Arrays

C# Programming
Using Arrays

What we can do so far...

- Store data (using variables)
- Change data (using expressions)
- Make decisions (using conditions)
- Create loops (using do – while and for)
- There is not much more that we need to know how to do
  - But we do need to know how to create arrays

Variables

- We have a reasonable idea of how to create a variable:

```
int sales;
```

- This will create a variable which can hold a single integer value
- The variable has the identifier **`sales`**

Storing a Sales Value

- Once we have a variable we can assign values to it

```
sales = 5;
```

- This sets the values of the sales achieved to a rather poor 5

Handling more data

- If we want to store more data, the simplest approach is to create more variables:

```
int sales1;
int sales2;
int sales3;
int sales4;
```

Storing more sales

- If we want to store more data, the simplest approach is to create more variables:

```
sales1 = 5;
sales2 = 10;
sales3 = 0;
sales4 = 30;
```

UNIVERSITY OF Hull

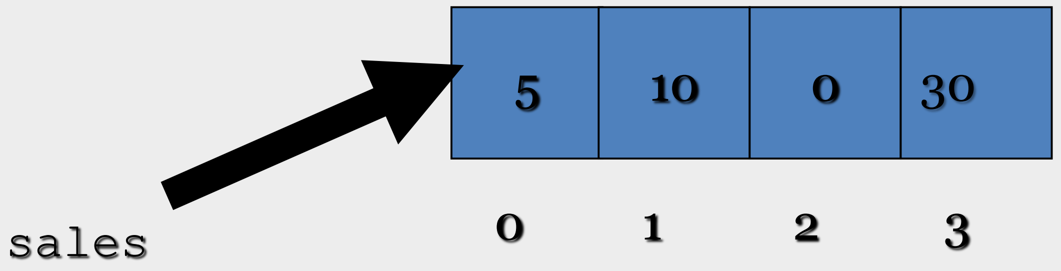
Manipulating data

- However, this makes the data hard to work with:

```
if ((sales1 > sales2) &&
    (sales1 > sales3) &&
    (sales1 > sales4) )
{
   Console.WriteLine (sales1);
}
```

Arrays

- An array lets us create a row of variables which we can *index* using a *subscript*



- **`sales`** is a reference to an array of integers which contains 4 elements

Creating an Array

- When you create an array you must say how many elements it is going to hold

```
int [] sales = new int [10];
```

- The keyword new is how I request the creation of new objects
- This creation takes place at run time

Using an Array

- Once you have your array you can place values into the elements in it

```
int [] sales = new int [4];
sales [0] = 5;
sales [1] = 10;
```

- The value in the square brackets is called a *subscript*
- Note that the initial element has a subscript of `0`

Subscripts Etiquette

- Subscripts start at 0
- If you try to access an element which is not in the array (perhaps by using a subscript which is too large) your program will fail
- Subscripts are checked as your program runs so that our programs never "fall off the end of an array"

The power of subscripts

- Subscripts become very powerful when we discover that we can use a variable as a subscript:

```
int [] sales = new int [4]  ;
for ( int i=0; i<4; i=i+1 )
{
   string salesString = Console.ReadLine();
   sales [i] = int.Parse(salesString );
}
```

- This will read in and store 4 sales values

The real power of subscripts

```
int maxSales =0 ;
for ( int i=0; i<4; i=i+1)
{
   if (sales [i] > maxSales )
   {
      maxSales = sales[i];
   }
}
```

- This will find the largest sales value in the array

UNIVERSITY OF Hull

Sensible Design

```
const int SALES_SIZE;
int [] sales = new int [SALES_SIZE]  ;
for ( int i=0; i< SALES_SIZE; i=i+1) {
   string salesString = Console.ReadLine();
   sales [i] = int.Parse(salesString );
}
```

- It makes sense to use constant values to set the size of the array and the limits of the loop

Two Dimensional Arrays

- You can add an extra dimension by creating another subscript:

```
int [,] board = new int [3,3];
board [1,1] = 1;
```

- The subscripts are now row and column values
  - – This is how spreadsheets work

More than two dimensions

- You can have as many array dimensions as you like
  - But my brain starts to hurt if you go beyond 3
- If you find yourself using lots of array dimensions you are probably not approaching the problem correctly

Changing Array Sizes

- It is not possible to change the size of an array once it has been created
- If a different storage size is required the program must create a new array
- However, you can use a variable to set the size of an array

Summary

- Arrays are the last thing that we need to know how to write every program in the world
- The allow us to store huge amounts of data and search and sort it
- The key to the power of an array is the use of variables as subscripts