Lesson 8 - Conditionals I

Recall from Lecture 5,

Relational Operators

Relational Operators are used to evaluate conditions. There are 6 main relational operators. These will return a `bool` type value.

Operator	Description	Example	Result
>=	Greater than or equal to	7 >= 4	True
<=	Less than or equal to	7 <= 4	False
==	Equal to	7 == 4	False
!=	Not equal to	7 != 4	True

Example

```
int age = 25;
bool isAdult = (age >= 18);

// You can also omit brackets and write this as below
// bool isAdult = age >= 18;
// however it looks confusing without the brackets

Console.WriteLine(isAdult);
```

Output

```
True
```

Conditionals

Conditionals are decision statements, on which a certain condition is checked and then based on the outcome, a decision is taken.

if

General form is as below:

```
if (condition)
{
        // code that will execute if the condition is true
}
```

When condition is true,

```
int x = 10;

if (x > 5)      // This returns true
{
        Console.WriteLine("x is greater than 5");
}

Console.WriteLine("Outside condition block.");
```

Output

```
x is greater than 5
Outside condition block.
```

When condition is `false`,

```
int x = 4;

if (x > 5)      // This returns false
{
        Console.WriteLine("x is greater than 5");
}
Console.WriteLine("Outside condition block.");
```

Output

```
Outside condition block.
```

You can skip the curly braces if there is only 1 statement in the `if` block.

```
int x = 4;

if (x > 5)     // This returns false
        Console.WriteLine("x is greater than 5");

Console.WriteLine("Outside condition block.");
```

Output

```
Outside condition block.
```

if - else

The `else` block extends functionality of `if`, and performs some functionality when the condition check in the `if` block fails (i.e., when it returns `false`). Note that else can only be used after an `if` block, it has no meaning on its own.

```
int x = 4;

if (x > 5)     // This returns false
        Console.WriteLine("x is greater than 5");
else
{
        Console.WriteLine("x is less than or equal to 5");
}

Console.WriteLine("Outside condition block.");
```

Output

```
x is less than or equal to 5
Outside condition block.
```

You can skip the curly braces in the `else` block as well if there is only one statement in it.

```
int x = 4;

if (x > 5)     // This returns false
        Console.WriteLine("x is greater than 5");
```

```
else
        Console.WriteLine("x is less than or equal to 5");

Console.WriteLine("Outside condition block.");
```

Output

```
x is less than or equal to 5
Outside condition block.
```

When statement is `true`,

```
int x = 10;

if (x > 5)      // This returns true
        Console.WriteLine("x is greater than 5");
else
        Console.WriteLine("x is less than or equal to 5");

Console.WriteLine("Outside condition block.");
```

Output

```
x is greater than 5
Outside condition block.
```

if - else if ...

What if we want to check multiple conditions? Then we can use `if` - `else if` in a chain combination. This is not the optimal and most cleanest way of doing things, however its very useful in some cases.

```
int x = 5;

if (x == 5) // This returns false
    Console.WriteLine("x is greater than 5");
else if (x == 5) // This returns true
    Console.WriteLine("x is equal to 5");
else
        Console.WriteLine("x is less than 5");

Console.WriteLine("Outside condition block.");
```

Output

```
x is equal to 5
Outside condition block.
```