Lesson 11 - Conditionals II

Recall from Lecture 8,

> ### 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.
> ```

There is another way to write a conditional statement with multiple outcomes, called the `switch-case` statement.

Switch - case

`switch-case` provides a more elegant way to test a variable for equality against a list of values. Each value is called a `case` and the variable being switched on i checked on each case.

```
int num = 3;
switch (num)
```

```
{
        case 1:
                Console.WriteLine("one");
                break;
        case 2:
                Console.WriteLine("two");
                break;
        case 3:
                Console.WriteLine("three");
                break;
}
```

Output

```
three
```

> Note: a `switch` statement can have any number of `case` statements, but case lables (the values to be checked) must not be identical. The values must be constants. And in most cases you will have a `break;` statement to terminate it.

The default case

The `default` case is similar to `else`, it is triggered when no other case is executed.

```
int age = 75;
switch (age)
{
        case 16:
                Console.WriteLine("Too young");
                break;
        case 42:
                Console.WriteLine("Adult");
                break;
        case 70:
                Console.WriteLine("Senior");
                break;
        default:
                Console.WriteLine("The default case");
                break;
}
```

```
The default case
```

The break statement

This statement terminates the `switch` statement. Without it, the `switch` statement would continue to execute, and would fall through to the next `case` statement, even when the labels don't match. This is called **fallthrough** and this is not recommended, you should end all `case` statements with a `break` statement.

Shorthand switch

You can use the shorthand `switch-case` syntax when you want to assign a value to a variable based on switching another variable.

```
int age = 75;

string ageInWords = age switch
{
    18 => "Eighteen",
    40 => "Forty",
    75 => "Seventy Five",
    _ => "Unlisted"
};

Console.WriteLine(ageInWords);
```

Output

```
Seventy Five
```

The Conditional Operator

Regular conditional using if-else for checking if a person is an adult or not,

```
int age = 20;
bool isAdult;

if (age >= 18)
        isAdult = true;
else
        isAdult = false;

Console.WriteLine($"Person is an adult: {isAdult}");
```

This checks the age variable and then assigns a value to another variable based on the result. We can do this in shorthand using the conditional operator.

```
int age = 20;
bool isAdult = (age >= 18) ? true : false;
Console.WriteLine($"Person is an adult: {isAdult}");
```

```
Person is an adult: True
```

A crazy way to use the conditional operator and check multiple conditions

```
int age = 20;

string ageString = (age >= 18)
? (age >= 40)
        ? (age >= 70)
                ? "Senior"
                : "Middle Aged"
        : "Adult"
: "Child";

Console.WriteLine(ageString);
```

Output

```
Adult
```