COMP315

Individual Project Documentation

|  |  |
| --- | --- |
| **Members** | |
| 1. Avron Ramlakan | 217044638 |
|  |  |

Contents

[Introduction 3](#_Toc47817710)

[Screenshots 3](#_Toc47817711)

[Programming Techniques 6](#_Toc47817712)

[1. Function 6](#_Toc47817713)

[2. Class 9](#_Toc47817714)

[3. Reference 10](#_Toc47817715)

[4. Data Structure 13](#_Toc47817716)

[5. Vector 14](#_Toc47817717)

[6. Function Template 16](#_Toc47817718)

[7. Operator Overloading 17](#_Toc47817719)

[Score Calculation 18](#_Toc47817720)

[Additional Item 18](#_Toc47817721)

[Additional Item 1 – High-score 18](#_Toc47817722)

[Your additional feature 18](#_Toc47817723)

[Additional Item 2 – Progress Bar 19](#_Toc47817724)

[Your additional feature 19](#_Toc47817725)

[Appendix 20](#_Toc47817726)

# Introduction

A console based quiz was created to test “true fans” of the Premier League. The English Premier League is the most popular and liked football league in the world. The data was obtained from the Premier League website. The quiz will ensure that players put on their thinking caps to test their knowledge of past facts and records of the premier league. The quiz consists of Multiple-choice questions, with hints available if requested, and a score accumulator, and high-score so that players can view scores saved by previous players.

# Screenshots

|  |  |
| --- | --- |
| **Screenshot** | **Explanation** |
| C:\Users\Avron\Pictures\Program\1 - Enter Name.png | The first request is to allow the player to enter their name. |
| C:\Users\Avron\Pictures\Program\2 - Menu.png | The player will be welcomed to the main menu, where they could either enter ‘Q’ to start their quiz, or ‘H’ to view the High-scores. Also ‘E’ to exit. All other inputs will not do anything, and the Main menu will remain. |
| C:\Users\Avron\Pictures\Program\3 - Wrong Answer.png | If the player requests to start the quiz, by entering ‘H’, then the following would be shown:  - the player name is shown  - the question that the player is on, and how many questions are to be asked.  - a score tally  -a progress bar in square brackets, where the ‘\*’ represents the number of questions complete, and the ‘–‘ represents the number of questions left. The percentage of progress is also shown.  - The player has an option to ask for a hint, by entering ‘H’.  As shown on the screenshot, if the incorrect answer is entered, the player will be notified that it’s incorrect, and the correct answer will be revealed. |
| C:\Users\Avron\Pictures\Program\4 - Hint & wrong.png | If the player request for a Hint by entering ‘H’, as shown in the screenshot, then a hint will be revealed. |
| C:\Users\Avron\Pictures\Program\5 - Hint & Correct.png | If the player enters the correct answer, then they’ll be notified that it’s correct. I’ve only included 3 questions in total, to make it more convenient for my running and testing purposes, but I can add up to 20 questions (which is the array length that I set) in the text files (which is what I used to store the question). |
| C:\Users\Avron\Pictures\Program\6 - Final Score.png | When complete, the final score will be shown. The player has the option to save their score among the high-scores, by entering ‘S’. |
| C:\Users\Avron\Pictures\Program\7 - Enter H.png | At the main menu, if the player enters ‘H’, then he will be shown the high-scores. |
| C:\Users\Avron\Pictures\Program\8 - Highscores.png | The high-scores are set sorted in descending order. As seen in the screenshot, Arnold’s scores are saved, as requested in the previous screenshots. |
| C:\Users\Avron\Pictures\Program\9 - Enter E.png | If ‘E’ is entered, then the console application is terminated, and the player can exit. |
|  |  |

# Programming Techniques

## Function

|  |
| --- |
| **Screenshot:**  **C:\Users\Avron\Pictures\Screenshots of Project\ReadTxt().png**  C:\Users\Avron\Pictures\Screenshots of Project\printProgressBar.pngC:\Users\Avron\Pictures\Screenshots of Project\writeToText.png  C:\Users\Avron\Pictures\Screenshots of Project\SortVect.png  C:\Users\Avron\Pictures\Screenshots of Project\readTxtToVector.png |
| **Motivation:**  I’ve used 5 functions in total which are readTxt, printProgressBar, writeToText, sortVect, and readTextToVector. Of course the primary advantage of a function is that you prevent having to copy the same chunk of code, if you need the same process to be execcuted again. With readTxt and readTextToVector, I take in the textfile name, and an array/vector as paramaters, and I would be able to copy the data from the text file into the array or vector. With writeToText, I could take in a variable of T type (template was used), and I could write to the textfile as indicated by it’s name which is in the argument. printProgressBar prints the progress bar together with the percentage, and that function is called after every question, and it takes the number of correct answers, and the total questions as parameters. sortVecter takes in a vector, and uses selection sort to sort the vector so it could be shown in descending order when the high-scores are shown. All these functions are called more than once, therefore they’re crucial. |

|  |  |  |
| --- | --- | --- |
| **How have you met the objectives?** | **Cross (X) the appropriate box** | **If you think that you have met the objective completely, provide a short explanation to support the claim** |
| Not met |  |
| Partially |  |
| Completely | x | I’ve used 5 functions to prevent having to copy code at numerous places in the main |

## Class

|  |
| --- |
| **Screenshot:**  C:\Users\Avron\Pictures\Screenshots of Project\Quiz Class.png  [*Provide a screenshot of your code. This will be code of a class in your game*] |
| **Motivation:**  I’ve used only one class in total, which is the Quiz class. This class has a name variable of type string (which stores a player name), and a correct variable of type int (which stores the score the player obtains). One method is included, which is used to increment the correct variable (called each time the player gets the answer correct). This Quiz class was really useful because it keeps my code “clean”, and especially more so because of the vector I used which was of type Quiz. The vector stored all the names and correct values that’s saved by previous players. I could have easily used a struct instead, and wouldn’t have the need to declare public: , but from previous Java programming experiences, I’m just so comfortable to the use of classes. |

|  |  |  |
| --- | --- | --- |
| **How have you met the objectives?** | **Cross (X) the appropriate box** | **If you think that you have met the objective completely, provide a short explanation to support the claim** |
| Not met |  |
| Partially |  |
| Completely | x | I’ve used one class in total, and there isn’t really a need for more. This Quiz class is especially useful because of the vector of type Quiz. |

## Reference

|  |
| --- |
| **Screenshot:**  **C:\Users\Avron\Pictures\Screenshots of Project\ReadTxt().png**  C:\Users\Avron\Pictures\Screenshots of Project\readTxtToVector.png  C:\Users\Avron\Pictures\Screenshots of Project\SortVect.png  C:\Users\Avron\Pictures\Screenshots of Project\Operator Overload.png |
| **Motivation:**  References (&) was only used in my code to pass by reference. Majority of my functions parameter had to pass by reference. This was extremely useful because it avoided having to create unnecessary copies of those variables, I could just directly modify the variable that comes in as a parameter. |

|  |  |  |
| --- | --- | --- |
| **How have you met the objectives?** | **Cross (X) the appropriate box** | **If you think that you have met the objective completely, provide a short explanation to support the claim** |
| Not met |  |
| Partially |  |
| Completely | x | I’ve used pass by reference in almost all my functions, so I can directly modify the parameter. |

## Data Structure

|  |
| --- |
| **Screenshot:** **C:\Users\Avron\Pictures\Screenshots of Project\main, variables, and main menu display.png** |
| **Motivation:**  The only data structure I’ve used was arrays. That was the most efficient from them all because I could iterate through it and access any specific value in the data structure, unlike other data structures like queues and stacks. I stored all my Questions, Answers, and Hints in Arrays. I set them to store a maximum of 20 values, because I was not planning on having more questions, and that’s a reasonable number of memory addresses being taken by the arrays. If I had used a stack or queue, then would not be able to ask randomize the order of the questions. |

|  |  |  |
| --- | --- | --- |
| **How have you met the objectives?** | **Cross (X) the appropriate box** | **If you think that you have met the objective completely, provide a short explanation to support the claim** |
| Not met |  |
| Partially |  |
| Completely | x | Arrays was the most efficient data structure in terms of iteration, because I could access variables at any index. There simply was no need for me to use any other data structure unnecessarily. |

## Vector

|  |
| --- |
| **Screenshot:C:\Users\Avron\Pictures\Screenshots of Project\main, variables, and main menu display.pngC:\Users\Avron\Pictures\Screenshots of Project\SortVect.pngC:\Users\Avron\Pictures\Screenshots of Project\readTxtToVector.png** |
| **Motivation:**  A Vector was the important aspect to my additional feature (displaying high scores). I used only one vector which was of type Quiz, and this because I could store various instances of Quiz, which will store “names”(player names) and “correct”(their score). All the names and scores were stored in a text file, and the readTxtToVector function placed those values into Quiz instances stored in the vector. A vector was best suited to this because player scores being saved are always incrementing, so there’s no set number of scores, and therefore a vector would be perfect because it’s not limit to a set number of index. |

|  |  |  |
| --- | --- | --- |
| **How have you met the objectives?** | **Cross (X) the appropriate box** | **If you think that you have met the objective completely, provide a short explanation to support the claim** |
| Not met |  |
| Partially |  |
| Completely | x | I used a vector to keep instances of Quiz (which keeps name and correct), and these increment as new player saves their score. There was no need to use more vectors in my program. |

## Function Template

|  |
| --- |
| **Screenshot:**  **C:\Users\Avron\Pictures\Screenshots of Project\writeToText.pngC:\Users\Avron\Pictures\Screenshots of Project\ReadTxt().png** |
| **Motivation:**  Function template was used on two occasions in my program. It was crucial as for one, in the writeToText method, any type of text could be written into the textfile (in my case, score is int and name is string), therefore I could not limit the type of text that’s being written into the text file, and two, in the readTxt function, when reading values from text files, it could be of any type, so therefore I allowed the array in the argument to be of any type (T). For eg. the MCQ answers are char values, so I could have stored them in a char array. Function templates are of utmost importance to prevent restriction to one data type. |

|  |  |  |
| --- | --- | --- |
| **How have you met the objectives?** | **Cross (X) the appropriate box** | **If you think that you have met the objective completely, provide a short explanation to support the claim** |
| Not met |  |
| Partially |  |
| Completely | x | I used function templates numerous times to ensure I’m not restricted to a variable or array being one data type. |

## Operator Overloading

|  |
| --- |
| **Screenshot:**  **C:\Users\Avron\Pictures\Screenshots of Project\Operator Overload.png**  [*Provide a screenshot of your code. This will be code of operator overloading in your game*] |
| **Motivation:**  Having been previously programming in Java and Python, one of the most amazing aspects of coming into c++, is operator overloading. I used it to overload the ‘<<’ operator of the cout. This allowed me to directly output all the values of my vector of type Quiz, so basically instead of creating a new print function, I just overloaded the ‘<<’ operator to do the same thing, and I used that to print all the names and scores recorded by previous players. |

|  |  |  |
| --- | --- | --- |
| **How have you met the objectives?** | **Cross (X) the appropriate box** | **If you think that you have met the objective completely, provide a short explanation to support the claim** |
| Not met |  |
| Partially |  |
| Completely | x | I used it once to conveniently print out all the names and scores. |

# Score Calculation

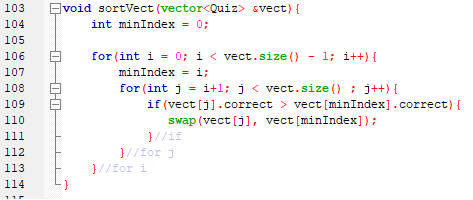
My quiz keeps track of the players score. As an additional item, the player is allowed to save his name and score, and then view it amongst others who have also saved it. This display is done according to descending values of correct, therefore it’s called a high-scores display. The score works as follows, an instance of the Quiz class is first created, and this stores the player name, and initially sets the ‘correct’ variable to 0. When the player decides to start his quiz, for each question he gets correct, the ‘correct’ variable is incremented by one. At the end, the player score is displayed, and he has the option to save it. Before he returns to the main menu, the correct variable resets to 0, to cater for the same process, if the player decides to take the quiz again.

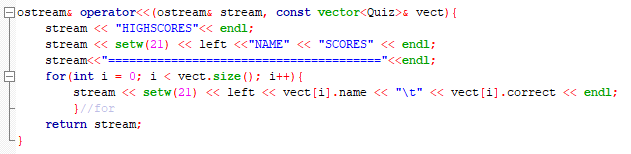
# Additional Item

|  |  |
| --- | --- |
| **What does your quiz include?** | **Cross (X) the appropriate box** |
| **Display of player high-scores, and option to save a players score among the high scores, if they require** | **x** |
| **Progress Bar & Percentage of questions complete** | **x** |
|  |  |

## Additional Item 1 – High-score

### Your additional feature

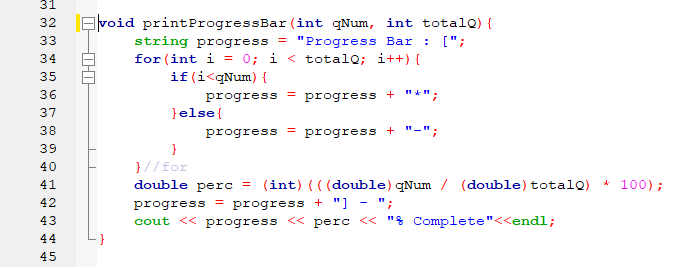




The first additional item is a high-score display, which displays saved scores in descending order. This option is to give a competitive feeling towards the players, allowing the player to view their score amongst others who have saved their scores. All I do in the code is sort the vector as shown in the first screenshot, and then display the vector with the overloaded “<<” operator, as provided by the second screenshot.

## Additional Item 2 – Progress Bar

### Your additional feature



The second additional item is a progress bar, which displays the a progress bar comprising or ‘\*’(indicating number of questions complete) and ‘-‘(number of question left, including the current question) in a square bracket. For eg. if the player is on question 6 out of 6 then the progress bar would be [\*\*\*\*-]. It also include the percentage of questions complete. This progress bar has been included to give the user a visible sense of their completion and question progress.

# Appendix

My full main code is below, which means that this document has all my code of the program.

