CE221- C++ Programming

Assignment 2- 22/23 academic year

Author: Michael Barros
Credit: 20% of the total module mark
<u>Deadline: 18th of January of 2023</u>

Introduction

This assignment comprises of a single exercise. Files for use in this assignment are available on Moodle in a zip file. Submission of this assignment will be via FASER.

You should refer to sections 5 and 7 of the Undergraduate Students' Handbook for details of the University policy regarding late submission and plagiarism; the work handed in must be entirely your own. Your programs will be checked by an intelligent plagiarism detection system that looks for similarities between the submitted programs. (Since some of the code has been supplied on Moodle there will inevitably be similarities so I will examine any submissions flagged by the checker to determine whether the similarities that have been detected are reasonable.)

In this assignment, you are asked to write ***a twitter content moderator software*** that:

1) Provides two different sets of users:
    a. the user: those who will read and write tweets.
    b. the managers: those who will set the content not allowed in the platform.
2) Displays the top 10 most dangerous users with blocked content
3) The user statistics based on reputation score

At this time, I will set the output format flexible. However, the user should have the following options:

```
1) Read/write tweet
2) Your statistics
3) Display top 10 most dangerous users
4) Exit game
```

For managers, the following options should appear:

```
1) Reset moderation content
```

```
2) Add blocked content
3) Add another manager
4) Set the sample file
5) Exit game
```

Our mini-project is about how the platform works, and how to update tweets content.

The Classes

Within the folder ex1 in the zip file, you will find files called Person.h , Person.cpp, User.h and Manager.h. These files provide a class to hold details of a person, and declarations for a derived class to hold details of a user and a manager. Such classes would generally be expected to have more data members but only those that are relevant to this assignment have been declared.

The Person class is complete, but you may, if you wish, add extra member functions to the class, e.g. an operator string method; *BUT you should not change any of the existing material, except where noted in this brief.*

The User class and Manager class declaration simply provide declarations for member functions that must be implemented; *you must provide complete implementation for these functions in a file called User.cpp and Manager.cpp, respectively.* The behaviour of the functions should be as specified in the comments supplied in the header file. Again you may, if you wish, add extra member functions to the class (e.g. a method to calculate the average reputation score); if you do so the function bodies should be written in the .cpp file.

There will be also a ReadTweet class discuss further down in the brief.

The Main

In a separate file, you should create your main function.

Ask the username and identify what type of user he is, and direct to their respective options. Managers and users may be identifiable by a mechanism of your choice (here you can modify the existing one). It can be, as an example, name, or a given registration number. For users, if the name does not match to a previous name, then a new user is created, and a message should be output to the screen. Create a template function login that works for both Managers and Users that attempt to open and read from a text file, whose name should be defined in the program, and in that file contains information for both managers and users. If the info of a particular line of that file is for a User, it will contain the registration number and name and scores of a User (accumulative and average scores); these should be passed as arguments to the User constructor to create a new user object, which should be added to a collection of Users. This collection may be a vector, a list or a set (from the standard template library). If the info of a particular line of that file is for a Manager, it will contain a registration number and a manager name. You should create an object of the class Manager that holds who will be the Manager this time. Only an existing manager should add another manager, and the first manager should be you.

For users options:

1) `Read/write tweet –` Function that has two options, read or write a tweet. Reading a tweet is made once at a time. Select a random tweet from the file and then ask the user if they want to read another tweet, report the tweet, or exit. Report the tweet means the tweet will automatically receive a reputation score deducted. Writing a tweet just ask the user to write something in 140 characters and stores in the file.
2) `Your statistics –` The reputation scores of the game should be defined by you and displayed here. However, you will have three scores: one that is per tweet, but also an accumulated score and an average score. These should be presented to the User when they wish to see their statistics. The accumulated score is the sum of all reputation scores a user has obtained. The average score should display the mean reputation score that a user has achieved. These scores should be frequently updated and displayed through a call for a for_each algorithm in the main.
3) `Display top 10 most dangerous users –` you must create an additional function in the main file. The function should print on the screen the lowest top 10 reputation scores. If there is a tie of 0, the number of badly reported tweets will break the tie, considering the *`#moderatedtweet`* flag. How you highlight the score is up to you. This will make sure we know where the user is in the top 10 if it really is in there. However, you should here use the STL sort algorithm to obtain the vector of Users with the highest score to the lowest, and should only display the first 10 elements of the vector. You can use more than one vector. You need to use iterators to go through the vector(s).
4) `Exit –` Just exit the program

For the managers options:

1. `Reset moderation content –` This will reset all the words and phrases that were used to moderate the content
2. `Add blocked content –` Adds words and/or phrases that will block the content
3. `Add another manager –` Set the name and registration number of another manager
4. `Set the sample file –` Set the name of the sample file that contains the tweets
5. `Exit –` Just exit the program

The content moderation engine

The engine must modify tweets of a user so that the blocked content is “blurred out”. We consider content as either a word or a phrase. For this assignment, a “word” should be regarded as being a sequence of non-whitespace characters containing more than one letter. Any punctuation at the beginning or end of a word should be removed (note that there may be more than once such punctuation symbol); punctuation symbols in the middle of a word and any digits are to be regarded as part of the word. You must use the function ispunct, defined in the header file cctype, to check if a character is a punctuation. All input should be converted to lower case, so ”IT”, ”It”, and ”it.” should all be converted to ”it”.

This tweet then consists of showing the regular tweet, but of course, omitting the blocked content. For example, if a word to guess is HATE the tweet should look like

Tweet: "Maria knew why her family did not help her so much, it was because they XXXX her #poorgirl #moderatedtweet"

The content can be more than one word at a time, and can be phrases for example (blocked content is: cried all night)

Tweet: "Ronaldo was so upset that he XXXXX XXX XXXXX #moderatedtweet"

<u>Note that every tweet that has been modified must include the *#moderatedtweet*</u>

The program should read the name of the text file to be used that contains the tweets. There is a sample file provided with assignment brief containing thousands of random tweets and presents the following file structure

userID tweetID tweet date/time

The program should terminate cleanly with an appropriate error message if any of the files cannot be opened using an error exception.

Your program must contain a class called ReadTweets, each with a .cpp and a .h file. The header file for the ReadTweets class and a partial implementation is provided; you must use these and must not make any changes to the header file other than the required changes to the comments. Changes to the provided code in the cpp file are permitted.

You must then provide an implementation of find tweet function that basically selects a random tweet based on a random position of the multimaps.

NOTE: no class for a tweet has been created, but feel free to do that.

Challenge

Make the ReadTweets class as a custom iterable container that operates similarly to a linked list. The top 10 now must combine the sort algorithm and the custom iterator for this class, and you must use iterators to display the top 10 as well.

Deliverables

You should submit a single zip file containing a single folder. The only file formats acceptable are .zip , .7z or a linux gzipped tar file. If you submit a file in any other format, you will lose 10% of the marks for this assignment.

This folder should contain a MAKE file, necessary for the compilation of the application. Also, you need to include all files necessary to make your program executable.

Marking Scheme

Issues that will may lose marks for efficiency could include unnecessary copying of objects, unnecessary repetition of code. Your comments should say precisely what each function does and what each non-local variable represents; within function bodies you should use only brief comments to say what groups of lines do – you must not say what every statement does.

Marking Criteria	**Marks**
Style/Efficiency/Commenting	25
Implementation of all classes (User and Manager)	10
User and manager options	15
Top 10 most dangerous users	5
ReadTweets Class	15
Find tweet with find function	5
Requested usage of STL functions	10
MAKE file for compilation	5
Challenge	10
Total	**100**

I must compile the code using the MAKE file and execute the application for testing.

Needless to say, but: use the STL as much as you like but wisely. You should use as much as you like!