CS 202 Spring 2022 - Assignment 1
Review + Intro to Object Oriented Programming

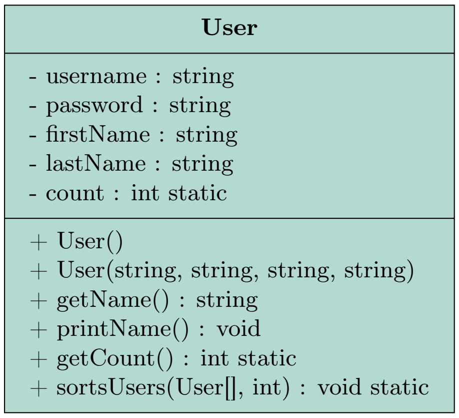
Overview

Welcome to CS 202. This assignment will serve as a brief review on some of the topics from CS 135 and introduce the concept of classes. Classes are similar to structs that you have used before, with the only difference being the default modifier. Classes are private by default, while structs are public; this means that unless we mark class members as public, they will only be accessible by other class members.

Classes and structs are examples of Abstract Data Types (ADTs), meaning they contain variable members and functions; that is, they contain some fields that define them as well as actions associated with objects of that class type. In this assignment, we will implement a simple User class that contains basic information about a User along with functions to set up, print, and access private members.

You will be asked to write a program that reads a list of Users from a file and stores them into an array for later usage. Then, the Users in the array should be sorted by their first names and printed to the screen. To accomplish this, you will be given an input file with each line containing information about one User: their first name, last name, username, and password (See the included file for an example). For each line, we will read this information, use it to make a new User object using a constructor that sets all of these fields, and then store the constructed User within an array.

UML Diagrams



User

- username : string
- password : string
- firstName : string
- lastName : string
- count : int static

+ User()
+ User(string, string, string, string)
+ getName() : string
+ printName() : void
+ getCount() : int static
+ sortsUsers(User[], int) : void static

Important Classes and Functions

Below is a list of functions and variables that are important for this assignment. ***Variables are in green***, **functions that you will need to write are in red**, **functions implemented for you already are in blue**, and **functions that are abstract that will be implemented later are in magenta**.

Global Functions and Constants

- ***const int MAX_ USERS*** - The maximum number of users we can take, being the size of our array of our array of Users in main.
- **User readUserFromFile(ifstream& file)** - This should read information for a single User from the given file and then return the User with that information. Each line in the file will contain one user, given by a first name, last name, username, and password in that order. Each string will be separated by a space, so it is recommended to just use the » operator for reading. After reading each string, construct a new User using the string constructor and return it.
- **int main(int argc, char** argv)** - main is partially done for you, but you will need to finish the code to do what is asked for with the comments in the skeleton. The program should take in a single file name as a command line argument, being a file containing Users to read and work with. An example of a valid usage might be **./a.out database.txt**. Main should check that the program was used correctly and that the number of arguments is correct. Recall that argc is an int telling how many command line arguments there are, including the command used to run the program, while argv is an array of strings containing each argument. In the previous example, argc is 2, argv[0] is "./a.out" and argv[1] is "database.txt". Assuming the program was used correctly, attempt to open the file and check if it was successful using infile.good(). Make sure to print error messages and return if you encounter any errors. Next, if the file was successful, proceed to read each User from the file using the readUserFromFile() function for each individual User until the end of file is reached. Finally, the existing main code will sort and print the array of Users read in. All locals should already be declared for main, but you are welcome to add extra variables if it makes the code easier to understand.

User

A simple class that holds User data and comes packaged with a few helper functions.

- ***string username*** - The User's username for logging in to our site and being displayed on our pages.
- ***string password*** - The password for logging in
- ***string firstName, lastName*** - The first and last name of the User
- ***static int count*** - The count of how many Users have been read. This is static, so it is shared between all Users. This should be incremented in the string constructor and is used by main to keep track of how much of the array has been filled.
- **User()** - Default constructor that initializes the names and password of the User to empty strings
- **User(string f, string l. string u, string p)** - Initializes all of the variable members for this User and then increments the total count of Users by one. This constructor should be used when reading Users from a file
- **static void sortUsers(User userArr[MAX_USERS], int length)** - This static function should take an array of Users and its length and then sort them by their first name. Use Bubble Sort to sort the array. You will want to swap Users that are out of order based on their first name. Because this function is static, it can be called without using an object. Think of it as similar to the functions you implemented in CS 135, but just organized under the class name. By making this a static member rather than global, it will have access to private variables.
- **void printUser()** - This will print the user used to call the function. This is used by main to print the sorted list of Users, but you can call this for debugging purposes.

TO-DO

There are a few sections marked with /* **YOUR CODE HERE***/ where you will need to finish the code. Main needs to finished to error check and open the file given in the command line argument. Assuming the file opens, read from the file using the **readUserFromFile** function for each User, and store the result in the array. After reading through the whole file, the array will be sorted and printed with the existing code at the bottom of main.
Try implementing main first and then gradually work through the necessary functions for the User class used by main.
Submit your updated main.cpp to CodeGrade via the assignment page on Canvas. CodeGrade will automatically try to run your code and provide feedback, but it is suggested to still test and run on a school server first.

Sample Run

Successful Run - database.txt

```
Read a total of 5 Users from file
User count is 5
-----------------------------------------
| Jane Doe           | jaynTho            |
-----------------------------------------


-----------------------------------------
| John Doe           | jTho               |
-----------------------------------------


-----------------------------------------
| Scarlet Jacobs     | Ninja8             |
-----------------------------------------


-----------------------------------------
| Tokio Morishima    | TurtleGuy          |
-----------------------------------------


-----------------------------------------
| Will Wedgewood     | WRed               |
-----------------------------------------
```

Invalid Usage - Incorrect Parameter Count

```
./a.out
ERROR: Incorrect usage.
./a.out <filename>
```

Invalid Usage - File Not Found

```
./a.out bad_file.txt
ERROR: Could not open file: bad_file.txt
```