LESSON 03

I. CẤU TRÚC RẼ NHÁNH IF ELSE TRONG TRONG C#

❖ Định nghĩa

➢ Xét 2 mệnh đề sau:

- Mệnh đề 1: "**Nếu** trời mưa **thì** đường trơn".
- Mệnh đề 2: "**Nếu** bạn rảnh **thì** đi chơi **ngược lại** thì thôi"

➢ Các bạn để ý những chữ in đậm trong 2 mệnh đề trên. Cấu trúc **Nếu. . . thì. . .** hoặc Nếu. . . thì. . . ngược lại thì. . . được gọi là cấu trúc rẽ nhánh.

➢ Trong hầu hết các ngôn ngữ lập trình đều có loại cấu trúc rẽ nhánh:

- . . else. . .
- . . case. . . (sẽ nghiên cứu ở bài sau)

➢ Cấu trúc rẽ nhánh **If. . . else. .** . còn có tên gọi khác là "**câu lệnh điều kiện**".

❖ Phân loại câu lệnh điều kiện If else

Trong C#, câu lệnh này được chia thành 2 loại chính:

- Dạng thiếu.
- Dạng đủ.

Câu điều kiện dạng thiếu:

➢ **Cú pháp:**

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

- 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

➢ **Ý nghĩa:**

Nếu <Biểu thức điều kiện> trả về true thì thực hiện <Câu lệnh thực hiện> ngược lại thì không làm gì cả.

➢ **Ví dụ:**

```
0 references
static void Main(string[] args)
{
    string T = "TT Design";
    if (T == "TT Design") Console.WriteLine("This is our company!");

    /*
     Biểu thức điều kiện sử dụng toán tử == để so sánh xem giá trị biến T có bằng “TT Design” hay không.
     Nếu bằng trả thì trả về true ngược lại thì trả về false.

     In ra màn hình chữ “This is our company!” nếu biểu thức trên đúng.
    */
}
```

Câu điều kiện dạng đủ:

➢ **Cú pháp:**

If **<Biểu thức điều kiện>**

<Câu lệnh thực hiện 1>

else

<Câu lệnh thực hiện 2

- **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

➢ **Ý nghĩa:**

Nếu <Biểu thức điều kiện> trả về true thì thực hiện <Câu lệnh thực hiện 1> ngược lại thì thực hiện <Câu lệnh thực hiện 2>.

➢ **Ví dụ:**

```
0 references
static void Main2(string[] args)
{
    string T = "TT Design";

    if (T == "TT Design") // Nếu giá trị K bằng “Kteam” thì
        Console.WriteLine("This is our company!"); // In ra màn hình “This is our company!”

    else // Ngược lại thì
        Console.WriteLine("This is not our company!"); // In ra màn hình “This is not our company!”
}
```

❖ Một số lưu ý khi sử dụng

- <Biểu thức điều kiện> có thể chứa nhiều biểu thức con bên trong và các biểu thức con liên kết với nhau bằng các **toán tử quan hệ** nhưng tất cả phải trả về kiểu boolean (true hoặc false).

```
0 references
static void Main3(string[] args)
{
    int T = 5;

    if (T>0 && T<10)
        Console.WriteLine("T is in range from 0 to 10");
    else
        Console.WriteLine("T is out of range from 0 to 10");

    Console.ReadKey();
}
```

- Trong câu lệnh có thể chứa một câu lệnh điều kiện con nữa. Bạn có thể vận dụng điều này để làm giải quyết những vấn đề phức tạp.

```
/*
If <Biểu thức điều kiện 1>
{
    If <Biểu thức điều kiện 2>
    {
        <Câu lệnh thực hiện 1>
    }
    else
    {
        <Câu lệnh thực hiện 2>
    }

    <Câu lệnh thực hiện 3>
}
Else
{
    If <Biểu thức điều kiện 3>
    {
        <Câu lệnh thực hiện 4>
    }

    If <Biểu thức điều kiện 4>
    {
        <Câu lệnh thực hiện 5>
    }
     <Câu lệnh thực hiện 6>
 }
```

- Ngoài cấu trúc **If. . . else. . .** cơ bản trên còn có cấu trúc nhỏ khác là **If. . . else if.. . else**

```
If <Biểu thức điều kiện 1>
{
        <Câu lệnh thực hiện 1> //Thực hiện khi <biểu thức điều kiện 1> đúng
}

ezlse if <Biểu thức điều kiện 2>
{
        <Câu lệnh thực hiện 2> // Thực hiện khi <biểu thức điều kiện 1> sai và
<biểu thức điều kiện 2> đúng
}

else
{
        <Câu lệnh thực hiện 3> //Thực hiện khi cả 2 biểu thức điều kiện trên đều
sai (các trường hợp còn lại)
}
```

- **Ví dụ**

Viết chương trình giải phương trình bậc 1: Ax + B = 0

❖ BÀI TẬP

1. Viết chương trình nhập vào một số tương ứng là năm. Xuất ra màn hình năm vừa nhập và tuổi của một người tương ứng với năm vừa nhập đó.
2. Từ câu 1. Thêm chức năng:
   - Nếu tuổi người đó < 16 thì hiện thông báo theo format:

     Bạn<Tuổi>, tuổi vị thành niên.
   - Nếu tuổi người đó >= 16 và <18 thì hiện thông báo theo format:

     Bạn <Tuổi>, tuổi trưởng thành.
   - Nếu tuổi người đó >= 18 thì hiện thông báo theo format:

     Bạn<Tuổi>, già rồi.
3. Viết trò chơi Kéo Búa Bao với cách chơi: Người dùng sẽ nhập vào các số 1 hoặc 2 hoặc 3 tương ứng với kéo hoặc búa hoặc bao. Máy sẽ ngẫu nhiên sinh ra một số trong 3 số và tính toán máy hoặc người chiến thắng. Nhớ phải in kết quả ra màn hình.

II. CẤU TRÚC RẼ NHÁNH SWITCH CASE TRONG C#

❖ Phân loại câu lệnh điều kiện Switch case

Trong C#, câu lệnh này được chia thành 2 loại chính:

- Dạng thiếu.
- Dạng đủ.

Câu lệnh dạng thiếu:

➢ **Cú pháp:**

```
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;
}
```

- **switch, case** là từ khóa bắt buộc
- break là một lệnh nhảy
- <biểu thức> phải là biểu thức trả về kết quả kiểu:

  o Số nguyên (int, long, byte, . . .)

  o Ký tự hoặc chuỗi (char, string)
- <giá trị thứ i> với i = 1..n là giá trị muốn so sánh với giá trị của <biểu thức>.
- <câu lệnh thứ i> với i = 1..n là câu lệnh muốn thực hiện khi <giá trị thứ i> tương ứng bằng với giá trị của <biểu thức>.

➢ **Ý nghĩa:**

Duyệt lần lượt từ trên xuống dưới va kiểm tra xem giá trị của <biểu thức> có bằng với <giá trị thứ i> đang xét hay không. Nếu bằng thì thực hiện<câu lệnh thứ i> tương ứng.

➢ **Ví dụ:**

```
0 references
static void Main7(string[] args)
{
    int T = 10;

    switch (T) // giá trị biểu thức la giá trị của biến T (kiểu số nguyên)
    {
        case 3: // các giá trị so sánh cũng la kiểu số nguyên
            Console.WriteLine("TT Design"); // lệnh thực hiện nếu T = 3
            break; // lệnh thoát ra khỏi cấu trúc
        case 9:
            Console.WriteLine("Design"); // tương tự
            break;
        case 10:
            Console.WriteLine("This is our company!"); // tương tự
            break;
    }
}
```

Câu lệnh dạng đủ:

Cú pháp:

```
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;
}
```

- Switch,case là từ khóa bắt buộc.
- break là một lệnh nhảy
- <biểu thức> phải là biểu thức trả về kết quả kiểu:

  o Số nguyên (int, long, byte, . . .)

  o Ký tự hoặc chuỗi (char, string)
- <giá trị thứ i> với i = 1..n là giá trị muốn so sánh với giá trị của <biểu thức>.
- <câu lệnh thứ i> với i = 1..n là câu lệnh muốn thực hiện khi <giá trị thứ i> tương ứng bằng với giá trị của <biểu thức>.
- <câu lệnh mặc định> là câu lệnh sẽ được thực hiện nếu giá trị <biểu thức> không bằng với <giá trị thứ i> nào

Ý nghĩa:

Duyệt lần lượt từ trên xuống dưới va kiểm tra xem giá trị của <biểu thức> có bằng với <giá trị thứ i> đang xét hay không. Nếu bằng thì thực hiện<câu lệnh thứ i> tương ứng. Nếu không bằng tất cả các <giá trị thứ i> thì sẽ thực hiện <câu lệnh mặc định>.

➢ **Ví dụ:**

```
0 references
static void Main8(string[] args)
{
    int T = 10;

    switch (T) // giá trị biểu thức la giá trị của biến T (kiểu số nguyên)
    {
        case 3: // các giá trị so sánh cũng la kiểu số nguyên
            Console.WriteLine("TT Design"); // lệnh thực hiện nếu T = 3
            break; // lệnh thoát ra khỏi cấu trúc
        case 9:
            Console.WriteLine("Design"); // tương tự
            break;
        case 10:
            Console.WriteLine("This is our company!"); // tương tự
            break;
        default:
            Console.WriteLine("Try to connect to TT Design"); // tương tự
            break;
    }
}
```

- **Ví dụ.**

Tim 1 số vi dụ về switch case cho mn

III. KIỂU DỮ LIỆU OBJECT TRONG C#

Ở bài KIỂU DỮ LIỆU TRONG C# chúng ta có đề cập đến một kiểu dữ liệu tham chiếu đó là **kiểu dữ liệu object**. Trong bài học này chúng ta sẽ cùng tìm hiểu chi tiết về kiểu dữ liệu này

❖ Khái niệm

- Là một kiểu dữ liệu cơ bản của tất cả các kiểu dữ liệu trong .NET.
- Mọi kiểu dữ liệu đều được kế thừa từ System.**Object** (sẽ được trình bày trong bài tính kế thừa).
- Thuộc kiểu dữ liệu tham chiếu (đã trình bày rồi).

Kiểu dữ liệu object cung cấp một số phương thức ảo cho phép mình overload để sử dụng: (tính kế thừa)

Một số phương thức tiêu biểu nằm trong **object:**

Phương thức	Ý nghĩa
ToString()	Trả về kiểu chuỗi của đối tượng (chuyển từ kiểu dữ liệu nào đó về kiểu chuỗi)
GetHashCode()	Trả về mã băm của đối tượng.
Equals()	So sánh 2 đối tượng và trả về true khi 2 đối tượng có giá trị bằng nhau, ngược lại trả về false.
GetType()	Trả về kiểu dữ liệu của đối tượng.

❖ Boxing vs Unboxing

Boxing là quá trình chuyển dữ liệu từ kiểu dữ liệu giá trị sang kiểu dữ liệu tham chiếu.

- Khởi tạo một đối tượng trong vùng nhớ Heap (đã được trình bày trong bài KIỂU DỮ LIỆU TRONG C#).
- Copy giá trị của biến có kiểu dữ liệu giá trị vào đối tượng này.
- Quá trình boxing được thực hiện ngầm định

```
// Khởi tạo biến Value kiểu int (kiểu dữ liệu giá trị)
        int Value = 109;

        /* thực hiện boxing bằng cách:
         * Khởi tạo đối tượng ObjectValue kiểu object
         * Gán giá trị của biến Value vào ObjectValue */
        object ObjectValue = Value;
```

Unboxing là quá trình ngược lại với boxing, tức là đưa dữ liệu từ kiểu dữ liệu tham chiếu về kiểu dữ liệu giá trị.

- Kiểm tra xem đối tượng cần un-boxing có thuộc đúng kiểu dữ liệu đưa ra hay không.
- Nếu đúng thì thực hiện copy giá trị của đối tượng sang biến dữ liệu kiểu giá trị. Ngược lại thì thông báo lỗi.
- Unboxing được thực hiện tường minh và thông qua cách ép kiểu tường minh

```
int Value = 109;

        // Boxing
        object ObjectValue = Value;

        /* thực hiện unboxing bằng cách:
         * Kiểm tra dữ liệu biến ObjectValue thấy thuộc đúng kiểu int.
         * Gán giá trị của biến ObjectValue vào biến NewValue bằng cách ép kiểu
tường minh.
         * Biến NewValue sẽ mang giá trị là 109*/
        int NewValue = (int)ObjectValue;
```

❖ Từ khóa var

var là từ khóa hỗ trợ khai báo biến mà không cần kiểu dữ liệu, kiểu dữ liệu sẽ được xác định khi gán giá trị cho biến, lúc đó chương trình sẽ tự ép kiểu cho biến.

Những lưu ý khi sử dụng từ khóa var:

Bắt buộc phải gán giá trị ngay khi khởi tạo biến và không thể khởi tạo giá trị null cho biến var.

```
// Lỗi vì chưa khởi tạo giá trị cho biến varInt
var varInt;

// Lỗi vì không được khởi tạo giá trị null cho biến varString.
var varString = null;

// Lỗi vì phải khởi tạo giá trị ngay khi khai báo
var varLong;
varLong = 109;

// Khai báo đúng!
var varBool = true;
```

var chỉ là từ khóa dùng để khai báo biến không phải là một kiểu dữ liệu.
Khai báo biến bằng từ khóa var thường được ứng dụng trong: Vòng lặp foreach hoặc truy vấn Linq.

```
static void Main9(string[] args)
{
    /* Vì biến StringVariable được khởi tạo giá trị kiểu chuỗi
     * nên trình biên dịch sẽ hiểu biến này như là biến kiểu string.
     */

    var varString = "TTDesign";
    // Khai báo tường minh biến kiểu string
    string Content = "TTDesign";

    // In giá trị của biến StringVariable và biến Content
    Console.WriteLine(varString);
    Console.WriteLine(Content);
}
```