Department of Computer Science

08101 Programming I

Week 6 Laboratory 2013/2014

Before you go onto this lab, please make sure that you have sorted out the array exercise from last week.

Processing Cricket Player Scores

This week you are going to perform some simple processing on the scores produced by a team after a cricket match.

For each batsman the following information will be entered:

- name of batsman
- score achieved in runs

Every member of the team must bat, so 11 scores will be entered into the program, along with 11 names. The customer is insisting that the program does not accept invalid inputs. You have been told:

- The scores will be entered at the start of the program. There are eleven scores.
- The batsmen will be referred to by name. No batsman may have an empty name.
- The lowest score which can be achieved by a batsman is 0. This is also called a “duck”.
- The highest score which the program should accept is 500, i.e. if the user tries to enter a score which is greater than this value it must be rejected.
- The average score is the total innings score divided by 11.
- If there are any other errors you can think of, your program should handle them too.

For example (this is just a sample format, you can arrange the input out how you like):

```
Rob
150
Jim
0
Ethel
2500
Score too large. Please enter a score between 0 and 500
25
```

This means that Rob scored 150 runs, Jim scored 0 runs (a duck) and Ethel scored 25 runs. One invalid score was rejected. You should use the number reading methods you have created in earlier weeks to read the numbers in and reject invalid ones.

Creating a Player structure

```
struct Player
{
    public string Name;
    public int Score;
}
```

You should create a structure to store the score information about each player and then create an array which will hold all 11 score values. Refer to the lecture notes on SharePoint for more details on how to do this, and how to store the scores in each element of the array.

Sorting Cricket Scores

Once all the information has been entered the program should print out a list of the cricket scores, sorted by the score value with the largest score first. You can use Bubble Sort to perform the sort. Bubble Sort works by comparing adjacent elements in the array and swapping them around if they are in the wrong order. Refer to the tutorial notes on SharePoint for more details on how to do this.

Selecting the Sorting option

Once you have completed the sorting by score, you must now modify the program so that it can also sort by the names of the players. After the scores have been entered the program should ask the following question:

```
Do you want to sort by name or by scores? (enter N or S): N
```

If the user types N the program will sort by the names of the players. The greater than (>) and less than (<) operators don't work between strings, but you can use the `CompareTo` method to compare one string with another:

```
string a = "Fred";
string b = "Jim";
if (a.CompareTo(b) < 0)
{
    Console.WriteLine("Fred is before Jim");
}
```

In the code above the string `a` is comparing itself with `b` using the `CompareTo` method. The `CompareTo` method compares the string with another and returns -1, 0 or +. It returns -1 if the string doing the comparing (in the above example `a`) is before the other (in the above example `b`) in alphabetical ordering. If the two strings are equal `CompareTo` returns zero. If the string doing the comparing is after the target string the `CompareTo` method returns +1. The above code snippet would print the message because Fred is before Jim in the alphabet.

Modify your program so that it prints out a list of players sorted by name or score.

Rob Miles

November 2013