Visual C# selection with if

Selection is the ability to look at a state in code and make a choice. One value will cause a section of code to run and another fire a different section of code.

This is the basic `if` code

```
int num = 12;
if(num < 12)   // look no ‘;’
       Console.WriteLine("12 is less than 12");
```

The final line of code is not going to run because 12 is never going to be less than 12. The following comparison operators are also available.

- <=
- >=
- >
- ==
- !=

The code above is the same as the following code

```
 if (num < 12)
 {
      Console.WriteLine("12 is less than 12");
 }
```

The brace ({ }) is used to divide up a set of code all linked to the same section (num < 12). If there is only 1 line running as a consequence of the if the { } is not required.

These 2 code sections do not have the same output

```
int num = 12;
if (num < 12)
{
     Console.WriteLine("12 is less than 12");
     num = 1;
}
Console.WriteLine(num);
```

Is not the same as:

```
int num = 12;
if (num < 12)
     Console.WriteLine("12 is less than 12");
num = 1;
Console.WriteLine(num);
```

The indentation is not required but helps show what is going on. The following is a valid but useless if clause. C# throws a warning here.

```
if (num <= 12) ;
```

The `if` can be expanded by use of else. The `else` clause fires if the `if` is not true.

```
int num = 12;
if (num < 12)
{
       Console.WriteLine("12 is less than 12");
}
else
       Console.WriteLine("Utter rot this line will fire");
Console.WriteLine(num);
```

The above code shows the option of using the { } to break up the code.

There is another option `else if` (2 words) that allows a range of choices to be made.

```
int num = 12;
if (num <= 12)
      Console.WriteLine("12 is less than or equal to 12");
else if (num == 12)
      Console.WriteLine("12 is 12");
else
      Console.WriteLine("Utter rot this line will fire");
Console.WriteLine(num);
```

The first 2 cases here are true but only the 1st in the set will fire. The `else if` will not fire. The following is a more useful example.

```
int num = 12;
if (num < 12)
      Console.WriteLine("12 is less than 12");
else if (num == 12)
      Console.WriteLine("12 is 12");
else
      Console.WriteLine("Utter rot this line will fire");
Console.WriteLine(num);
```

By using `if` we can accept some input make a decision and choose the appropriate output.

1. Write a program to accept a name and an valid age and if the person is 21 or over print "NAME is AGE years old and too old for this program!!!".

2. Ask for 2 numbers and divide one into the other only if the second number is not 0.

3. Ask for a number and output the square root of that number if it is not a negative number.

4. Ask for 2 numbers and output if the second number is a factor of the first (that is the first is a multiple of the second). The % operator will be needed here.

5. Write a program to accept a name and hours worked from the keyboard. The rate of pay for 40 hours and below is £9.50. Above 40 hours the rate is £12.50. Calculate the total pay and print out the name and total pay.

6. Modify program 4 to accept a tax code. If the tax code is 1 deduct tax at 20%, if the tax code is 2 deduct tax at 40%. Print out the name, gross pay, tax and net pay.

7. Write a program that asks for a person's age. If the age is less than 12 then print a message "Too young to watch this film." If the person is over 15 print a message "Too old to watch this film." If the age is 12 to 15 then print a message "You can watch the film."

8. Write a program to accept a password from the keyboard. If the password is not "C#" then display a message that tells the user s/he cannot use the program. If the password is correct display a message that access is granted.

Previously there was a problem with a user entering invalid data from a console such as entering a word when a number was expected. If the word is then passed to some mathematical operations we are asking for trouble. The `if` statement can be used to check on data entered and make appropriate choices.

```
int result;
string word;
word = Console.ReadLine();
if(Int32.TryParse(word, out result))
     Console.WriteLine(result);  //put the value of word in result
else
     Console.WriteLine("not good"); //handle the mess up
Console.ReadLine();
```

The crucial `if` line can also be written as

```
if(Int32.TryParse(word, out result)== true)
```

There are similar methods for other variables such as `float.TryParse()`, `bool.TryParse()` and `double.TryParse()`. There is not an equivalent for strings “12” and “fred” are both valid strings; neither can be passed to mathematical operators.

To find if part of a word is a letter or number `char.IsLetter()`,`char.IsDigit`, `char.IsLetterOrDigit()` are all useful. Obviously they only work on a char not a number or a string. There are ways of finding letters within words and of running through all the letters in turn within the word (but this latter requires a knowledge of loops).

```
string word = "test1";
//look at the 4th letter counting from 0
bool isIt = char.IsDigit(word, 4);
if (isIt)
     Console.WriteLine("it is a number");
else
     Console.WriteLine("no letter there");
Console.ReadLine();
```

9. Ask for a character input and output whether the character is a number, a letter or some keyboard character that is neither a number nor a letter such as # or /.

10. Create a program to accept 2 numbers from the console, check that the input is valid and then divide the first number by the second number. If both are valid output the numbers entered, the result and remainder of the division. If one or both of the values entered are not valid numbers output a warning message but do not run the calculations.