UNIVERSITY OF Hull

Neater Printing

C# Programming

UNIVERSITY OF Hull

Improving Print Formatting

- At the moment we have been using the default printing behaviours which are somewhat limited
- The C# printing library provides additional ways to control the printing process
- We are going to investigate these here

UNIVERSITY OF Hull

Simple Printing

```
int i;
i = 99;
Console.WriteLine (i);
```

- The `Write` and `WriteLine` methods provide a way of getting the values of variables onto the screen
- The simple string version of the item is created and displayed

UNIVERSITY OF Hull

Complicated Printing

```
Console.WriteLine( "X is " +
x + " and y is " + y + ".");
```

- If we want to merge text and values we have to use the + operator to concatenate the strings
- This can make the write statements look quite complicated

UNIVERSITY OF Hull

Using Placeholders

```
Console.WriteLine(
"X is {0} and Y is {1}.", x, y);
```

- You can use placeholders in a string being written
- These are replaced by the values of the given variables when they are printed
- The items are numbered starting at 0

UNIVERSITY OF Hull

Controlling the Print Behaviour

```
double f = 1234.56789;
Console.WriteLine(
         "f: {0:000000.00}", f);
```

- Placeholders can have additional information added to them
- This would output the value of f with 6 digits and two decimal places:

```
F: 001234.57
```

UNIVERSITY OF Hull

Suppressing Leading Zeroes

```
double f = 1234.56789;
Console.WriteLine(
         "f: {0:#####0.00}", f);
```

- If you use # rather than 0 this prints a space if the digit is a leading zero
- Note that we need at least one zero so that the value 0 is printed correctly:

```
F:    1234.57
```

Neater Printing 9-Nov-12 ©Rob Miles 7

UNIVERSITY OF Hull

Adding other Formatting

```
double f = 1234.56789;
Console.WriteLine(
         "f: {0:###,##0.00}", f);
```

- By putting extra characters in the format string we can add commas for thousands
- If the leading digits are spaces the extra characters are not printed:

```
F:   1,234.57
```

Neater Printing 9-Nov-12 ©Rob Miles 8

UNIVERSITY OF Hull

Setting the Print Width

```
double f = 1234.56789;
Console.WriteLine(
      "f: {0,15:###,##0.00}", f);
```

- You can add a width value which will cause the output to be right justified that width
- In the example the width is 15 characters:

```
F:        1,234.57
```

Neater Printing 9-Nov-12 ©Rob Miles 9

Left Justifying the Output

```
double f = 1234.56789;
Console.WriteLine(
      "f: {0,-15:###,##0.00}E", f);
```

- If you give the width as a negative value the number is left justified
- In the example the width is 15 characters:

```
F:1,234.57       E
```

Formatting Integers

```
int i = 5;
Console.WriteLine(
      "i: {0,-15:##00}X", i);
```

- Integers are formatted exactly as floating point values, but without the decimal places:

```
F:05            E
```

Summary

- The `Write` and `WriteLine` methods can be given formatting instructions when they output a value
- This allows for great flexibility when producing printed output