BMI Calculator

Jessica Fealy

Table of Contents

[Main 2](#_Toc26983810)

[Main Diagram 4](#_Toc26983811)

[addUserList 5](#_Toc26983812)

[saveToFile 7](#_Toc26983813)

[userSelect 8](#_Toc26983814)

[listUserNames 9](#_Toc26983815)

[readFromFile 9](#_Toc26983816)

[showUserDetails 9](#_Toc26983817)

[bmiCalc 10](#_Toc26983818)

[bmiClassification 11](#_Toc26983819)

[bmiUserCalc 11](#_Toc26983820)

[showUserDetails 12](#_Toc26983821)

[showUserHistoricDetails 12](#_Toc26983822)

[updateUserWeight 13](#_Toc26983823)

[updateUserWeight 14](#_Toc26983824)

[numberValidator 15](#_Toc26983825)

[Number Validation diagram 15](#_Toc26983826)

[stringValidator 16](#_Toc26983827)

[String Validation Diagram 16](#_Toc26983828)

[User 17](#_Toc26983829)

[User UML 17](#_Toc26983830)

[Possible improvements 17](#_Toc26983831)

[Testing 18](#_Toc26983832)

[References 18](#_Toc26983833)

[Testing evidence 19](#_Toc26983834)

[1.1 19](#_Toc26983835)

[1.2 19](#_Toc26983836)

[2.1 19](#_Toc26983837)

[2.2 19](#_Toc26983838)

[3 19](#_Toc26983839)

[4 20](#_Toc26983840)

[5 20](#_Toc26983841)

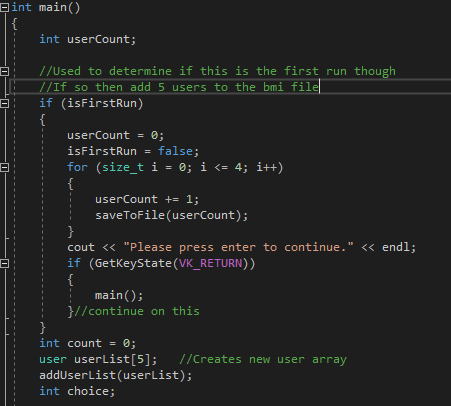
[6 20](#_Toc26983842)

[7 21](#_Toc26983843)

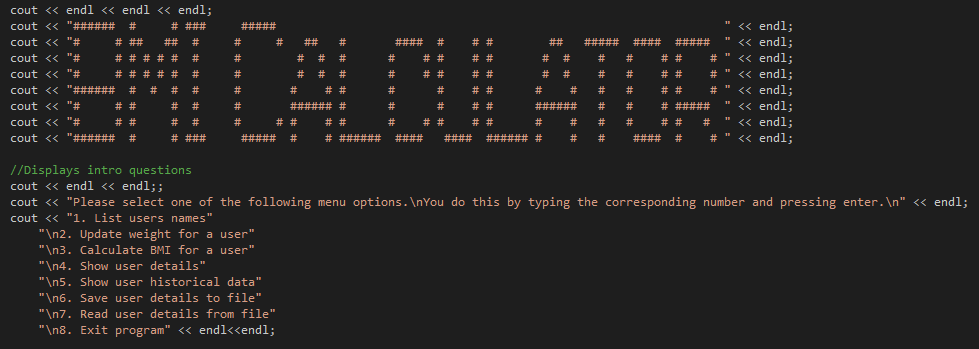
[8 21](#_Toc26983844)

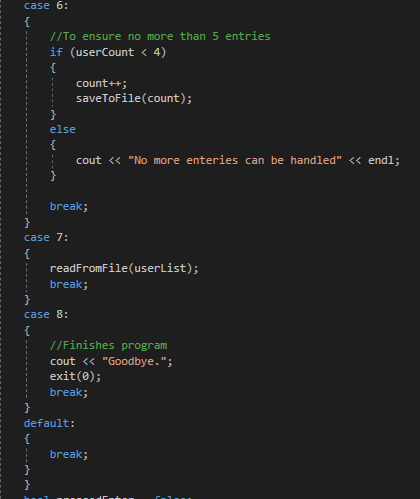
## Main

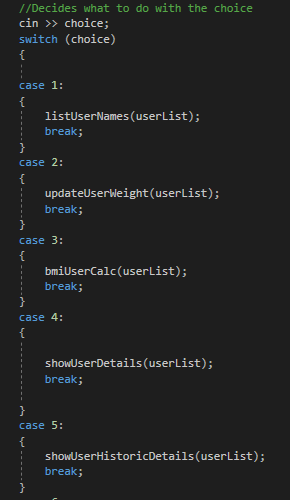
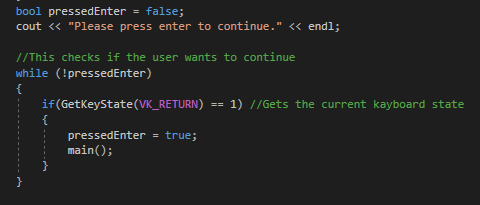
Upon first run through of Main an if statement is used to determine whether the program has ran before. If this is the first iteration through main the user is asked input 5 users before continuing. If the program has been ran before then a userList will be initialised with 5 users in an array. And then the bmisuserdata.txt will be read and all data save to their respective user in userList. This will be done every iteration of main to ensure that all the data held by user is accurate.



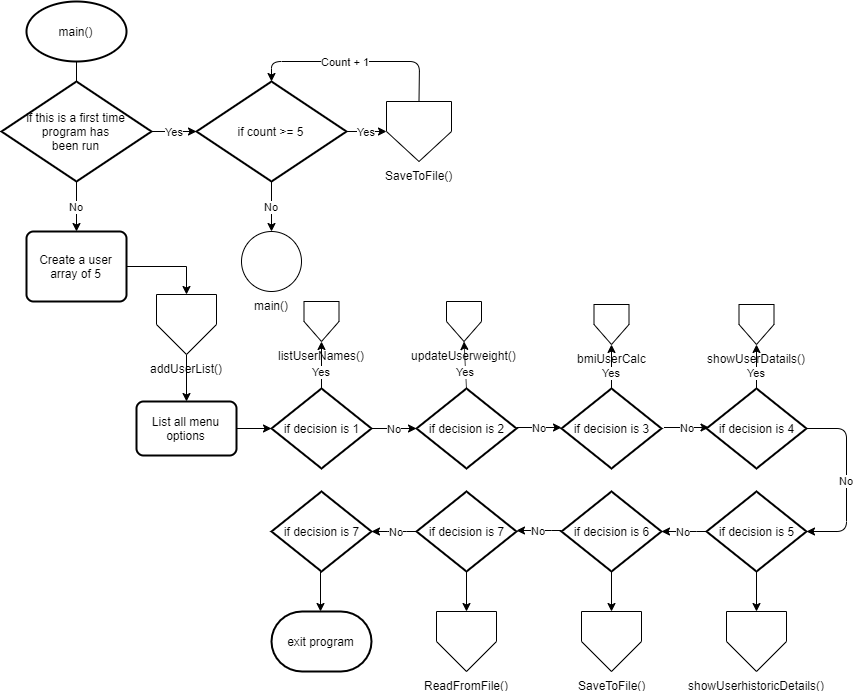
Once all the set up has been completed the main menu system will load, this will present the user with all menu choices that are available.



The user will then be asked to input their menu choice. With their choice a switch case will be used to utilise the correct function that answers the users decision.



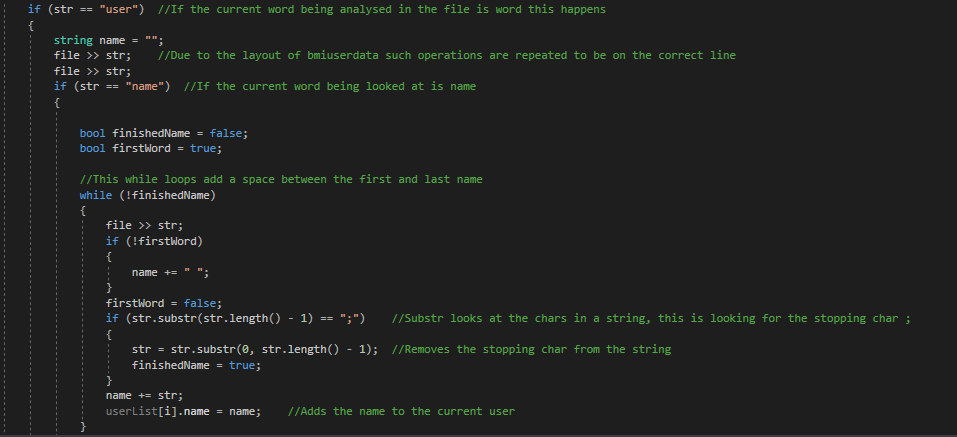
## Main Diagram



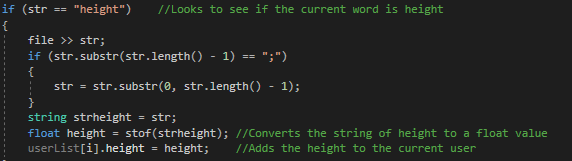
## addUserList

This function is essential for the programs functionality. This reads through the bmiuseredata.txt to gather all relevant information.

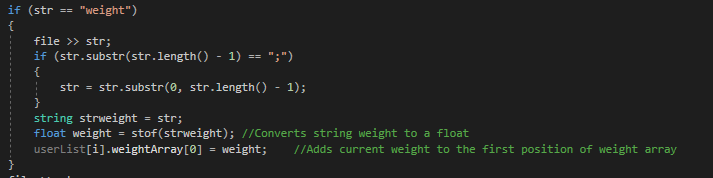
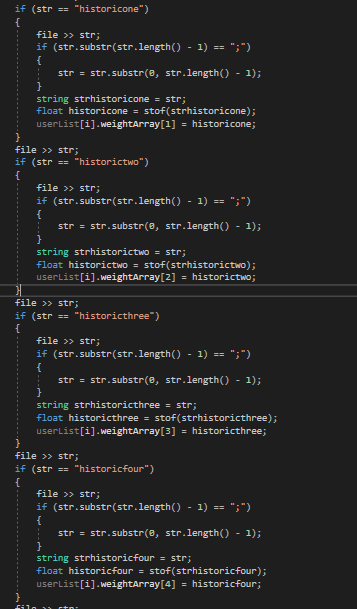
First the programs loops for a user tag, this user tag in the text file is used to determine one user from another user. Once a user tag has been found, the program loops for the name tag, when the name tag has been found the name is added to the relevant user in userList.



The program then looks for a height tag. When found the height is converted from a string to a float, then added to the relevant user.



This process continues for each other tags, weight, historicOne, historicTwo, historicThree, historicFour. With each weight the values found are converted to a float then added to the users weightArray in their correct ordering.

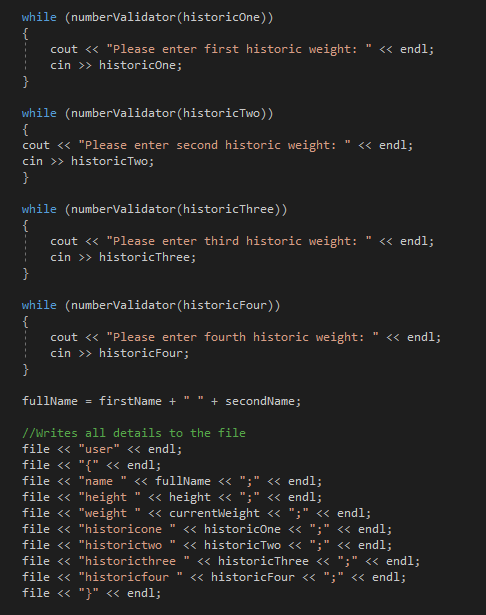


## saveToFile

This function is used to create the file and set up the format in a way so that is used for addUserList can understand and interpret the file.

The user is asked to give the name, height, weight, and the 4 historic weight. These are added to the bmiuserdata file with all delimiters and all essential file layout formats. All values entered are examined to ensure they are the correct data type.

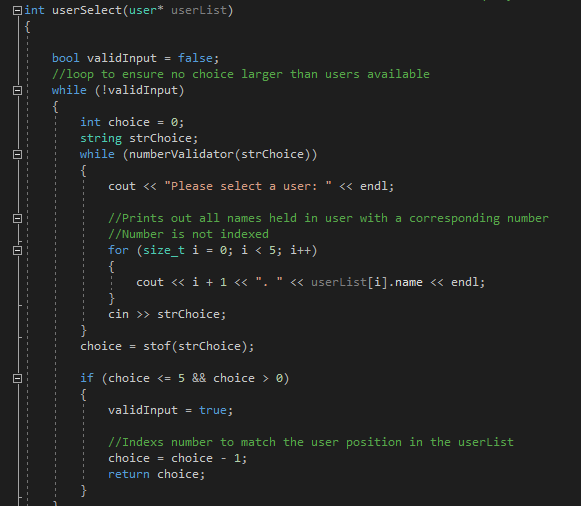




## userSelect

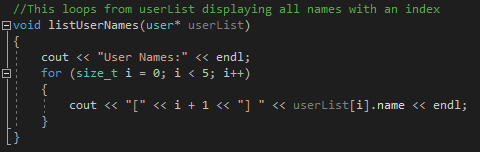
This function is used to ask the user which entity in userList they want information about. Many functions call this function to discover who data is being accessed.

This is done by displaying all user names on to the screen with an index. The user is asked to give a value from that index. If the value is a correct input for the index then that value is returned as an int.



## listUserNames

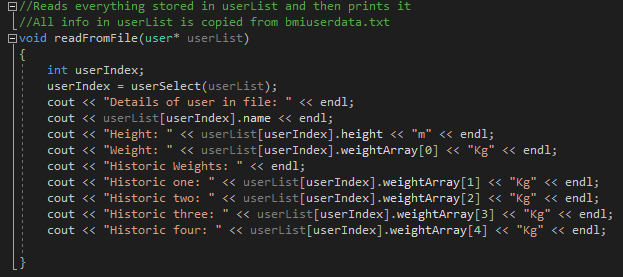
This function is to display all user names onto the console. To do this a for loop is used to cycle through the userList and display each name for the users.



## readFromFile

This file while called read from file is used to cycle though all the users in userList and display all information stored in a user.

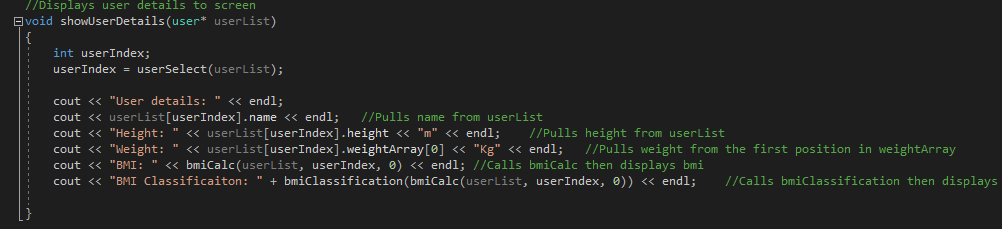
The user will be asked to select a entity from userList. Knowing what entity is being examined all the information for this entity can be displayed on screen.



## showUserDetails

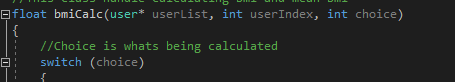
This functions gets the index for the entity from userList, and then displays:

* Name
* Height
* Current weight
* BMI
* BMI classification

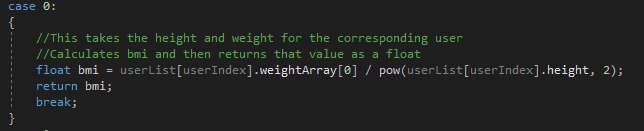


## bmiCalc

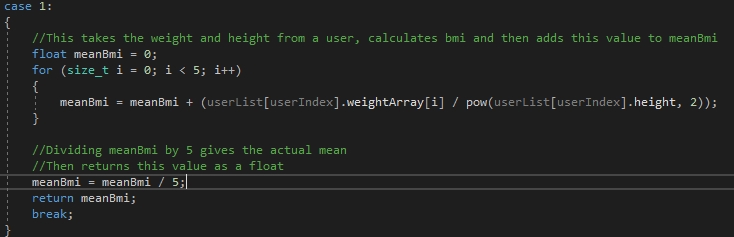
This function uses a switch case statement. The function takes a choice as an argument this is used to decide what operations will be performed.



If this choice is 0 then the chosen users bmi will be calculated, this is done by pulling their height and current weight from userList data and then doing weight / height squared. This value is then returned as a float.

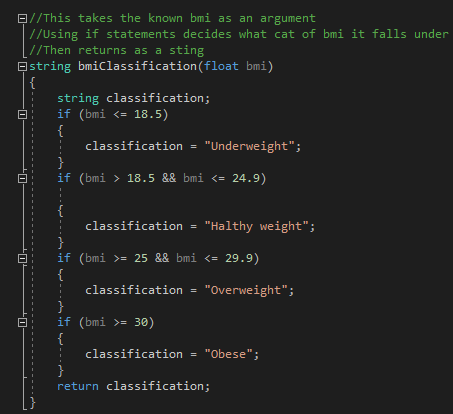


If this choice is 1 then the mean bmi of the user is calculated. This is done by looping 5 times to get each different weight from weight array, and doing the bmi calculations for all the weights. Then diving the final value by 5 to get the mean bmi. This value is then returned as a float.

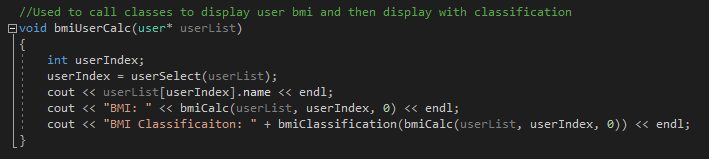


## bmiClassification

This functions takes bmi as a float argument. 4 if statements decide what classification of the bmi is. Then returns that classification as a float.

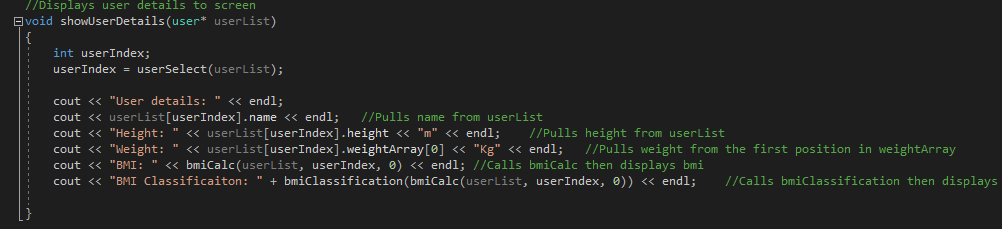


## bmiUserCalc

This function calls to other function for information gathering. Both bmiCalc and bmiClassification are called to get the users bmi and the users bmiClassification. This information is displayed with the name and index of the user. 

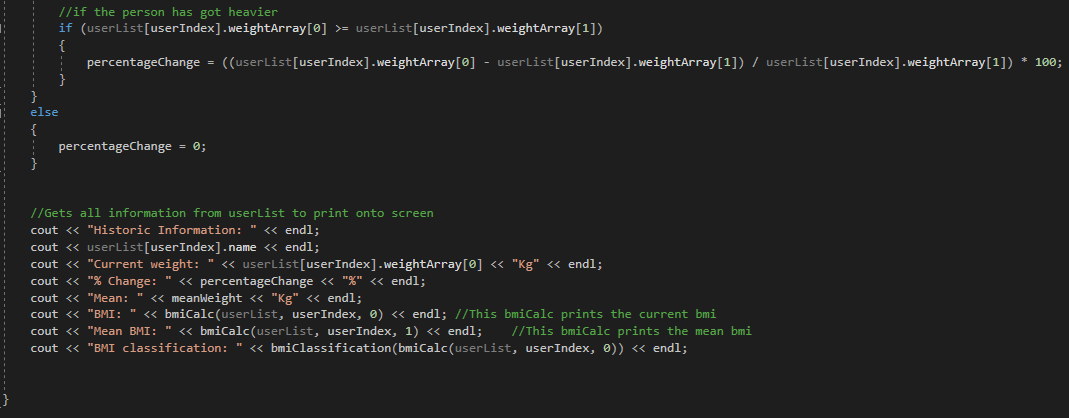
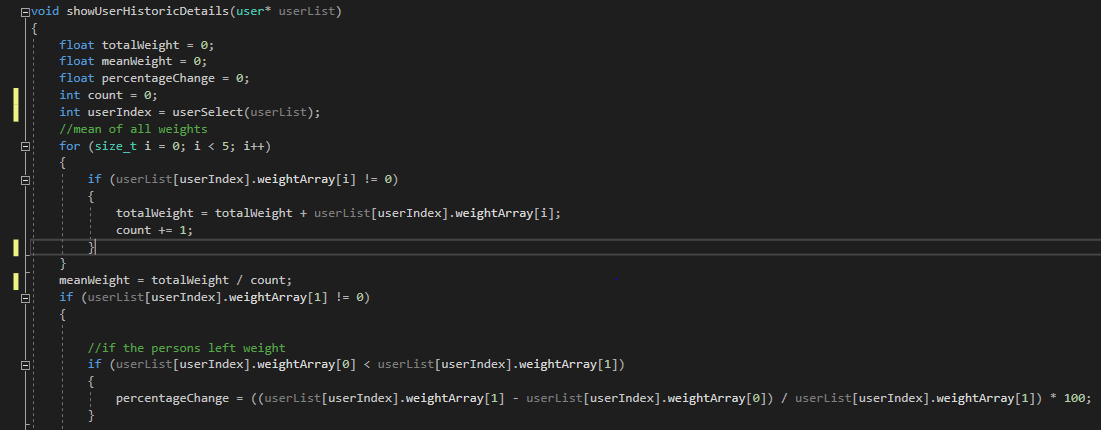
## showUserDetails

showUserDetails is a method whose function is to display information held about a user. The information displayed for a user is: Name, height, current weight, bmi, and the bmi classification. Name, height and current weight are stored in the User class data. While bmi and bmi classification require function calls to calculate that information. userIndex is again used in this function to determine the entity in userList that is being examined.



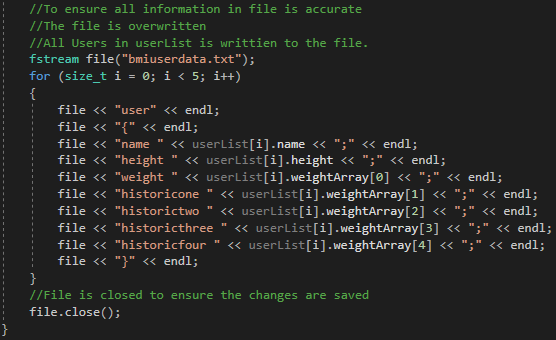
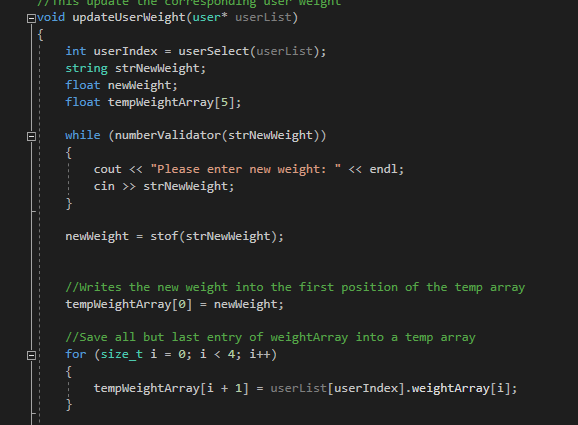
## showUserHistoricDetails

This function is similar to showUserDetails, except this function examine past user data. userIndex is used to determine the entity that is being evaluated. Knowing the user being examined, we can work out the users mean weight, this is done by taking the mean from all 5 weights there is records for. Knowing the users current weight, and the person’s previous weights the function can determine whether or not weight has been gained or lost, then find the percentage for gain or loss.

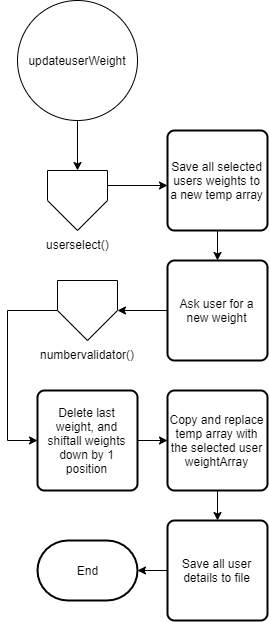
Once all this information has been determined the function will access the user class information and display the name, current weight, change in weight, mean weight, bmi, mean bmi, and bmi classification. 

## updateUserWeight

This function is used to update user records in regards to their stored weight.

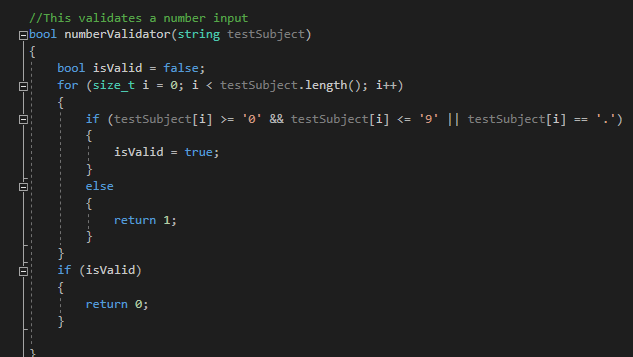
The user will be asked to select an entity using userIndex, and then with that information the user will be asked to update the weight. The value inputed will be validated using numberValidator. If correct the value will be converted to a float. All values in float array will be shift down by 1 position, with the last weight getting deleted. The new weight is added into position 0 of the users weightArray. All information for all the users in userList will then be readded to bmiuserdata.txt to ensure the accuracy of the save data. 

## updateUserWeight

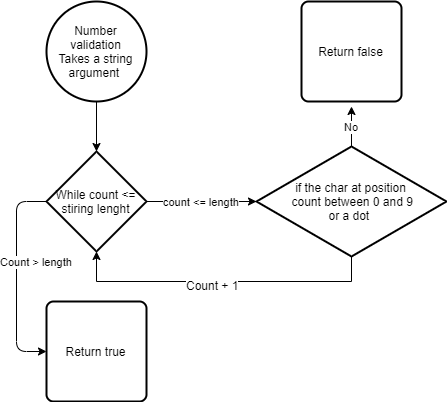


## numberValidator

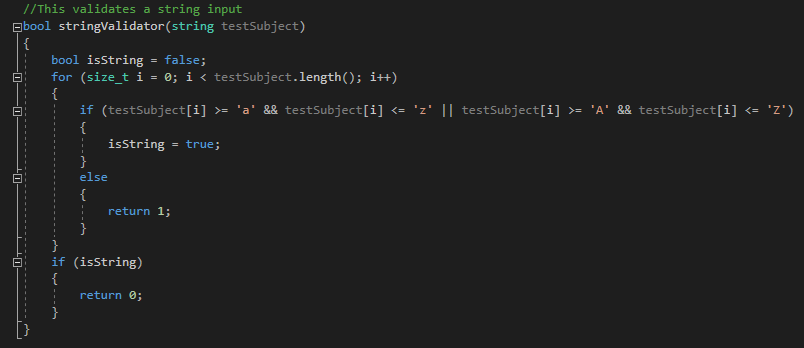
This method takes a string as an argument. A for loop (with iterations dependant on the length of the string input) is used, the string is searched char by char, if the current char is between 0 and 9, or a decimal (.) then this is considered true. Else a return of 0 is given to indicate false.



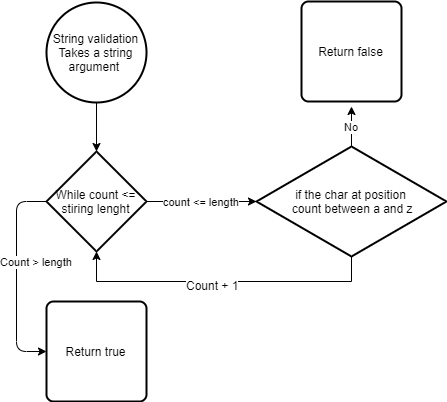
## Number Validation diagram



## stringValidator

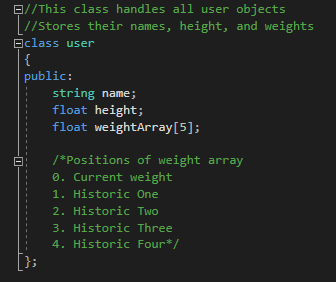
This function behaves similarily to numberValidator, except with the comparisons. This compares each char of the string to ensure that is falls between a and z, and A and Z in the ascii char set. 

## String Validation Diagram

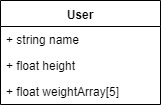


## User

This is the class that handles all user data. The class stores the name, height, and a weight array with a size of 5, these are all public.



## User UML



# Possible improvements

There are a few potential ways in which the program can be improved upon if I was able to repeat the assignment:

* Using vectors over standard arrays. Vectors allow for dynamically sized array meaning there can be as many users and weights as needed, implementing this change would be relatively simple
* Changing the file layout and having all the weights saved on a single this would allow for practically unlimited saved weights to give a true history.

# Testing

|  |  |  |
| --- | --- | --- |
| Condition | Expect results | Evidence |
| Entering a string when expecting a string | Input to be accepted | 1.1 |
| Entering an int when expecting a string | Input to be rejected and question to be repeated | 1.2 |
| Inputting an int when an int is expected | Input is accepted | 2.1 |
| Inputting a string when an int is expected | Input is rejected and the question to be repeated | 2.2 |
| User updating weight | The new weight is added, and each previous weight is shifted down the historic weights | 3 |
| Saving user details to file | All details will appear in bmiuserdata.txt | 4 |
| BMI calculation | Correctly calculated bmi to be shown | 5 |
| Change in weight percentage | Percentage is shown with the correct percentage | 6 |
| List user name | All users names are displayed with a corresponding index | 7 |
| Read from file | Displays all information held in file for a user | 8 |

# References

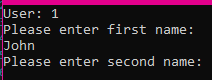
Cplusplus.com. (n.d.). *stof - C++ Reference*. [online] Available at: http://www.cplusplus.com/reference/string/stof/ [Accessed 9 Dec. 2019].

Stack Overflow. (2016). *How do I check if a Key is pressed on C++*. [online] Available at: https://stackoverflow.com/questions/41600981/how-do-i-check-if-a-key-is-pressed-on-c [Accessed 9 Dec. 2019].

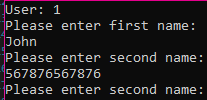
Reddy, A. (2018). *Substring in C++*. [online] Tutorialspoint.com. Available at: https://www.tutorialspoint.com/substring-in-cplusplus [Accessed 9 Dec. 2019].

# Testing evidence

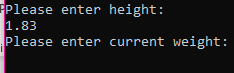
## 1.1



## 1.2



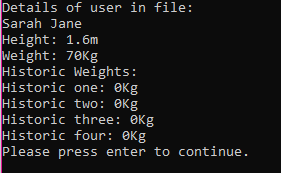
## 2.1

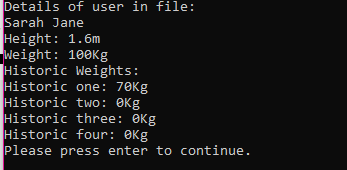


## 2.2

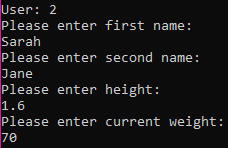


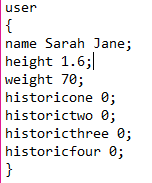
## 3



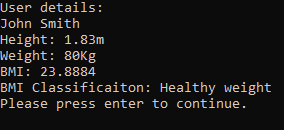


## 4

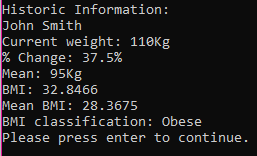




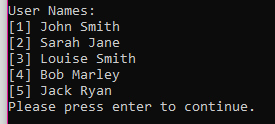
## 5



## 6



## 7



## 8

