Creating Loops

C# Programming

What we can do so far

- Store data (using variables)
- Change data (using assignments)
- Make decisions (using conditions)
- There is not much more that we need to know how to do
  - But we do need to know how to create loops

Loops

- We create a loop so that we can repeat one or more statements
- A condition is used to determine whether or not the loop stops
- The condition is either true or false, just like that used in an if construction

A Stupid Loop

- We can write never ending loops if we like:

```
do
  Console.WriteLine ("Har har");
while (true);
```

- This loop will never finish (use CTRL+C to kill a program if it does this..

The do – while loop

- do-while continues while the condition is true

```
int i = 0;
do {
  Console.WriteLine (i);
  i = i + 1;
} while ( i < 4 );
```

- We can use a block to get more than one statement repeated

Another Stupid Loop

- We can write "non loops" if we like:

```
do
  Console.WriteLine ("Har har");
while (false);
```

- In this case the loop will not repeat, but it will execute once as the test is at the end
- Remember that statements are executed in sequence

Doing the test at the end of the loop

- In the do – while loop the test to see if the loop continues is performed **after** the statements in the loop have been performed
- This is useful if you want the code to do something and then check the result
  - For example if you were reading numbers in from a user..

Reading in Numbers

```
do {
  Console.Write("Enter width:");
  widthString = Console.ReadLine();
  width = double.Parse(widthString);
} while ((width<0) || (width>3.0));
```

- This will repeatedly read the width value until a valid one is entered
- Make sure you test with invalid values too

Doing the test first

- Sometimes you want to do the test before you perform the loop code
- There is a C# construction for this too:

```
while (false)
  Console.WriteLine("Never Printed");
```

- Note that the word do is not required
- Note that the statement could be a block

For loops

- We have already seen how we can create code which will repeat something a particular number of times
- However, since this is something that we need to do a lot, C# provides a special constructions for this, the for loop

The For loop

- The for loop has the following form:

```
for ( setup ; finish test ; update ) {
      // things we want to do a given
      // number of times
}
```

- The setup, finish test, and update are added to get the loop that we want

A working For Loop

```
int i ;
for ( i = 1 ; i < 11 ; i = i+1 )
{
  Console.WriteLine ( "Hello" ) ;
}
```

- This will print out Hello 10 times
- When the value in `i` reaches 11 the loop stops

A stupid For Loop

```
int i ;
for ( i = 0 ; i < 11 ; i = i-1 )
{
  Console.WriteLine ( "Hello" ) ;
}
```

- This will print out Hello for ever because the control variable is updated in the wrong direction

Another stupid For Loop

```
int i ;
for ( i = 0 ; i < 11 ; i = i+1 )
{
  Console.WriteLine ( "Hello" ) ;
  i = 0;
}
```

- This will print out Hello for ever because the control variable is reset in the code inside the loop

Breaking out

```
int i ;
for ( i = 0 ; i < 11 ; i = i+1 )
{
  Console.WriteLine ( "Hello" ) ;
  if ( i==3 ) break;
}
```

- The `break` keyword lets us escape from any loop
- You can use it in do-while, while and for loops

Continuing

```
int i ;
for ( i = 0 ; i < 11 ; i = i+1 )
{
  Console.WriteLine ( "Hello" ) ;
  if ( i==3 ) continue;
  Console.WriteLine ( "Not 3" ) ;
}
```

- The continue keyword takes us back to the "top" of any loop

Summary

- We now have the three fundamental loop constructions
- The trick with programming is to use the construction which is appropriate to the task in hand
- You can make the code work with any loop design