Department of Computer Science

08101 Programming I

Week 5 Laboratory 2013/2014

Before you go onto this lab, please make sure that you have sorted out the Visual Studio program from last week and got your program working in Visual Studio.

Using Arrays

In this laboratory we are going to explore how to use an array to store data, and how we can avoid having to "hard wire" items into programs.

```
Welcome to our Multiplex
We are presently showing:
    1. Rush (15)
    2. How I Live Now (15)
    3. Thor: The Dark World (12A)
    4. Filth (18)
    5. Planes (U)
Enter the number of the film you wish to see: 1
Enter your age: 12
Access denied – you are too young
```

Above you can see the menu from the cinema entry program. At the moment we are creating the menu by writing out the individual film items:

```
Console.WriteLine("    1. Rush (15)");
Console.WriteLine("    2. How I Live Now (15)");
Console.WriteLine("    3. Thor: The Dark World (12A)");
Console.WriteLine("    4. Filth (18)");
Console.WriteLine("    5. Planes (U)");
```

This will work, but it means that every time the films change we have to make a new program. What the program should do is read in the film names at the start, and then display them for each customer. Later we can make the program read the names from a file, or perhaps even a web feed.

Creating a New Project

1. Log in and start up Visual Studio 2012..
2. Use **`File>New>Project`** to open the New Project dialogue. The dialog you see might not exactly match the one below, but it will have a C# type which contains a Windows Console Application template. Be careful not to create a Visual Basic project as this would be very confusing.
3. Give the project the name **`CinemaEntryArray`**

Adding a film array

We are going to use an array of strings to hold the names of the films it will hold 5 names:

```
string[] filmNames = new string [5];
```

This statement creates an array which can hold five strings.

> If we wanted to handle 7 films, what would we change? Can you see a need for a "magic number" here?

Reading in the film names

Now that we have an array we can read in some film names. The best way to do this is to use a loop that goes round five times. We can use the control variable (the counter) in the loop to access each element in the array in turn.

```
// Read in the film names
for (int i = 0; i < 5; i = i + 1)
{
    int displayNumber = i + 1;
    Console.Write("Enter the name of film number " + displayNumber + ": ");
    filmNames[i] = Console.ReadLine();
}
```

This loop will read in the names. Note that C# will let us declare a control variable (in this case the integer variable `i`) when the loop is set up. This variable will exist for the entire loop body.

> We know that the array is indexed from 0 (in other words the first element in the array has the subscript of zero. However, users would not expect to be counting from 0 in situations like this. Can you see how the programmer has dealt with this?

> 4. Add the declaration of the array and the loop that reads in the film names. Run the program. Of course, at the moment it will not print anything because we are just reading in the names.

Displaying the Menu

Once you have the array set up in the program you can use it to display the menu from which the user will select the film that they want to see. You can use exactly the same for-loop structure before (but you have to make a **new** for loop to display the menu).

> 5. Add a for loop to print out the names that were entered.

Finding the selected film

The user will enter the number of the film that they want to see. However, they will start counting at 1, whereas the film at the start of your array has the subscript 0. This means that you will have to subtract 1 from the number they enter, to find the name of the film that they have selected.

Handing the film age ratings

This seems to work well, but at the moment it looks like we have no way of handling the age ratings of the films. Since that is what the program was created for, we need to sort this out.

It turns out that the C# string type has a really useful method called `EndsWith`. You can use it to see if a string ends with a particular piece of text. This is just what we want, since the age rating is given right at the end of the film name. We will use a construction like this:

```
if ( filmNames[chosenFilm].EndsWith("(12A)") )
{
    // The user has selected a film that is 12A rated
}
```

In the code snippet above the variable `chosenFilm` holds the number of the film that was selected by the customer. This will be an integer in the range 0 to 4 (remember that arrays are indexed from 0). The number was obtained by asking the user for the number of the film they want to see, and then subtracting 1 from it.

The if condition looks for the string `(12A)` on the end of the name, so that it can detect films of that rating.

6. Use the above technique to allow the correct film age ratings to be selected. Your program can use the age rating information on the end of each film name rather than having the ratings for each film “baked in” to the program itself.

Tidying Up

You can use the above techniques to make the ultimate Cinema Entry program. Here are some thoughts to make it even better.

Use the Length property of the array to control your loops

At the moment you are counting up to 5 each time you work through the array. It turns out that an array instance has a Length property that you can use. It tells the program how many elements the array contains:

```
for (int i = 0; i < filmNames.Length; i = i + 1)
```

This means that if the size of the array changes (for example if more films are shown one week) the program will still work correctly.

Allow the user to enter how many films are being shown

A C# array can be dynamically created, i.e. you can set the size of the array when the program runs. This means that you could ask the user how many films that are being shown and then make an array of the required size. This would also make your program quicker to test, because you could enter the value 1 and just enter a single film name.

Add data tidying error checking

The technique of using `EndsWidth` to get the age ratings will work, but if the user doesn't type the rating value correctly, or adds a space on the end of the film name by mistake the rating behaviour will not work as it should.

There is a method called `Trim` that you can use obtain a version of a string with all the leading and trailing spaces removed:

```
trimmedName = enteredName.Trim();
```

The above statement sets the variable `trimmedName` (which must be a string) the value of `enteredName` (which must also be a string) with all the leading and trailing spaces removed.

You should also make sure that when the user enters the film names you check to make sure that each has a sensible age rating. You will have to add some tests to do this. One way would be to make a method called `validateRating` which accepts a string and returns if the name is a valid one. Here is an empty version of the method to fill in. Note the comments

```
/// <summary>
/// Validates the film name to make sure
/// the name ends with a valid rating string:
///
/// (15) (12) (18) (U) or (12A)
/// </summary>
/// <param name="filmName">name of the film to be checked</param>
/// <returns>true if the film name is valid</returns>
static bool validateRating(string filmName)
{
    // need to fill in this code
    return false;
}
```

If you enter this method as above you will find that interesting things happen in Visual Studio when you use it in your programs. If you want to add these comments to any methods that you write you can produce the template by typing /// (three forward slashes) in the editor just above the method.

Lab Summary

Don't forget to go onto eBridge and do the summary test for this lab.

Rob Miles
October 2013