Department of Computer Science

08101 Programming I

Week 2 Laboratory 2013/2014

More Complex Programs

We now know how to create simple C# programs and make them run. In this session you are going to develop the programs that have already been written, and do some more complicated data processing. Note that you will now have to invent your own code, as opposed to typing in what we provide.

Making Decisions

The program that we wrote last week simply runs to completion. It reads in some data, performs a calculation and then prints out the result. In this respect it is working just like a pocket calculator. However, most programs also need to have the ability to make decisions. This is one of the things which makes computers truly useful, in that your program can react to different types of data in an appropriate way.

Cinema Entry Problem

As an example of conditions in action we could consider a program which decides whether or not a particular person is allowed to go and see a film. If the age of the user and the film match up the program will print "Enjoy the film" otherwise it will print out "Access Denied - You are too young". Note that the system does not have any way of verifying the age, we are taking on trust that the user is not telling fibs! The first program we are going to write will perform the test for the film "Rush" which has a minimum age requirement of 15 years. The program will have the following user interface:

```
The film you want to see is Rush
Enter your age: 21
Enjoy the film
```

What the user types in is shown as underlined.

Test Data

Before you can be sure that your program will work you must make sure that you have some test data to use with it.

Before you go any further; perform the following:

1. Write down some test data (four values should be sufficient) to make sure that your program works (you should pick values which are close to the test boundary and others which are not).

Now that you have some test values you can write the code which does the testing. You will need to make use of the C# language construction which can make decisions. Otherwise known as the "if" construction.

Writing a Program

Now you can write the program.

Before you go any further; perform the following:

2. Log on to a workstation using your username and password.
3. Start a command prompt and find your way to the G: drive.
4. Create a directory to store the files for this week:

   `md 08101lab02`

5. Use the **`cd`** command at the command prompt to move into your work directory.
6. Use the notepad program to create a program in the file **`Cinema.cs`** which works as required. A good starting point is to use the copy command to make a copy of your **`Sums.cs`** program from last week. This already has C# code which will print out messages and read numbers from the user.

Completing the Solution

We can now go ahead and create the final version of the program. This will print a menu of films from which the user will choose the one they want to see. The user will then enter their age. The program will be used like this:

```
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
```

Your program will have to read and store the number of the required film. It will then read and store the age. You must then use an appropriate arrangement of conditions to decide whether or not the customer can see the film.

Before you go any further; perform the following:

1. Devise a set of test cases which will allow you to test your program. There should be two test cases per film, one which works and one which does not.
2. Modify the **`Cinema.cs`** program so that it works as above and use your test data to prove this.

Input Validation

At the moment the program is not perfect. It could be made to do stupid things. There are a number of ways in which a user could upset your program:

- enter an age value which is stupidly large
- enter an age value which is stupidly small
- enter a film value which is stupidly large
- enter a film value which is stupidly small

You will need to devise some "metadata" which describes the age and film values and what constitute valid values, as an example, the film number should never go below 1.

Before you go any further; perform the following:

1. Devise the valid values for the age and film values in your program.
2. You can add them as comments to the code as follows:

```
int FilmNo; // must not be less than 1 or bigger than 8
```

3. Add the comments to your program and improve the code so that values outside the given ranges are rejected with an appropriate message.

eBridge Test

Now do the eBridge test for this week. Don't worry; the scores are not used as part of your mark for this module, but to help with the tutorials later today.

A program of your own

For the remaining part of the laboratory you must create a program which works in a similar way to the cinema entry one. Pick one of the situations below and create a program with a sensible name which behaves in the required manner:

Grading potatoes

A program to select the grade letter of a given potato. (Note that these grades are made up; you may be able to find proper potato weights on the web). The user will enter the weight of the potato and the grade will be displayed.

1. Less than 200 gms – grade X
2. Between 200 and 400 gms – grade A
3. Between 400 and 800 gms – grade B
4. Above 800 gms – grade Z

Exam scores and degree classifications

A program to print out the degree classification (or not) of a candidate with a particular score. The user will enter the mark and the program would print out the classification:

1. Less than 35% - fail
2. Between 35% and 40% - can be "compensated"
3. Between 40% and 50% - third class degree - III
4. Between 50% and 60% - lower second class degree – II(ii)
5. Between 60% and 70% - upper second class degree – II(i)
6. Above 70% - first class degree - I

Boxing Weights

A program to allow the user to determine the weight classification of a boxer. The user will enter the weight of the boxer in pounds and the program will print out the classification:

1. Heavyweight: Unlimited
2. Cruiserweight: Limit - 190 pounds
3. Light Heavyweight: Limit - 175 pounds
4. Super Middleweight: Limit - 168 pounds
5. Middleweight: Limit - 160 pounds
6. Junior Middleweight: Limit - 154 pounds
7. Welterweight: Limit - 147 pounds
8. Junior Welterweight: Limit - 140 pounds

9. Lightweight: Limit - 135 pounds
10. Junior Lightweight: Limit - 130 pounds
11. Featherweight: Limit - 126 pounds
12. Junior Featherweight: Limit - 122 pounds
13. Bantamweight: Limit - 118 pounds
14. Junior Bantamweight: Limit - 115 pounds
15. Flyweight: Limit - 112 pounds
16. Junior Flyweight: Limit - 108 pounds
17. Strawweight: Limit - 105 pounds

Your program should assume that a boxer will always fight in the lightest classification their weight allows, i.e. someone who weighs 104 pounds could enter a Heavyweight contest, but I don't think they would enjoy it...

Rob Miles
07/10/12