

PSP0015: PROBLEM SOLVING IN PROGRAMMING AND SYSTEM DESIGN

PROJECT DOCUMENTATION

\*\*\*\*\*

LECTURER: SUHAINI BINTI NORDIN

GROUP: FT41

SUBMISSION DATE: 24-01-2014

|  |  |  |
| --- | --- | --- |
| NAME | ID | SIGNATURE |
| Lee Zhen Zhi | 1122702996 |  |
| Ooi Choon Ho | 1122702995 |  |

# TABLE OF CONTENTS

Contents

[TABLE OF CONTENTS 1](#_Toc378253663)

[INTRODUCTION 2](#_Toc378253664)

[INPUT PROCESS OUTPUT CHART 3](#_Toc378253665)

[PROBLEM ANALYSIS CHART 4](#_Toc378253666)

[DATA DICTIONARY 5](#_Toc378253667)

[PSEUDOCODE AND C++ CODE 8](#_Toc378253668)

[FLOWCHART 39](#_Toc378253669)

[CONCLUSION & COMMENTS 57](#_Toc378253670)

# INTRODUCTION

This is a project conducted under the subject PSP0015 – Problem Solving in Programming and System Design on Trimester 2 2013/2014.

In this project, we are required to build a simple program and a matching documentation for the program.

The program that we built is called “MyCheckList”. It is a program that allows multiple users to use it to store their daily tasks (like a daily to-do list).

The feature of this program is as stated before, it allows multiple users to use it to store their data and the program makes sure that there is no conflict of data between multiple users.

We were able to get the program running well without any error through countless trial and errors and alpha testing.

This program involves the use of <fstream>, the library for file manipulation.

The program allows the user to:

1. Create a user account
2. Delete a user account
3. Create a task
4. Delete a task
5. Check a task

# INPUT PROCESS OUTPUT CHART

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Process Output Chart** | | | |
|
| **Input** | **Process** | **Module** | **Output** |
| 1. user\_choice (before login) | GET user\_choice | main |  |
|  | CASE user\_choice == '1' | main |  |
| 2. username (create user) | GET username | userCreator |  |
| 3. password (create user) | GET password | userCreator |  |
|  | Create user | userCreator | User is created |
|  | CASE user\_choice == '2' | main |  |
| 4. input\_id (delete user) | GET input\_id | userRemover |  |
| 5. password1 | GET password1 | userRemover |  |
|  | Delete user | userRemover | User is deleted |
|  | CASE user\_choice == '3' | main |  |
| 6. username (login user) | GET username | userAuthenticator |  |
| 7. password (login user) | GET password | userAuthenticator |  |
|  | Authenticate user | userAuthenticator | Login user and go to user menu |
|  | CASE user\_choce == '4' | main |  |
|  | Exit program | main |  |
|  |  |  |  |
| 8. user\_choice (after login) | GET user\_choice | main |  |
|  | CASE user\_choice == '0' | main |  |
|  | Logout user | main | Logout user and go to main menu |
|  | CASE user\_choice == '1' | main |  |
| 9. task | GET task | taskCreator |  |
|  | Create task | taskCreator | Task is created |
|  | CASE user\_choice == '2' | main |  |
| 10. input\_id (delete task) | GET input\_id | taskRemover |  |
|  | Delete task | taskRemover | Task is deleted |
|  | CASE user\_choice == '3' | main |  |
| 11. input\_id (tick task) | GET input\_id | taskMarker |  |
|  | Tick/untick task | taskMarker | Task is ticked/unticked |
|  | CASE user\_choice == '4' | main |  |
|  | Exit program | main |  |
|  | END | main |  |

# PROBLEM ANALYSIS CHART

|  |  |
| --- | --- |
| **Problem Analysis Chart** | |
|
| **Given data** | **Required results** |
| 1. user\_choice (before login) | 1. User is created |
| 2. username (create user) | 2. User is deleted |
| 3. password (create user) | 3. Login user and go to user menu |
| 4. input\_id (delete user) | 4. Logout user and go to main menu |
| 5. password1 | 5. Task is created |
| 6. username (login user) | 6. Task is deleted |
| 7. password (login user) | 7. Task is ticked/unticked |
| 8. user\_choice (after login) |  |
| 9. task |  |
| 10. input\_id (delete task) |  |
| 11. input\_id (tick task) |  |
|  |  |
| **Processing Required** | **Solution Alternatives** |
| 1. IF authenticate return FALSE | 1. Use XML to store data |
| Keep looping the program | 2. Use config file to store data |
| CASE |  |
| user\_choice = '1' |  |
| Process userCreator |  |
| user\_choice = '2' |  |
| Process userRemover |  |
| user\_choice = '3' |  |
| Process userAuthenticator |  |
| user\_choice = '4' |  |
| Thank the user |  |
| otherwise |  |
| Program tells user that the input is invalid |  |
|  |  |
| 2. If authenticate return TRUE |  |
| Keep looping the program |  |
| CASE |  |
| user\_choice = '0' |  |
| authenticate = false |  |
| user\_choice = '1' |  |
| Process taskCreator |  |
| user\_choice = '2' |  |
| Process taskRemover |  |
| user\_choice = '3' |  |
| Process taskMarker |  |
| user\_choice = '4' |  |
| Thank the user |  |
| otherwise |  |
| Program tells user that the input is invalid |  |

# DATA DICTIONARY

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Data Dictionary** | | | | | | |
|
| **Item** | **Variable name** | **Data type** | **Module** | **Scope** | **Pseudonym** | **Error check** |
| Username | username | string | main | Global | username (userCreator, userAuthenticator, taskCreator, taskRemover, taskMarker, createUser, loginUser, verifyTask, createTask, deleteTask, tickTask), username1 (verifyUser) | None |
| Password | password | string | main | Global | password (userCreator, userAuthenticator, createUser, loginUser) | None |
| Test for authentication | authenticate | bool | main | Global | authenticate (userAuthenticator) | None |
| User's choice before login | user\_choice | char | main | Local | user\_choice (userRemover) | Only accept '1', '2', '3' or '4' |
| User's choice after login | user\_choice | char | main | Local | user\_choice (taskRemover, taskMarker) | Only accept '0', '1', '2', '3', or '4' |
| List of users | user\_list | vector<string> | main | Local | user\_list (userCreator, userRemover, listUsers, deleteUser) | None |
| User's checklist | user\_checklist | string | main | Local | file\_name (createList) | None |
| List of tasks | task\_list | vector<string> | main | Local | task\_list (taskCreator, taskRemover, taskMarker, listTasks, deleteTask, tickTask) | None |
| List of text | text\_list | string | createList | Local | None | None |
| A string | line | string | createList | Local | None | None |
| Boolean test | test\_bool | boolean | verifyUser | Local | None | None |
| A string | line | string | verifyUser | Local | None | None |
| 2nd username | username2 | string | verifyUser | Local | None | None |
| User's information | user\_info | string | createUser | Local | None | None |
| User's checklist | user\_checklist | string | createUser | Local | None | None |
| User's ID | user\_id | integer | listUsers | Local | None | None |
| User's information | user\_info | string | listUsers | Local | None | None |
| Long username | long\_username | string | listUsers | Local | None | None |
| Split username | split\_username | vector<string> | listUsers | Local | None | None |
| New list | new\_list | vector<string> | deleteUsers | Local | None | None |
| User's information | user\_info | string | deleteUsers | Local | None | None |
| Username | username | string | deleteUsers | Local | None | None |
| User's checklist | user\_checklist | string | deleteUsers | Local | None | None |
| Boolean test | test\_bool | boolean | loginUser | Local | None | None |
| User's information | user\_info | string | loginUser | Local | None | None |
| A string | line | string | loginUser | Local | None | None |
| Boolean test | test\_bool | boolean | verifyTask | Local | None | None |
| User's checklist | user\_checklist | string | verifyTask | Local | None | None |
| A string | line | string | verifyTask | Local | None | None |
| User's Checklist | user\_checklist | string | createTask | Local | None | None |
| Task's ID | task\_id | integer | listTasks | Local | None | None |
| Split task | split\_task | vector<string> | listTasks | Local | None | None |
| New list | new\_list | vector<string> | deleteTask | Local | None | None |
| User's checklist | user\_checklist | string | deleteTask | Local | None | None |
| Boolean test | test\_bool | boolean | tickTask | Local | None | None |
| New list | new\_list | vector<string> | tickTask | Local | None | None |
| Task | task | string | tickTask | Local | None | None |
| Task | task | string | tickTask | Local | None | None |
| User's checklist | user\_checklist | string | tickTask | Local | None | None |
| Input ID | input\_id | string | userRemover | Local | None | None |
| 1st password | password1 | string | userRemover | Local | None | None |
| Converted ID | conv\_id | integer | userRemover | Local | None | None |
| User's ID | user\_id | integer | userRemover | Local | user\_id (deleteUser) | None |
| User's information | user\_info | string | userRemover | Local | None | None |
| 2nd password | password2 | string | userRemover | Local | None | None |
| Task | task | string | taskCreator | Local | task (verifyTask, createTask) | None |
| Input ID | input\_id | string | taskRemover | Local | None | None |
| Converted ID | conv\_id | integer | taskRemover | Local | None | None |
| Task ID | task\_id | integer | taskRemover | Local | task\_id (deleteTask) | None |
| Input ID | input\_id | string | taskMarker | Local | None | None |
| Converted ID | conv\_id | integer | taskMarker | Local | None | None |
| Task ID | task\_id | integer | taskMarker | Local | task\_id (tickTask) | None |
| New string | new\_s | string | centerStr | Local | None | None |
| Midpoint of string | s\_mid\_point | integer | centerStr | Local | None | None |
| New string | new\_s | string | bSpaceStr | Local | None | None |
| New string | new\_s | string | splitStrPos | Local | None | None |
| Token | token | string | splitStrPos | Local | None | None |
| New string | new\_s | string | eraseStr | Local | None | None |

# PSEUDOCODE AND C++ CODE

**main.cpp**

|  |
| --- |
| **Pseudocode** |
| Include library and header files |
|
|
|
|
|
|
|
|
|  |
| Declare variable username of type string |
| Declare variable password of type string |
| Declare variable authenticate of type boolean = FALSE |
|  |
| START: |
| Create directory “config” |
| Change working directory to “config” |
| DO |
|  |
|
|
|
|
| WHILE authenticate EQUALS FALSE |
| Declare variable user\_choice of type character |
| Declare variable user\_list of type vector string = CALL createList(parameter: "users.txt") |
| Clear screen |
| DISPLAY Welcome message and instructions |
|
|
|
|
|
|
|
|
|
|
| IF user\_choice NOT EQUALS '2' |
| Ask the user to enter user\_choice |
| Clear any state on cin |
| Sync cin stream |
| GET user\_choice |
| END IF |
| SWITCH user\_choice |
| CASE '1' |
| CALL userCreator(parameters:username,password,user\_list) |
| BREAK |
| CASE '2' |
| CALL userRemover(parameters:user\_choice,user\_list) |
| BREAK |
| CASE '3' |
| CALL userAuthenticator(parameters: username,password,authenticate) |
| BREAK |
| CASE '4' |
| CALL exitProgram() |
| Exit program |
| DEFAULT |
| CALL invalidInput(parameter: "Please choose between 1, 2, 3 or 4!") |
| END SWITCH |
| END WHILE |
|  |
|
|
|
|
| WHILE authenticate EQUALS TRUE |
| Declare variable user\_choice of type character |
| Declare variable user\_checklist of type string = username + “.txt” |
| Declare variable task\_list of type vector string = CALL createList(parameter: user\_checklist) |
| Clear screen |
| DISPLAY Welcome message and instructions |
|
|
|
|
|
|
|
|
|
|
|
|
| IF size of task\_list MORE THAN 0 |
| CALL listTasks(parameters:task\_list,”Your tasks:”) |
| ELSE |
| DISPLAY “You don't have any task(s).” |
|
|
|
| END IF |
| IF user\_choice NOT EQUALS '2' AND user\_choice NOT EQUALS '3' |
| Ask the user to enter user\_choice |
| Clear any state on cin |
| Sync cin stream |
| GET user\_choice |
| END IF |
| SWITCH user\_choice |
| CASE '0' |
| authenticate = false |
| BREAK |
| CASE '1' |
| CALL taskCreator(parameters:username,task\_list) |
| BREAK |
| CASE '2' |
| CALL taskRemover(parameters:username,user\_choice,task\_list) |
| BREAK |
| CASE '3' |
| CALL taskMarker(parameters:username,user\_choice,task\_list) |
| BREAK |
| CASE '4' |
| CALL exitProgram() |
| Exit program |
| DEFAULT |
| CALL invalidInput(parameter: "Please choose between 0, 1, 2, 3 or 4!") |
| END SWITCH |
| END WHILE |
| WHILE 1 EQUALS 1 |
| END |
|

|  |
| --- |
| **C++** |
| #include <iostream> |
| #include <fstream> |
| #include <sstream> |
| #include <cstdlib> |
| #include <direct.h> |
| #include <vector> |
| using namespace std; |
| #include "zString.hpp" |
| #include "main.hpp" |
|  |
| string username; |
| string password; |
| bool authenticate = false; |
|  |
| int main() { |
| \_mkdir("config"); |
| \_chdir("config"); |
| do { |
| // |
| // |
| // Start |
| // |
| // |
| while (authenticate == false) { // MAIN MENU |
| char user\_choice; |
| vector<string> user\_list (createList("users.txt")); |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Welcome to MyCheckList!",78)<<"|\ |
| |"<< centerStr("What would you like to do?",78)<<"|\ |
| |"<< centerStr("1 - Create a new user",78)<<"|\ |
| |"<< centerStr("2 - Delete a user ",78)<<"|\ |
| |"<< centerStr("3 - Login ",78)<<"|\ |
| |"<< centerStr("4 - Quit ",78)<<"|\ |
| |"<< centerStr("(Enter the number before an operation to start)",78)<<"|\ |
| |"<< centerStr("(For example: key in 1 if you want to create a new user)",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| if (user\_choice != '2') { |
| cout << ">> "; |
| cin.clear(); |
| cin.sync(); |
| cin.get(user\_choice); |
| } |
| switch (user\_choice) { |
| case '1': // USER CREATION |
| userCreator(username,password,user\_list); |
| break; |
| case '2': // USER DELETION |
| userRemover(user\_choice,user\_list); |
| break; |
| case '3': // USER AUTHENTICATION |
| userAuthenticator(username,password,authenticate); |
| break; |
| case '4': // EXIT PROGRAM |
| exitProgram(); |
| return 0; |
| default: // INVALID INPUT |
| invalidInput("Please choose between 1, 2, 3 or 4!"); |
| } |
| } |
| // |
| // |
| // Successfully logged in |
| // |
| // |
| while (authenticate == true) { // USER MENU |
| char user\_choice; |
| string user\_checklist = username + ".txt"; |
| vector<string> task\_list (createList(user\_checklist)); |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Welcome",78)<<"|\ |
| |"<< centerStr(username,78)<<"|\ |
| |"<< centerStr("What would you like to do?",78)<<"|\ |
| |"<< centerStr("0 - Logout ",78)<<"|\ |
| |"<< centerStr("1 - Create a task",78)<<"|\ |
| |"<< centerStr("2 - Delete a task",78)<<"|\ |
| |"<< centerStr("3 - Tick a task ",78)<<"|\ |
| |"<< centerStr("4 - Quit ",78)<<"|\ |
| |"<< centerStr("(Enter the number before an operation to start)",78)<<"|\ |
| |"<< centerStr("(For example: key in 1 if you want to create a task)",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| if (task\_list.size() > 0) { |
| listTasks(task\_list); |
| } else { |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("You don't have any task(s).",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| } |
| if (user\_choice != '2' && user\_choice != '3') { |
| cout << ">> "; |
| cin.clear(); |
| cin.sync(); |
| cin.get(user\_choice); |
| } |
| switch (user\_choice){ |
| case '0': |
| authenticate = false; |
| break; |
| case '1': // TASK CREATION |
| taskCreator(username,task\_list); |
| break; |
| case '2': // TASK DELETION |
| taskRemover(username,user\_choice,task\_list); |
| break; |
| case '3': // TASK COMPLETION |
| taskMarker(username,user\_choice,task\_list); |
| break; |
| case '4': // EXIT PROGRAM |
| exitProgram(); |
| return 0; |
| default: // INVALID INPUT |
| invalidInput("Please choose between 0, 1, 2, 3 or 4!"); |
| } |
| } |
| } while (1==1); |
| return 0; |
| } |

**main.hpp**

|  |
| --- |
| **Pseudocode** |
| Function createList(parameter: file\_name of type string) of type vector string |
| Declare variable text\_list of type vector string |
| Create input file stream infile(parameters: file\_name, read mode) |
| Declare variable line of type string |
| WHILE getline(parameters: infile,line) |
| Add line into text\_list |
| END WHILE |
| RETURN text\_list |
| End Function |
|  |
| Function verifyUser(parameter: username1 of type string) of type boolean |
| Declare variable test\_bool of type boolean = FALSE |
| Create input file stream infile(parameters: "users.txt" , read mode) |
| Declare variable line of type string |
| WHILE getline(parameters: infile, line) |
| Declare variable username2 of type string = substring of line from position 0 to the position of “,” |
| IF username1 EQUAL username2 |
| Close input file stream infile |
| test\_bool = TRUE |
| BREAK |
| END IF |
| END WHILE |
| RETURN test\_bool |
| End Function |
|  |
| Subroutine createUser(parameters: username of type string, password of type string) |
| Declare variable user\_info of type string = username + "," + password |
| Declare variable user\_checklist of type string = username + ".txt" |
| Create output file stream outfile1(parameters: "users.txt", append mode) |
| Create output file stream outfile2(parameters: user\_checklist, append mode) |
| Output user\_info to file stream outfile1 |
| Output "" to file stream outfile2 |
| Close output file stream outfile1 |
| Close output file stream outfile2 |
| End Subroutine |
|  |
| Subroutine listUsers(parameter: user\_list of type vector string) |
| Declare variable user\_id of type integer |
| DISPLAY "User(s): " |
|
|
|
|
| FOR each i in user\_list |
| user\_id = i + 1 |
| Declare variable user\_info of type string = the string at position i of user\_list |
| Declare variable long\_username of type string = the substring of user\_info from position 0 to the position of "," |
| Declare variable split\_username of type vector string = CALL splitStrPos(parameters: long\_username, 73) |
| IF user\_id LESS THAN 10 |
| DISPLAY user\_id |
| ELSE IF user\_id MORE THAN OR EQUAL TO 10 |
| DISPLAY user\_id |
| END IF |
| FOR each u in split\_username |
| IF u EQUAL 0 |
| DISPLAY the string at position u of split\_username |
| ELSE |
| DISPLAY the string at position u of split\_username |
| END IF |
| END FOR |
| END FOR |
| DISPLAY line |
|
|
| End Subroutine |
|  |
| Subroutine deleteUser(parameters: user\_id of type integer, user\_list of type vector string) |
| Declare variable new\_list of type vector string |
| FOR each i in user\_list |
| IF user\_id NOT EQUAL i |
| Add the string at position i of user\_list into new\_list |
| END IF |
| END FOR |
|  |
| Declare variable user\_info of type string = the string at position user\_id of user\_list |
| Declare variable username of type string = substring of user\_info from position 0 to the position of the “,” |
| Declare variable user\_checklist of type string = username + ".txt" |
| Remove the file with the name user\_checklist |
|  |
| Create output file stream outfile(parameters: "users.txt", write mode) |
| FOR each i in new\_list |
| Output the string at position i of new\_list to file stream outfile |
| END FOR |
| Close the output file stream outfile |
| End Subroutine |
|  |
| Function loginUser(parameters: username of type string, password of type string) of type boolean |
| Declare variable test\_bool of type boolean = FALSE |
| Declare variable user\_info of type string = username + "," + password |
| Create input file stream infile(parameters: "users.txt", read mode) |
| Declare variable line of type string |
| WHILE getline(parameters: infile,line) |
| IF user\_info EQUAL line |
| Close input file stream infile |
| test\_bool = TRUE |
| BREAK |
| END IF |
| END WHILE |
| RETURN test\_bool |
| End Function |
|  |
| Function verifyTask(parameters: username of type string, task of type string) of type boolean |
| Declare variable test\_bool of type boolean = FALSE |
| Declare variable user\_checklist of type string = username + ".txt" |
| Create input file stream infile(parameters: user\_checklist, read mode) |
| Declare variable line of type string |
| WHILE getline(parameters: infile,line) |
| IF task EQUAL line |
| Close input file stream infile |
| test\_bool = TRUE |
| BREAK |
| END IF |
| END WHILE |
| RETURN test\_bool |
| End Function |
|  |
| Subroutine createTask(parameters: username of type string, task of type string) |
| Declare variable user\_checklist of type string = username + ".txt" |
| Create output file stream outfile(parameters: user\_checklist, append mode) |
| Output task to file stream outfile |
| Close output file stream outfile |
| End Subroutine |
|  |
| Subroutine listTasks(parameter: task\_list of type vector string) |
| Declare variable task\_id of type integer |
| DISPLAY "Your task(s): " |
|
|
|
|
| FOR each i in task\_list |
| task\_id = i + 1 |
| Declare variable split\_task of type vector string = CALL splitStrPos(parameters: the string at position i of task\_list, 73) |
| IF task\_id LESS THAN 10 |
| DISPLAY task\_id |
| ELSE IF task\_id MORE THAN OR EQUAL TO 10 |
| DISPLAY task\_id |
| END IF |
| FOR each u in split\_task |
| IF u EQUAL 0 |
| DISPLAY the string at position u of split\_task |
| ELSE |
| DISPLAY the string at position u of split\_task |
| END IF |
| END FOR |
| END FOR |
| DISPLAY line |
|
|
| End Subroutine |
|  |
| Subroutine deleteTask(parameters: username of type string, task\_id of type integer, task\_list of type vector string) |
| Declare variable new\_list of type vector string |
| FOR each i in task\_list |
| IF task\_id NOT EQUAL i |
| Add the string at position i of task\_list into new\_list |
| END IF |
| END FOR |
|  |
| Declare variable user\_checklist of type string = username + ".txt" |
| Create output file stream outfile(parameters: user\_checklist, write mode) |
| FOR each i in new\_list |
| Output the string at position i of new\_list into file stream outfile |
| END FOR |
| Close output file stream outfile |
| End Subroutine |
|  |
| Subroutine tickTask(parameters: username of type string, task\_id of type integer, task\_list of type vector string) |
| Declare variable test\_bool of type boolean |
| Declare variable new\_list of type vector string |
| FOR each i in task\_list |
| IF task\_id EQUAL i |
| IF the string at position i of task\_list contains the string "COMPLETED: " |
| Declare variable task of type string = "COMPLETED: " + the string at position i of task\_list |
| Add task into new\_list |
| test\_bool = true |
| ELSE |
| Declare variable task of type string = CALL eraseStr(parameters: the value at position i of task\_list, "COMPLETED: ") |
| Add task into new\_list |
| test\_bool = false |
| END IF |
| ELSE |
| Add the string at position i of task\_list into new\_list |
| END IF |
| END FOR |
|  |
| Declare variable user\_checklist of type string = username + ".txt" |
| Create output file stream outfile(parameters: user\_checklist, write mode) |
| FOR each i in new\_list |
| Output the string at position i of new\_list into file stream outfile |
| END FOR |
| Close output file stream outfile |
| RETURN test\_bool |
| End Subroutine |
|  |
| Subroutine userCreator(parameters: username of type string by ref, password of type string by ref, user\_list of type vector string by ref) |
| IF size of user\_list LESS THAN 99 |
| Clear any state on cin |
| Sync cin stream |
| Clear screen |
| DISPLAY “Please key in your preferred username: (no limitation)“ |
|
|
|
| Ask the user to enter username |
| GET username |
| DISPLAY “Please key in your preferred password: (no limitation)” |
|
|
|
| Ask the user to enter password |
| GET password |
| IF username NOT EQUALS BLANK AND password NOT EQUALS blank |
| IF verifyUser(parameter: username) NOT EQUALS TRUE |
| CALL createUser(parameters:username,password) |
| Clear screen |
| DISPLAY “User successfully created!” |
|
|
|
| Wait for user input to continue |
| ELSE |
| Clear screen |
| DISPLAY “User already exists.” |
|
|
|
| Wait for user input to continue |
| END IF |
| ELSE |
| Clear screen |
| DISPLAY “Invalid input.” |
|
|
|
| Wait for user input to continue |
| END IF |
| ELSE |
| Clear screen |
| DISPLAY “You can only have a maximum of 99 users at the moment.” |
|
|
|
| Wait for user input to continue |
| END IF |
| End Subroutine |
|  |
| Subroutine userRemover(parameters: user\_choice of type char by ref, user\_list of type vector string by ref) |
| Declare variable input\_id of type string |
| Declare variable password1 of type string |
| IF size of user\_list MORE THAN 0 |
| Clear screen |
| DISPLAY Instructions |
|
|
|
|
|
| CALL listUsers(parameters:user\_list) |
| Ask the user to enter input\_id |
| Clear any state on cin |
| Sync cin stream |
| GET input\_id |
| IF input\_id EQUALS “0” |
| user\_choice = 'w' |
| ELSE |
| Declare variable conv\_id of type integer = input\_id converted to integer |
| IF conv\_id NOT EQUAL 0 |
| Declare variable user\_id of type integer = conv\_id - 1 |
| IF user\_id LESS THAN size of user\_list |
| Clear any state on cin |
| Sync cin stream |
| Clear screen |
| DISPLAY "Please enter your password: " |
|
|
|
| Ask the user to enter password1 |
| GET password1 |
|  |
| Declare variable user\_info of type string = the string at position user\_id of user\_list |
| Declare variable password2 of type string = the substring of user\_info from the position of ","+1 until the end |
| IF password1 EQUALS password2 |
| CALL deleteUser(parameters: user\_id,user\_list) |
| Clear screen |
| DISPLAY "User successfully deleted." |
|
|
|
| Wait for user input to continue |
| ELSE |
| Clear screen |
| DISPLAY "The password that you have entered does not match." |
|
|
|
| Wait for user input to continue |
| END IF |
| ELSE |
| Clear screen |
| DISPLAY "There is no user with that ID." |
|
|
|
| Wait for user input to continue |
| END IF |
| ELSE |
| Clear screen |
| DISPLAY “Please key in the number before the user.” |
|
|
|
| Wait for user input to continue |
| END IF |
| END IF |
| ELSE |
| user\_choice = 'w' |
| Clear screen |
| DISPLAY “There are no user(s).” |
|
|
|
| Wait for user input to continue |
| END IF |
| End Subroutine |
|  |
| Subroutine userAuthenticator(parameters: username of type string by ref, password of type string by ref, authenticate of type boolean by ref) |
| Clear any state on cin |
| Sync cin stream |
| Clear screen |
| DISPLAY “Username:” |
|
|
|
| Ask the user to enter username |
| GET username |
| DISPLAY “Password:” |
|
|
|
| Ask the user to enter password |
| GET password |
| IF username NOT EQUALS BLANK AND password NOT EQUALS blank |
| IF loginUser(parameter:username,password) EQUALS TRUE |
| authenticate = TRUE |
| ELSE |
| Clear screen |
| DISPLAY “User authentication failed.” |
|
|
|
| Wait for user input to continue |
| END IF |
| ELSE |
| Clear screen |
| DISPLAY “Invalid input.” |
|
|
|
| Wait for user input to continue |
| END IF |
| End Subroutine |
|  |
| Subroutine exitProgram |
| Clear screen |
| DISPLAY "Thank you for using MyCheckList!" |
|
|
|
| Wait for user input to continue |
| End Subroutine |
|  |
| Subroutine invalidInput(parameter: error\_msg of type string) |
| Clear screen |
| DISPLAY error\_msg |
|
|
|
| Wait for user input to continue |
| End Subroutine |
|  |
| Subroutine taskCreator(parameters: username of type string by ref, task\_list of type vector string by ref) |
| IF size of task\_list LESS THAN 99 |
| Declare variable task of type string |
| Clear any state on cin |
| Sync cin stream |
| Clear screen |
| DISPLAY “Please key in your task.” |
|
|
|
| Ask the user to enter task |
| GET task |
| IF task NOT EQUALS BLANK |
| IF verifyTask(parameters:username,task) NOT EQUALS TRUE |
| CALL createTask(parameters:username,task) |
| Clear screen |
| DISPLAY “Task successfully created!” |
|
|
|
| Wait for user input to continue |
| ELSE |
| Clear screen |
| DISPLAY “Task already exists.” |
|
|
|
| Wait for user input to continue |
| END IF |
| ELSE |
| Clear screen |
| DISPLAY “Invalid input.” |
|
|
|
| Wait for user input to continue |
| END IF |
| ELSE |
| Clear screen |
| DISPLAY “You can only have a maximum of 99 tasks at the moment.” |
|
|
|
| Wait for user input to continue |
| END IF |
| End Subroutine |
|  |
| Subroutine taskRemover(parameters: username of type string by ref, user\_choice of type char by ref, task\_list of type vector string by ref) |
| Declare variable input\_id of type string |
| IF size of task\_list MORE THAN 0 |
| Clear screen |
| DISPLAY Instructions |
|
|
|
|
|
| CALL listTasks(parameters:task\_list,”Your tasks:”) |
| Ask the user to enter input\_id |
| Clear any state on cin |
| Sync cin stream |
| GET input\_id |
| IF input\_id EQUALS “0” |
| user\_choice = 'w' |
| ELSE |
| Declare variable conv\_id of type integer = input\_id converted to integer |
| IF conv\_id NOT EQUAL 0 |
| Declare variable task\_id of type integer = conv\_id - 1 |
| IF task\_id LESS THAN size of task\_list |
| CALL deleteTask(parameters: username,task\_id,task\_list) |
| Clear screen |
| DISPLAY "Task successfully deleted." |
|
|
|
| Wait for user input to continue |
| ELSE |
| Clear screen |
| DISPLAY "There is no task with that ID" |
|
|
|
| Wait for user input to continue |
| END IF |
| ELSE |
| Clear screen |
| DISPLAY "Please key in the number before the task" |
|
|
|
| Wait for user input to continue |
| END IF |
| END IF |
| ELSE |
| user\_choice = 'w' |
| Clear screen |
| DISPLAY “You don't have any task(s).” |
|
|
|
| Wait for user input to continue |
| END IF |
| End Subroutine |
|  |
| Subroutine taskMarker(parameters:username of type string by ref, user\_choice of type char by ref, task\_list of type vector string by ref) |
| Declare variable input\_id of type string |
| IF size of task\_list MORE THAN 0 |
| Clear screen |
| DISPLAY Instructions |
|
|
|
|
|
| CALL listTasks(parameters:task\_list,”Your tasks:”) |
| Ask the user to enter input\_id |
| Clear any state on cin |
| Sync cin stream |
| GET input\_id |
| IF input\_id EQUALS “0” |
| user\_choice = 'w' |
| ELSE |
| Declare variable conv\_id of type integer = input\_id converted to integer |
| IF conv\_id NOT EQUAL 0 |
| Declare variable task\_id of type integer = conv\_id - 1 |
| IF task\_id LESS THAN size of task\_list |
| IF tickTask(parameters: username,task\_id,task\_list) EQUALS TRUE |
| Clear screen |
| DISPLAY "Task successfully ticked." |
|
|
|
| Wait for user input to continue |
| ELSE |
| Clear screen |
| DISPLAY "Task successfully unticked." |
|
|
|
| Wait for user input to continue |
| END IF |
| ELSE |
| Clear screen |
| DISPLAY "There is no task with that ID." |
|
|
|
| Wait for user input to continue |
| END IF |
| ELSE |
| Clear screen |
| DISPLAY “Please key in the number before the task.” |
|
|
|
| Wait for user input to continue |
| END IF |
| END IF |
| ELSE |
| user\_choice = 'w' |
| Clear screen |
| DISPLAY “You don't have any task(s).” |
|
|
|
| Wait for user input to continue |
| END IF |
| End Subroutine |

|  |
| --- |
| **C++** |
| vector<string> createList(string file\_name) { |
| vector<string> text\_list; |
| ifstream infile(file\_name.c\_str(), ios::in); |
| string line; |
| while (getline(infile,line)) { |
| text\_list.push\_back(line); |
| } |
| return text\_list; |
| } |
|  |
| bool verifyUser(string username1) { |
| bool test\_bool = false; |
| ifstream infile("users.txt", ios::in); |
| string line; |
| while (getline(infile,line)) { |
| string username2 = line.substr(0,line.find(",")); |
| if (username1 == username2) { |
| infile.close(); |
| test\_bool = true; // User exists |
| break; |
| } |
| } |
| return test\_bool; |
| } |
|  |
| void createUser(string username,string password) { |
| string user\_info = username + "," + password; |
| string user\_checklist = username + ".txt"; |
| ofstream outfile1("users.txt", ios\_base::app); |
| ofstream outfile2(user\_checklist.c\_str(), ios\_base::app); |
| outfile1 << user\_info << endl; |
| outfile2 << ""; |
| outfile1.close(); |
| outfile2.close(); |
| } |
|  |
| void listUsers(vector<string> user\_list) { |
| int user\_id; |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("User(s): ",78) <<"|\ |
| --------------------------------------------------------------------------------\ |
| | |"; |
| for (int i = 0 ; i < user\_list.size(); i++) { |
| user\_id = i + 1; |
| string user\_info = user\_list.at(i); |
| string long\_username = user\_info.substr(0,user\_info.find(",")); |
| vector<string> split\_username (splitStrPos(long\_username,73)); |
| if (user\_id < 10) { |
| cout << "| " << user\_id << " . "; |
| } else { |
| cout << "| " << user\_id << ". "; |
| } |
| for (int u = 0 ; u < split\_username.size() ; u++) { |
| if (u == 0) { |
| cout << bSpaceStr(split\_username.at(u),73) << "|"; |
| } else { |
| cout << "| " << bSpaceStr(split\_username.at(u),73) << "|"; |
| } |
| } |
| } |
| cout << "\ |
| | |\ |
| --------------------------------------------------------------------------------"; |
| } |
|  |
| void deleteUser(int user\_id,vector<string> user\_list) { |
| vector<string> new\_list; |
| for (int i = 0; i < user\_list.size() ; i++) { |
| if (user\_id != i) { |
| new\_list.push\_back(user\_list.at(i)); |
| } |
| } |
|  |
| string user\_info = user\_list.at(user\_id); |
| string username = user\_info.substr(0,user\_info.find(",")); |
| string user\_checklist = username + ".txt"; |
| remove(user\_checklist.c\_str()); |
|  |
| ofstream outfile("users.txt", ios::out); |
| for (int i = 0;i < new\_list.size();i++) { |
| outfile << new\_list.at(i) << endl; |
| } |
| outfile.close(); |
| } |
|  |
| bool loginUser(string username,string password) { |
| bool test\_bool = false; |
| string user\_info = username + "," + password; |
| ifstream infile("users.txt",ios::in); |
| string line; |
| while (getline(infile,line)) { |
| if (user\_info == line) { |
| infile.close(); |
| test\_bool = true; // Login success |
| break; |
| } |
| } |
| return test\_bool; |
| } |
|  |
| bool verifyTask(string username,string task) { |
| bool test\_bool = false; |
| string user\_checklist = username + ".txt"; |
| ifstream infile(user\_checklist.c\_str(), ios::in); |
| string line; |
| while (getline(infile,line)) { |
| if (task == line) { |
| infile.close(); |
| test\_bool = true; // Task exists |
| break; |
| } |
| } |
| return test\_bool; |
| } |
|  |
| void createTask(string username,string task) { |
| string user\_checklist = username + ".txt"; |
| ofstream outfile(user\_checklist.c\_str(),ios\_base::app); |
| outfile << task << endl; |
| outfile.close(); |
| } |
|  |
| void listTasks(vector<string> task\_list) { |
| int task\_id; |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Your task(s): ",78) <<"|\ |
| --------------------------------------------------------------------------------\ |
| | |"; |
| for (int i = 0 ; i < task\_list.size(); i++) { |
| task\_id = i + 1; |
| vector<string> split\_task (splitStrPos(task\_list.at(i),73)); |
| if (task\_id < 10) { |
| cout << "| " << task\_id << " . "; |
| } else { |
| cout << "| " << task\_id << ". "; |
| } |
| for (int u = 0 ; u < split\_task.size() ; u++) { |
| if (u == 0) { |
| cout << bSpaceStr(split\_task.at(u),73) << "|"; |
| } else { |
| cout << "| " << bSpaceStr(split\_task.at(u),73) << "|"; |
| } |
| } |
| } |
| cout << "\ |
| | |\ |
| --------------------------------------------------------------------------------"; |
| } |
|  |
| void deleteTask(string username, int task\_id, vector<string> task\_list) { |
| vector<string> new\_list; |
| for (int i = 0 ; i < task\_list.size() ; i++) { |
| if (task\_id != i) { |
| new\_list.push\_back(task\_list.at(i)); |
| } |
| } |
|  |
| string user\_checklist = username + ".txt"; |
| ofstream outfile(user\_checklist.c\_str(),ios::out); |
| for (int i = 0 ; i < new\_list.size() ; i++) { |
| outfile << new\_list.at(i) << endl; |
| } |
| outfile.close(); |
| } |
|  |
| bool tickTask(string username, int task\_id, vector<string> task\_list) { |
| bool test\_bool; |
| vector<string> new\_list; |
| for (int i = 0 ; i < task\_list.size() ; i++) { |
| if (task\_id == i) { |
| if ((task\_list.at(i).find("COMPLETED: ")) == string::npos) { |
| string task = "COMPLETED: " + task\_list.at(i); |
| new\_list.push\_back(task); |
| test\_bool = true; |
| } else { |
| string task = eraseStr(task\_list.at(i),"COMPLETED: "); |
| new\_list.push\_back(task); |
| test\_bool = false; |
| } |
| } else { |
| new\_list.push\_back(task\_list.at(i)); |
| } |
| } |
|  |
| string user\_checklist = username + ".txt"; |
| ofstream outfile(user\_checklist.c\_str(),ios::out); |
| for (int i = 0 ; i < new\_list.size() ; i++) { |
| outfile << new\_list.at(i) << endl; |
| } |
| outfile.close(); |
| return test\_bool; |
| } |
|  |
| void userCreator(string& username,string& password,vector<string>& user\_list){ |
| if (user\_list.size() < 99) { |
| cin.clear(); |
| cin.sync(); |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Please key in your preferred username: (no limitation)",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| cout << ">> "; |
| getline(cin, username); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Please key in your preferred password: (no limitation)",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| cout << ">> "; |
| getline(cin, password); |
| if (username != "" && password != "") { |
| if (!verifyUser(username)) { |
| createUser(username,password); |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("User successfully created!",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("User already exists.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Invalid input.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("You can only have a maximum of 99 users at the moment.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } |
|  |
| void userRemover(char& user\_choice,vector<string>& user\_list){ |
| string input\_id; |
| string password1; |
| if (user\_list.size() > 0) { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Delete a user by entering the number before the user",78)<<"|\ |
| |"<< centerStr("0 - Back",78)<<"|\ |
| |"<< centerStr("(Key in 0 if you want to go back to the main menu)",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| listUsers(user\_list); |
| cout << ">> "; |
| cin.clear(); |
| cin.sync(); |
| cin >> input\_id; |
| if (input\_id == "0") { |
| user\_choice = 'w'; |
| } else { |
| int conv\_id = atoi(input\_id.c\_str()); |
| if (conv\_id != 0) { |
| int user\_id = conv\_id - 1; |
| if (user\_id < user\_list.size()) { |
| cin.clear(); |
| cin.sync(); |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Please enter your password: ",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| cout << ">> "; |
| getline(cin,password1); |
|  |
| string user\_info = user\_list.at(user\_id); |
| string password2 = user\_info.substr(user\_info.find(",")+1); |
| if (password1 == password2) { |
| deleteUser(user\_id,user\_list); |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"+centerStr("User successfully deleted.",78)+"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"+centerStr("The password that you have entered does not match.",78)+"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"+centerStr("There is no user with that ID.",78)+"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Please key in the number before the user.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } |
| } else { |
| user\_choice = 'w'; |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("There are no user(s).",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } |
|  |
| void userAuthenticator(string& username,string& password,bool& authenticate){ |
| cin.clear(); |
| cin.sync(); |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Username: ",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| cout << ">> "; |
| getline(cin, username); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Password: ",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| cout << ">> "; |
| getline(cin, password); |
| if (username != "" && password != "") { |
| if (loginUser(username,password)) { |
| authenticate = true; |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("User authentication failed.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Invalid input.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } |
|  |
| void exitProgram(){ |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Thank you for using MyCheckList!",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
|  |
| void invalidInput(string error\_msg){ |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr(error\_msg,78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
|  |
| void taskCreator(string& username,vector<string>& task\_list){ |
| if (task\_list.size() < 99) { |
| string task; |
| cin.clear(); |
| cin.sync(); |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Please key in your task: ",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| cout << ">> "; |
| getline(cin, task); |
| if (task != "") { |
| if (!verifyTask(username,task)) { |
| createTask(username,task); |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Task successfully created!",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Task already exists.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Invalid input.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("You can only have a maximum of 99 tasks at the moment.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } |
|  |
| void taskRemover(string& username,char& user\_choice,vector<string>& task\_list){ |
| string input\_id; |
| if (task\_list.size() > 0) { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Delete a task by entering the number before the task",78)<<"|\ |
| |"<< centerStr("0 - Back",78)<<"|\ |
| |"<< centerStr("(Key in 0 if you want to go back to the user menu)",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| listTasks(task\_list); |
| cout << ">> "; |
| cin.clear(); |
| cin.sync(); |
| cin >> input\_id; |
| if (input\_id == "0") { |
| user\_choice = 'w'; |
| } else { |
| int conv\_id = atoi(input\_id.c\_str()); |
| if (conv\_id != 0) { |
| int task\_id = conv\_id - 1; |
| if (task\_id < task\_list.size()) { |
| deleteTask(username,task\_id,task\_list); |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"+centerStr("Task successfully deleted.",78)+"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"+centerStr("There is no task with that ID.",78)+"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Please key in the number before the task.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } |
| } else { |
| user\_choice = 'w'; |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("You don't have any task(s).",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } |
|  |
| void taskMarker(string& username,char& user\_choice,vector<string>& task\_list){ |
| string input\_id; |
| if (task\_list.size() > 0) { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Tick a task by inputting the number before the task",78)<<"|\ |
| |"<< centerStr("0 - Back",78)<<"|\ |
| |"<< centerStr("(Key in 0 if you want to go back to the user menu)",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| listTasks(task\_list); |
| cout << ">> "; |
| cin.clear(); |
| cin.sync(); |
| cin >> input\_id; |
| if (input\_id == "0") { |
| user\_choice = 'w'; |
| } else { |
| int conv\_id = atoi(input\_id.c\_str()); |
| if (conv\_id != 0) { |
| int task\_id = conv\_id - 1; |
| if (task\_id < task\_list.size()) { |
| if (tickTask(username,task\_id,task\_list)){ |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"+centerStr("Task successfully ticked.",78)+"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"+centerStr("Task successfully unticked.",78)+"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"+centerStr("There is no task with that ID.",78)+"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } else { |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("Please key in the number before the task.",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } |
| } else { |
| user\_choice = 'w'; |
| system("cls"); |
| cout << "\ |
| --------------------------------------------------------------------------------\ |
| |"<< centerStr("You don't have any task(s).",78)<<"|\ |
| --------------------------------------------------------------------------------"; |
| system("pause"); |
| } |
| } |

**zString.hpp**

|  |
| --- |
| **Pseudocode** |
| Function centerStr(parameters: input\_s of type string, container\_width of type integer) of type string |
| Declare variable new\_s of type string |
| Declare variable s\_mid\_point of type integer = (container\_width - length of input\_s) / 2 |
| FOR each i in (container\_width + 1) |
| IF i EQUALS to s\_mid\_point |
| new\_s += input\_s |
| i = i + length of input\_s |
| ELSE |
| new\_s += " " |
| END IF |
| END FOR |
| RETURN new\_s |
| End Function |
|  |
| Function bSpaceStr(parameters: input\_s of type string, container\_width of type integer) of type string |
| Declare variable new\_s of type string |
| FOR each i in (container\_width + 1) |
| IF i EQUALS to 0 |
| new\_s += input\_s |
| i = i + length of input\_s |
| ELSE |
| new\_s += " " |
| END IF |
| END FOR |
| RETURN new\_s |
| End Function |
|  |
| Function splitStrPos(parameters: input\_s of type string, max\_width of type integer) of type vector string |
| Declare variable new\_s of type vector string |
| Declare varibale token of type string |
| WHILE length of input\_s > max\_width |
| token = substring of input\_s from position 0 to position max\_width |
| Add token into new\_s |
| Erase characters in input\_s starting from position 0 until max\_width |
| END WHILE |
| Add input\_s into new\_s |
| RETURN new\_s |
| End Function |
|  |
| Function eraseStr(parameters: str\_to\_use of type string, str\_to\_erase of type string) of type string |
| Declare variable new\_s of type string |
| new\_s = erase str\_to\_erase in str\_to\_use |
| RETURN new\_s |
| End Function |

|  |
| --- |
| **C++** |
| string centerStr(string input\_s,int container\_width) { |
| string new\_s; |
| int s\_mid\_point = (container\_width - input\_s.length()) / 2; |
| for (int i = 0 ; i < container\_width+1 ; i++) { |
| if (i == s\_mid\_point) { |
| new\_s += input\_s; |
| i = i + input\_s.length(); |
| } else { |
| new\_s += " "; |
| } |
| } |
| return new\_s; |
| } |
|  |
| string bSpaceStr(string input\_s,int container\_width) { |
| string new\_s; |
| for (int i = 0 ; i < container\_width+1 ; i++) { |
| if (i == 0) { |
| new\_s += input\_s; |
| i = i + input\_s.length(); |
| } else { |
| new\_s += " "; |
| } |
| } |
| return new\_s; |
| } |
|  |
| vector<string> splitStrPos(string input\_s,int max\_width) { |
| vector<string> new\_s; |
| string token; |
| while (input\_s.length() > max\_width) { |
| token = input\_s.substr(0, max\_width); |
| new\_s.push\_back(token); |
| input\_s.erase(0, max\_width); |
| } |
| new\_s.push\_back(input\_s); |
| return new\_s; |
| } |
|  |
| string eraseStr(string str\_to\_use,string str\_to\_erase) { |
| string new\_s; |
| new\_s = str\_to\_use.erase(str\_to\_use.find(str\_to\_erase),str\_to\_erase.length()); |
| return new\_s; |
| } |

# FLOWCHART







# CONCLUSION & COMMENTS

In conclusion, we have learnt a lot when conducting this project. We’ve learnt that planning and designing is the most important phase in any software development life cycle and the importance of a matching documentation.

In the future, given enough time, we would like to:

1. Get user input without pressing enter to minimize the steps needed to perform an operation.
2. Fix the issue that will occur when changing the width of the command prompt.
3. Add restrictions or limitations when creating a new user or new task.
4. Change the color scheme of the command prompt for user attractiveness.
5. Build a GUI version of this program (using Qt GUI framework) so that it is more user-friendly and easily usable.