+ Trong đó :

- <kiểu dữ liệu> là kiểu dữ liệu mà mình muốn chuyển sang.

- <biến cần ép kiểu> là biến chưa dữ liệu cần ép kiểu.

\*Lưu ý : Phải có cặp dấu ngoặc tròn ().

+ Ý nghĩa:

- Ép kiểu của <biến cần ép kiểu> về <kiểu dữ liệu> nếu thành công sẽ trả ra giá trị kết quả ngược lại sẽ báo lỗi chương trình. Đặc biệt đối với số:

- Ta có thực hiện ép kiểu dữ liệu lớn hơn về kiểu dữ liệu nhỏ hơn mà không báo lỗi.

- Nếu dữ liệu cần ép kiểu vượt quá miền giá trị của kiểu dữ liệu muốn ép kiểu về thì chương trình sẽ tự cắt bit cho phù hợp với khả năng chứa kiểu dữ liệu đó (cắt từ bên trái qua).

- Sử dụng phương thức, lớp hỗ trợ sẵn:

+ Dùng phương thức Parse(), TryParse().

+ Dùng lớp hỗ trợ sẵn (Convert).

**E. Toán Tử TrongC#**

**Toán tử toán học**

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

+ Toán tử toán học bao gồm :

- Toán tử '+' : Thực hiện cộng 2 toán hạng, VD : a + b = 19

- Toán tử '-' : Thực hiện trừ 2 toán hạng, VD : a - b = 1

- Toán tử '\*' : Thực hiện nhân 2 toán hạng, VD : a \* b = 90

- Toán tử '/' : Thực hiện chia lấy phần nguyên 2 toán hạng nếu chúng là số nguyên, ngược lại thì chia như bình thường, VD : a / b = 1

- Toán tử '%' : Thực hiện chia lấy dư 2 toán hạng, VD : a % b = 1

- Toán tử "++" : Tăng giá trị của toán hạng lên 1 đơn vị, VD : a++ = 11

- Toán tử "--" : Giảm giá trị của toán hạng đi 1, VD : a-- = 9

+ Lưu ý: đối với toán tử ++ và -- cần phần biệt a++ và ++a (hoặc a-- và --a):

- a++: là sử dụng giá trị của biến a để thực hiện biểu thức trước rồi mới thực hiện tăng lên 1 đơn vị. Tương tự cho a--.

- ++a: là tăng giá trị biến a lên 1 đơn vị rồi mới sử dụng biến a để thực hiện biểu thức. Tương tự cho --a.

**Toán tử khởi tạo và gán**

+ Toán tử khởi tạo và gán thường được sử dụng nhằm mục đích lưu lại giá trị cho một biến nào đó. Một số toán tử khởi tạo và gán hay được sử dụng :

- Toán tử '=' : Gán giá trị của toán hạng bên phải cho toán hạng bên trái, VD : K = 10 sẽ gán 10 cho biến K

- Toán tử "+=" : Lấy toán hạng bên trái cộng toán hạng bên phải sau đó gán kết quả lại cho toán hạng bên trái, VD : K += 1 tương đương với K = K + 1

- Toán tử "-=" : Lấy toán hạng bên trái trừ toán hạng bên phải sau đó gán kết quả lại cho toán hạng bên trái, VD : K -= 1 tương đương với K = K – 1

- Toán tử "\*=" : Lấy toán hạng bên trái nhân toán hạng bên phải sau đó gán kết quả lại cho toán hạng bên trái, VD : K \*= 1 tương đương với K = K \* 1

- Toán tử "/=" : Lấy toán hạng bên trái chia lấy phần nguyên với toán hạng bên phải sau đó gán kết quả lại cho toán hạng bên trái, VD : K /= 1 tương đương với K = K / 1

- Toán tử "%=" : Lấy toán hạng bên trái chia lấy dư với toán hạng bên phải sau đó gán kết quả lại cho toán hạng bên trái, VD : K %= 1 tương đương với K = K % 1

+ Một số lưu ý khi sử dụng các toán tử trên:

- Toán tử bên trái thường là một biến, còn toán tử bên phải có thể là biến có thể là biểu thức đều được.

- Một phép toán gán hoặc khởi tạo có thể được sử dụng như là toán hạng bên phải cho một phép gán hoặc khởi tạo khác.

**Toán tử quan hệ**

- Toán tử "==" : So sánh 2 toán hạng có bằng nhau hay không. Nếu bằng thì trả về true nếu không bằng thì trả về false, VD : a == b sẽ trả về false

- Toán tử "!=" : So sánh 2 toán hạng có bằng nhau hay không. Nếu không bằng thì trả về true nếu bằng thì trả về false, VD : a != b sẽ trả về true

- Toán tử '>' : 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, VD : a > b sẽ trả về true

- Toán tử '<' : 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, VD : a < b sẽ trả về false

- Toán tử ">=" : 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, VD : a >= b sẽ trả về true

- Toán tử "<=" : 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, VD : a <= b sẽ trả về false

+ Lưu ý:

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

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

**Toán tử khác**

+ Ngoài những toán tử đã giới thiệu ở trên chúng ta vẫn còn nhiều toán tử khác cũng hay sử dụng :

- Toán tử "sizeof()" : Trả về kích cỡ của một kiểu dữ liệu, VD : sizeof(int) sẽ trả về 4

- Toán tử "typeof()" : Trả về kiểu dữ liệu của một lớp, VD : typeof(string) sẽ trả về kiểu dữ liệu System.String

- Toán tử "new" : Cấp phát vùng nhớ mới, áp dụng cho kiểu dữ liệu tham chiếu, VD : DateTime dt = new DateTime()

- Toán tử "is" : Xác định đối tượng có phải là một kiểu cụ thể nào đó hay không. Nếu đúng sẽ trả về true ngược lại trả về false

- Toán tử "as" : Ép kiểu mà không gây ra lỗi. Nếu ép kiểu không thành công sẽ trả về null

- Toán tử "?:" : Được gọi là toán tử 3 ngôi. Tương đương với cấu trúc điều kiện

**F. Cấu Trúc Lệnh Rẽ Nhánh Trong C#**

**1. If-else**

- Dạng thiếu:

if ([Biểu thức điều kiện]) <Câu lệnh thực hiện>

+ Trong đó :

- if là từ khóa bắt buộc.

- <Biểu thức điều kiện> là biểu trức dạng boolean (trả về true hoặc false).

- <Câu lệnh thực hiện> là câu lệnh muốn thực hiện nếu <Biểu thức điều kiện> là đúng.

- Dạng đủ:

if <Biểu thức điều kiện>

<Câu lệnh thực hiện 1>

Else

<Câu lệnh thực hiện 2>

+ Trong đó :

- if, else là từ khóa bắt buộc.

- <Biểu thức điều kiện> là biểu trức dạng boolean (trả về true hoặc false).

- <Câu lệnh thực hiện 1> là câu lệnh muốn thực hiện nếu <Biểu thức điều kiện> là đúng.

- <Câu lệnh thực hiện 2> là câu lệnh muốn thực hiện nếu <Biểu thức điều kiện> là sai.

**2. Switch – case:**

- Dạng thiếu :

switch (<biểu thức>)

{

case <giá trị thứ 1>:

<câu lệnh thứ 1>;

break;

case <giá trị thứ 2>:

<câu lệnh thứ 2>;

break;

. . .

case <giá trị thứ n>:

<câu lệnh thứ n>;

break;

}

Dạng đủ :

switch (<biểu thức>)

{

case <giá trị thứ 1>:

<câu lệnh thứ 1>;

break;

case <giá trị thứ 2>:

<câu lệnh thứ 2>;

break;

. . .

case <giá trị thứ n>:

<câu lệnh thứ n>;

break;

default:

<câu lệnh mặc định>;

break;

}