**Conditionals**

Code runs in sequence meaning that it runs from top to bottom executing each line of code as it goes. This does not work as our programs get more and more complex as we go. We may want the program to print a statement if something is true or if it is false. We may just need to determine if a variable meets some criteria so the program can re-act. Conditionals will serve our purpose for this. This will be if, if/else, if/else if, and a Switch statement.

Let us start with something simple.

int myNumber = 5;

Console.WriteLine("Number guessing Game");

Console.Write("Guess a whole number from 1 to 10 > ");

int numberGuess = int.Parse(Console.ReadLine());

if (numberGuess != myNumber)

{

Console.WriteLine("Nope not the number!");

Console.ReadLine();

}

This works, not well but works. Let’s talk about this for a moment. The syntax is simple:

If (Expression)

{

Code

}

In the expression you will need to use Relational operators: page 202

‘>’ – Greater than

‘<’ – Lesser than

‘>=’ – Greater than or equal to

‘<=’ – Less than or equal to

‘==’ – Equal to

‘!=’ Not equal to

In the expression you see that I used the not equal to make suer the user’s input is not equal to the hard coded number in the myNumber. If this expression is true then it will print the string but this does not help us much does it. Let us use the if/else statement to tell the user if they are too high or too low.

if (numberGuess > myNumber)

{

Console.WriteLine("Nope number too high!");

Console.ReadLine();

}

else

{

Console.WriteLine("Nope number too low!");

Console.ReadLine();

}

This works but there is a problem when the user does guess the correct number. The program tells us that the number is too low. To fix this lets use the if/else if statement.

if (numberGuess > myNumber)

{

Console.WriteLine("Nope number too high!");

Console.ReadLine();

}

else if (numberGuess < myNumber)

{

Console.WriteLine("Nope number too low!");

Console.ReadLine();

}

else

{

Console.WriteLine("That is right!");

Console.ReadLine();

}

Our program now work right. Always remember that the program will stop on the first if statement it comes to that is true, so plan out you code before you start so that you have less bugs.

At this point it may be a good idea to talk about Logical Operators.

&& - And

|| - OR

! – Not

You can have more than one Expression in an if statement

If (num1 == 1 && num2 == 2)

Both Expressions must be true

If (num1 == 1 || num2 == 2)

Only one needs to be true

If (! (num1 < num2)

The Expression inside the second parenthesis cannot be true for the if statement to be true.

A Switch statement lets you compare a variable a little more easily.

string myNumber = "Five";

switch (myNumber)

{

case "One":

Console.WriteLine('1');

break;

case "Two":

Console.WriteLine('2');

break;

case "Three":

Console.WriteLine('3');

break;

case "Four":

Console.WriteLine('4');

break;

case "Five":

Console.WriteLine('5');

break;

default:

break;

}

Console.ReadLine();

The Switch takes the variable and checks it value to the case value then it will execute the code in that case.

All these will work, however maybe the program needs to check more expressions to determine what to do. TYPE OUT NEST IF ON PAGE 220. Talk about what is doing.