ENG2002

Application Development Assignment Report

Class 2

Group 20

Assignment 4

Task No. 0

Group members:

Ding Tong 17082217D

Wang Wenxing 17083069D

Abstract

In this assignment we built a text-mode user interface application that provides each user an email database. Through this application, users are able to log into his account with password, add new emails to the database, display email information and sort received emails by their senders' names. The information of emails is stored in user specific txt files by the application and will be retrieved when login succeeds, achieving 'memory' of email information and permanent storage of database content.

Introduction

The objective of this assignment is to develop an object-oriented console application that can handle emails. The application will be able to provide an environment to create databases of emails, display the emails in the database, and handle the emails. In addition, each user will need to log in first and enter his or her own email database, which is stored in a ".txt" file in the folder that contains the ".cpp" file.

Concretely, there are two classes, cppEmail and enHandle. While the objects of cppEmail correspond to the emails, the objects of emHandle represent different databases. In cppEmail, it is required to have 6 member variables: To_From in the format of "name [email address] F/T" where "F/T" is used to indicate whether the email is a sender-to-user or a received-from-sender, emSubject that contains the subject, emCC in the format of "name [email address]", emMessage that contains the message, and emDate that is a 6-digit positive number. In emHandle, referring to our team number – 20, we are required to develop a member function to sort the received emails based on the name of the sender in ascending order, and display the sorted list in the format of "name subject date". The member functions of emHandle will also be able to add certain number of emails, list all the emails in the database, read certain user's database from a file and save the current database into a file.

Methodology

Author Contribution

The work division is listed in Table 1.

Table 1. Work division in the team

Ding Tong	**Wang Wenxing**
Design the structure and the flow of the application	Design the structure and the flow of the application
Class emHandle except read and write files	Class cppEmail
Function main()	Function readfile() in emHandle
Function menu()	Function writefile() in emHandle
Function login()	
Testing	Testing
Help each other debug	Help each other debug

** The functions listed above were not developed only by the corresponding authors but developed mostly by the corresponding authors. We helped each other solve problems, debug, and improve the functions. Also, we modified each other's functions to make the development more convenient especially in passing parameters. The workflow and the structures of the application were discussed and designed together.

Schedule and stages

The schedule and stages of the development is listed in Table 2.

Table 2. Schedule and stages of the development

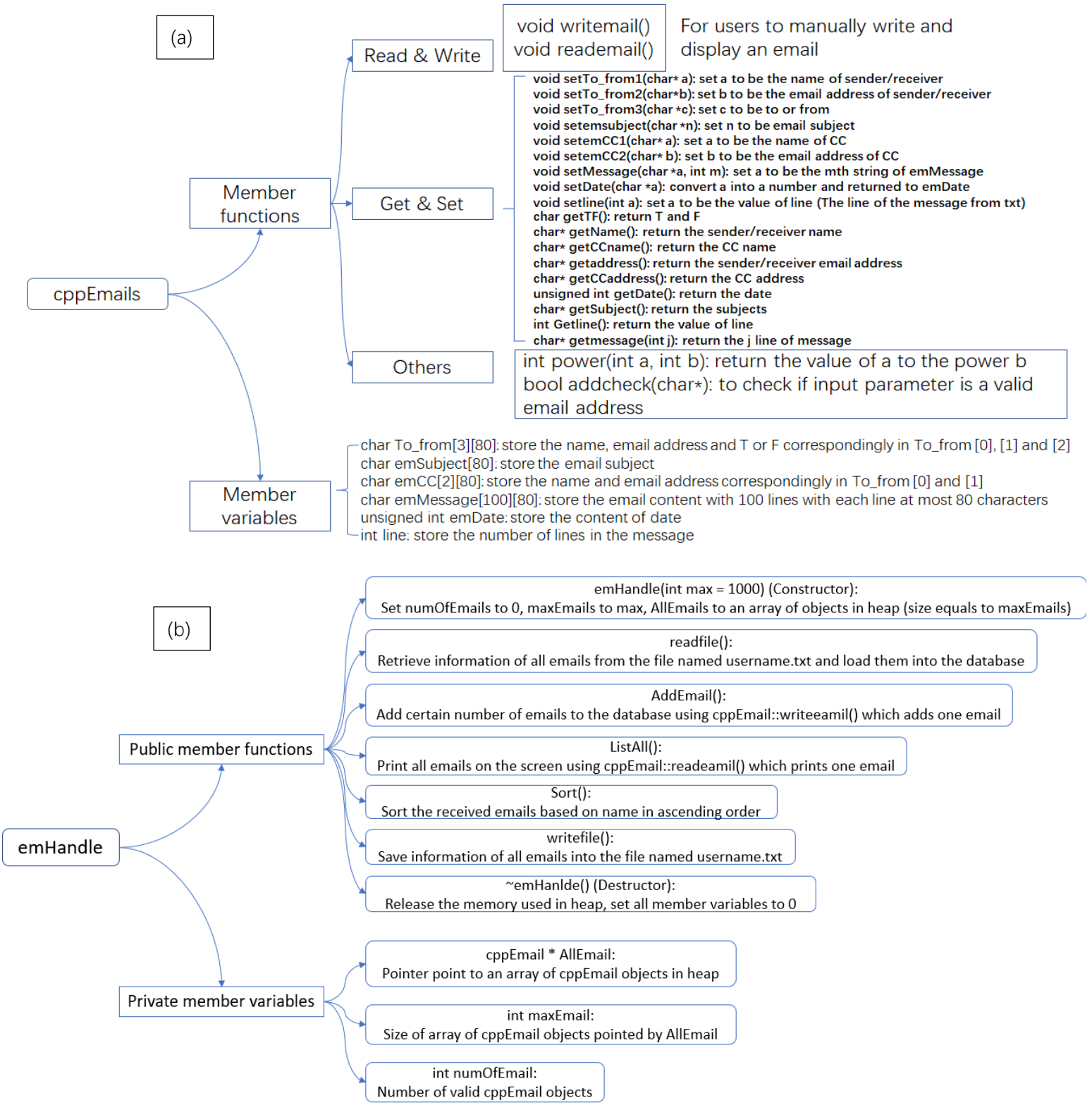
Time	Stage
Week 9	Discuss the structure of cppEmail and emHandle and divide the work
Week 10	Programming for cppEmial, emHandle, and menu respectively
Week 12	Build a console application to test the functions in the two classes Discuss the structure of user login and saving database into a file
Week 13	Redesign the way of passing parameters from cppEmail to emHanlde Modify and add functions to the two classes accordingly Programming for user login and saving files
Week 14	Combine all the functions and test Final debug and modify to improve the functions

**We did not do much things in week 11 because of the special situation happened in

Hong Kong that week.

Structure of the program

The member functions and the variables of class cppEmail and emHandle, together with the flow of the whole application are shown in Fig. 1.



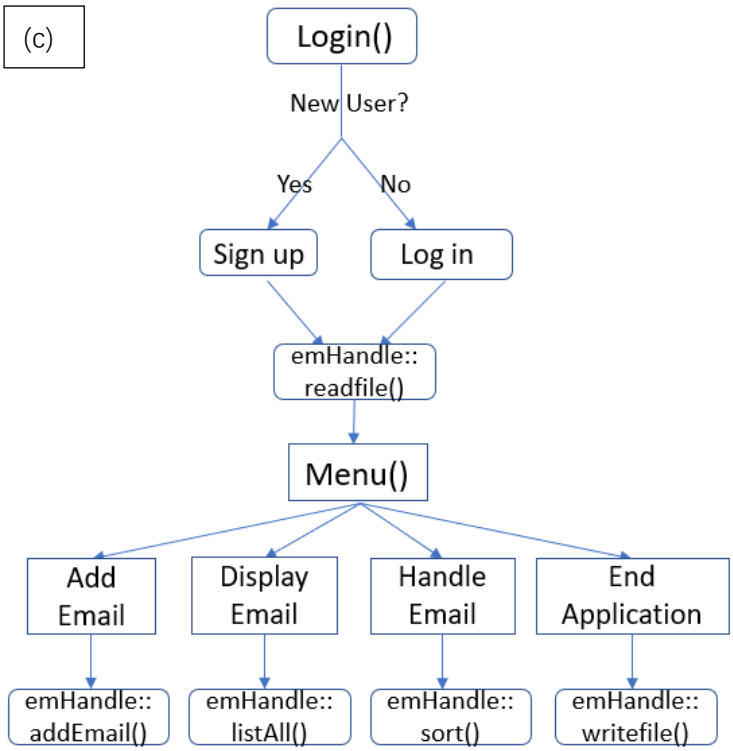


Figure 1. Flow chart of the (a) class cppEmail; (b) class emHandle; (c) the whole application.

The rationale behind the design of the member functions and variables of the two classes in this way is that cppEmail is used to handle each email which is an object of cppEmail, and emHandle mainly handle the database. Therefore, each email is processed first by the member functions of cppEmail, so that all emails in the database can be processed in the same way by using a for loop in member functions of emHandle. As the member variables of cppEmail are private, lots of functions start with get are needed to return those private variables to make it convenient to pass parameters. Apart from the required member variables, we also include line in cppEmail. This member variable is used to help the function reademail and writefile to cout/fout the email content (to specify when the for loop will end for cout/fout). The reason why we have a pointer AllEmail points to an array of cppEmail objects is that we need to pass all the emails which are cppEmail objects to emHandle for processing the whole database. Member variables numOfEmail and maxEmail are included because we need to know the current number of emails in the database to set the upper limit of every for loop for processing the database and we need to set an upper limit for the size of the array in heap.

Problems encountered

At first, we designed the way of passing parameters from cppEmail to emHandle in a bad way that we did not pass the objects of cppEmail but the name, subject, and date separately. To perform sorting, we needed to create array of names, subjects, and dates with size equals to number of emails in the database. In such way, we needed to use second rank pointers to pass parameters, which was complicated and easy to cause run-time errors. Also, the program was not well-structured. As we were not that familiar with the new knowledge and lacked practice at that time, we were struggling in figuring out a better structure but with no good results. We solved the problems after we did the past papers of programming test 2 and the homework, which gave us lots of practice and a clear flow of how to design a database application. Then, we decided to pass objects of cppEmail to emHandle.

The misuse of pointers could also be serious. There were dozens of run-time errors when we tested the program. About half of them are related to misuse of pointers. Error information had almost no value of reference when dealing with this kind of problem so we had to add many breakpoints or cout key parameters at different places to see at which place the program went wrong.

Things got rather difficult when developing the member function writefile and readfile. We encountered the problem of passing two-dimensional array again. Since the member variable of emMessage is a two-dimensional character array containing at most 100 strings, it is not easy to read the information in the txt file and pass the parameters to the corresponding objects. It is also not easy to judge how many lines the message is since it is determined by users. We tackled with the following ways.

1. We let the function setMessage have 2 input parameters, with one being a string and another being an integer to specify which line the user wants for the input string. The function actually passes only one string at a time to the two-dimensional array emMessage and a for loop is used to copy all the messages into the emMessage.
2. During the output of results into the txt file, we manually added a line of ‘end of message’. When reading the file, if this line is detected, the for loop will end and the writing of the emMessage will stop.

Some run-time errors due to the whole structure of program also came into being when we completed the readfile and writefile functions. The structure got quite complicated when developing these two functions since they were added lastly. As a result, any small change made for these 2 functions would affect the whole program, leading to run-time error. We had to patiently debug and most of the completed codes were changed to let the program run smoothly.

Testing of the program

We tested the program by simply running it under a new project 'Test'. Tests were carried out when the following works were done.

1. Finishing writing the cppEmail library & emHandle library
2. Finishing writing the user interface
3. Finishing implementing the emHandle library with functions to write and read files.

Testing cppEmail & emHandle

We tested the library by using another project called test. We tested all the member functions by simply referring to them and cout the results to see if they matched our expectations. We also tested some kinds of special requirements. For example, when checking if there is only one '@' in the middle of email address, we typed @a, a@, a@@a. aa for testing.

Testing user interface

We tested the user interface in the following aspects.

1. We created a user account and open the user.txt file to see if the format is exactly what we expected.
2. We created another account with the same name to see if the program can detect whether the account has existed or not.
3. We then tested the password system by
    a) Entering 3 times and all of them are wrong to see if the program will end
    b) Entering 3 times with the first two of them being wrong and the last one being right to see if we can successfully log in.
4. We used option a to add 1, 2 and 3 emails correspondingly and the used

option b and c to see if the output really matches what we expected.

5. We used option q to see if the program can end successfully.

Testing read and write file functions

1. We wrote emails in the database as usual.
2. We ended the program and started it again to see if the emails we wrote during the last program running was read from the file and could be displayed by using option b and c.
3. We then opened the file created and saw if the file content was exactly what we wanted.

Result

Assume we had three existing user accounts whose name is blue shown in figure 1.

Figure 2. Existing accounts. 1A: Files storing the account database contents. 1B. File storing the user information with the formula “user password”.

The program will end if one wants to create an existing account.

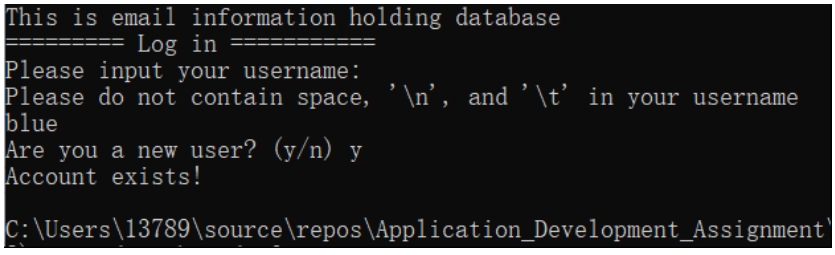


Figure 3. A user who is rejected when he is creating existing account.

We then successfully created a new account and wrote 2 emails.

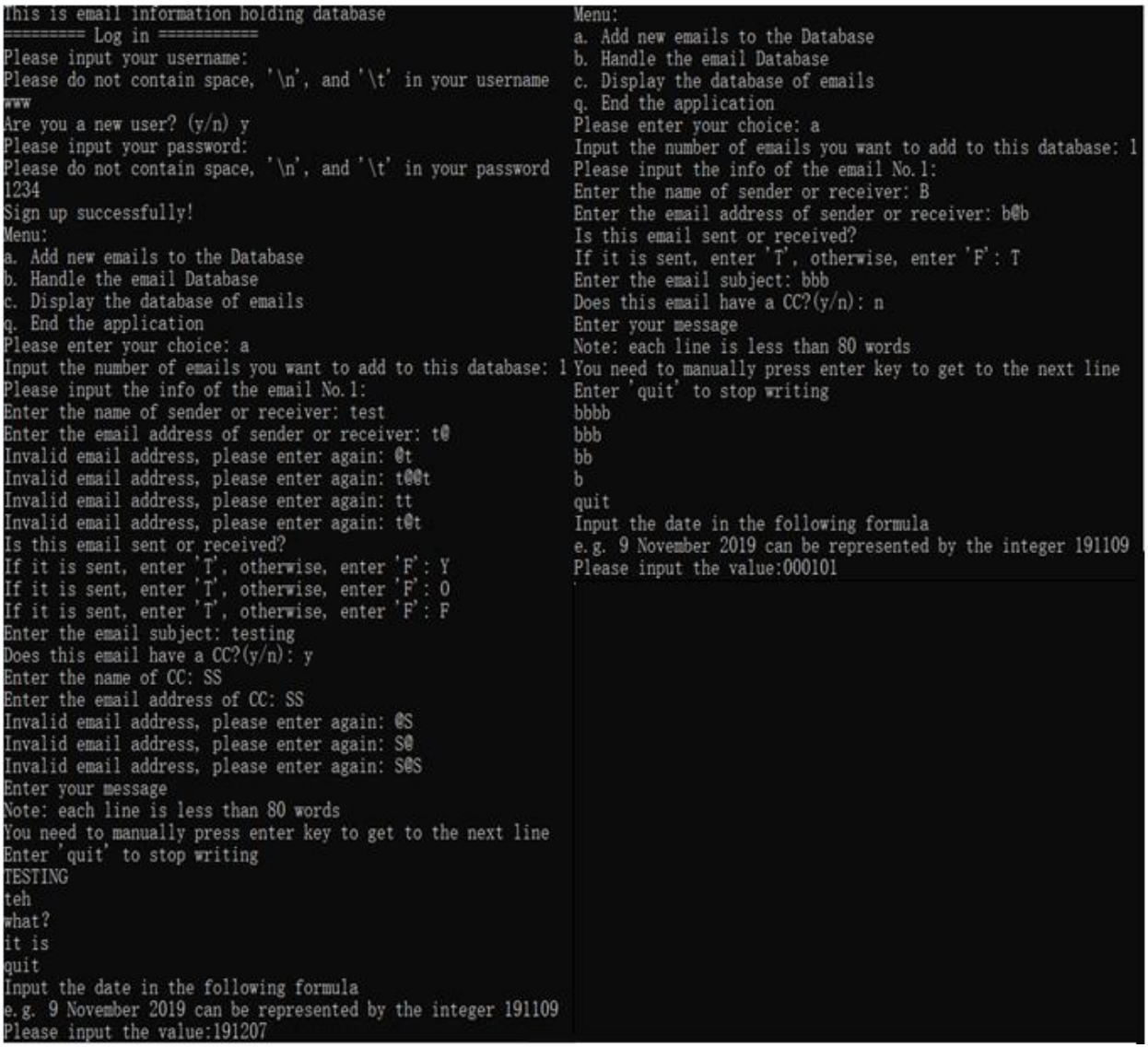


Figure 4. Creating new account and wrote 2 emails by user.

The content is stored in the database and written in the txt file named by the user account.

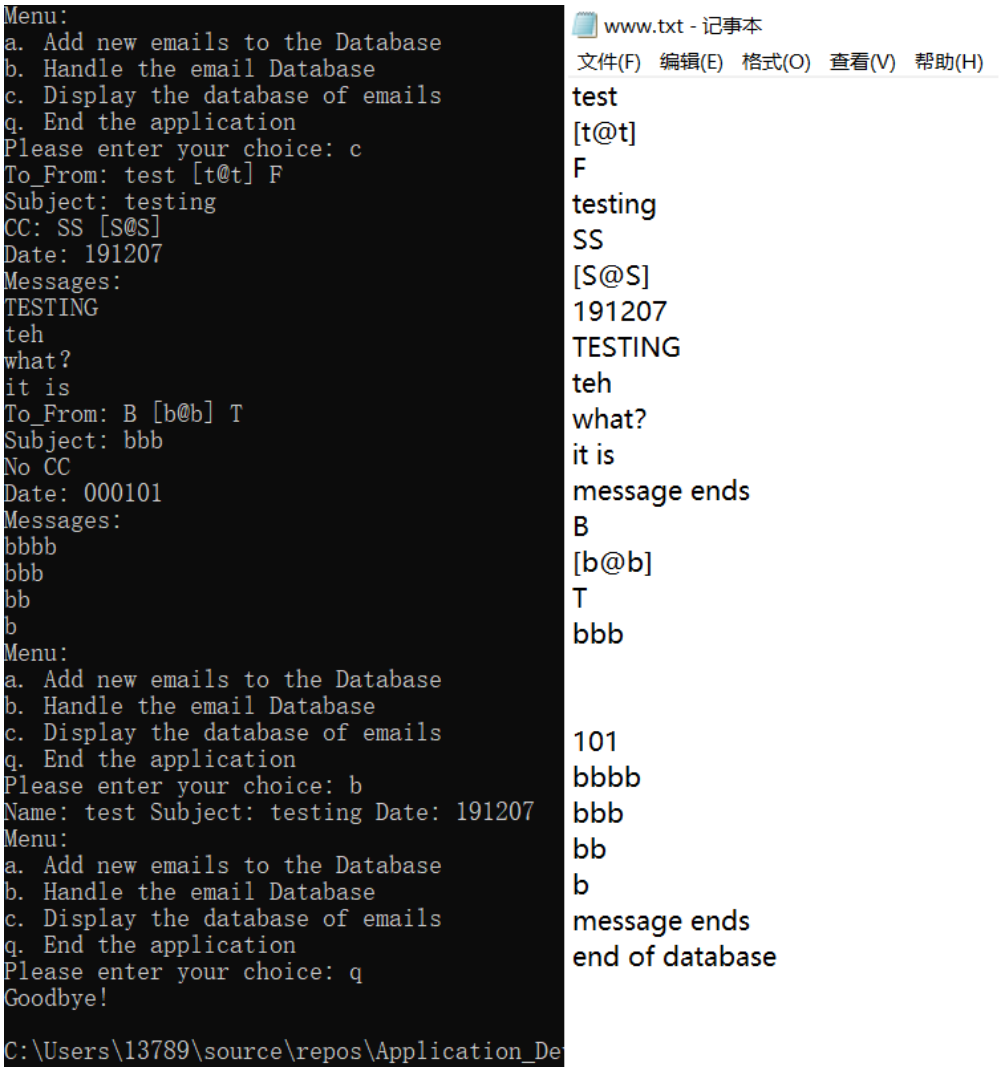


Figure 4. The display of email contents and file storing result.

The program will end if user enter 3 times of wrong password

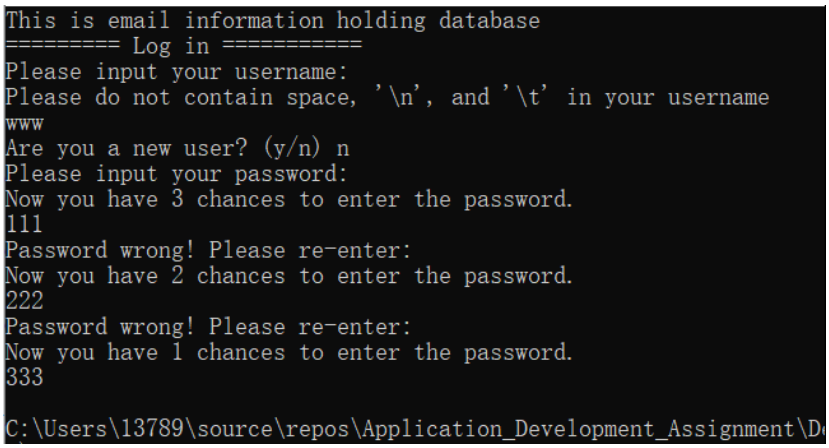


Figure 5. A user who is rejected when he is enters 3 times of wrong password.

When successfully logging in, the database will be automatically loaded and the newly created emails will also be stored in the file after program ends.

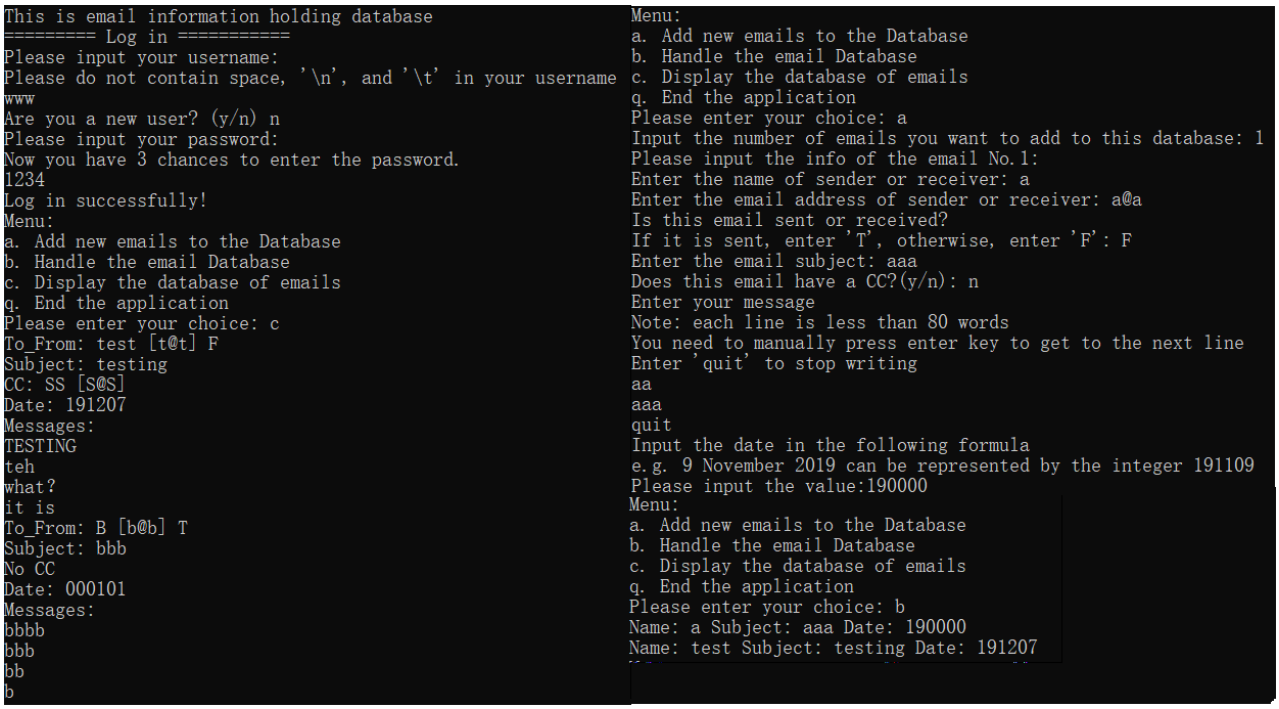
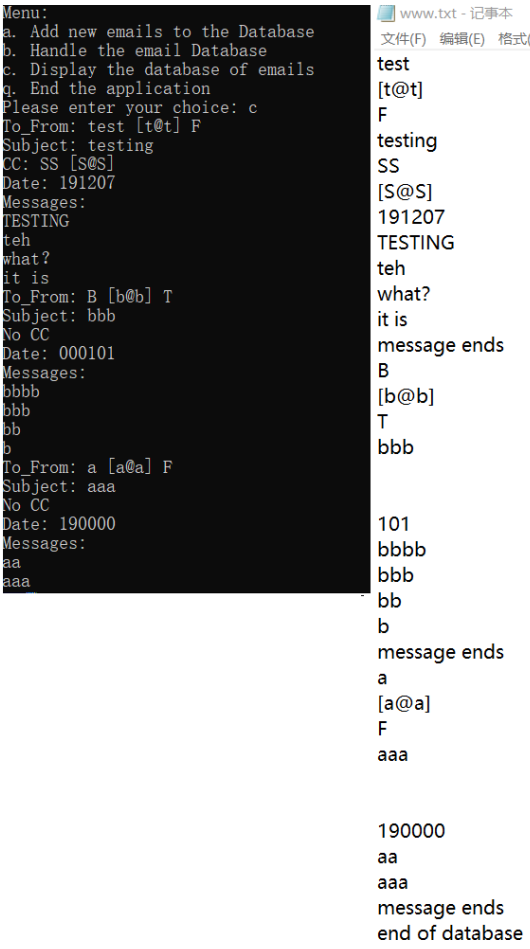


Figure 6. The loading of database after login. The newly created email will also be added to the database together with old ones in the txt file.



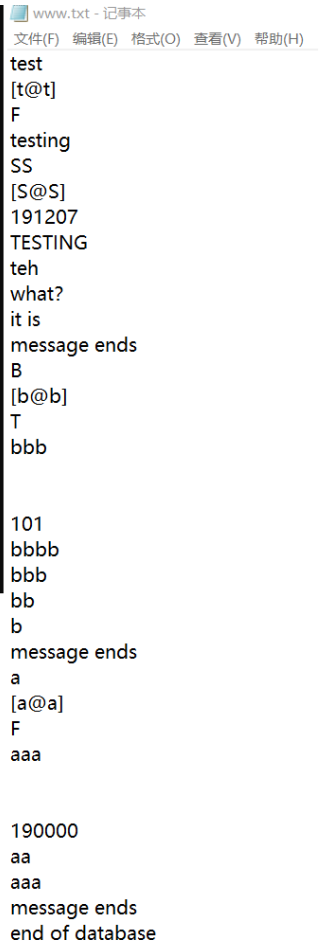


Figure 7. The newly added information is stored in the database as well as the txt file.

Conclusion & further development

In this assignment, we improved our understanding on lots of concepts such as pointers and multidimensional array significantly, learnt a lot on how to design a good program that caters the users and makes other programmers easy to read, and how to communicate and work efficiently with teammate.

If more time is allowed, on the one hand, we would like to add some more functions of handling the database to extend our program for users. In the developed program, users can only add emails to the database but they cannot delete. This is important when users add wrong information because of typo or something else. We should also add a function to make users able to search for certain emails, especially when the number of emails stored is large.

On the other hand, we also want to further improve our current structures to make the program use less memories so that it can always run fast even when the number of emails stored becomes large. For example, every time when a user log in and want to handle the emails, we first read all emails from the file into the heap, which will occupy lots of resources when there are a lot of emails.

All in all, we developed an email database console application that achieves all the requirements and is easy to use. Although we paid lots of efforts to improve it, because of the time limitation, there are lots of room for further extension and improvement.