NATIONAL UNIVERSITY OF HO CHI MINH CITY

HO CHI MINH UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ASSIGNMENT REPORT

SUBJECT: DIGITAL SYSTEMS

Instructor: MR.TRAN TUAN ANH

No.	Name	ID	Contribution
1	PHAM TRONG NHAN	1752394	
2	HUYNH NGUYEN THANH HAI	1752016	
3	TRANG NGOC THAO NGUYEN	1752037	
4	PHAM DOAN TRANG	1752550	
5	HUYNH THAI DUONG	1752150	
6	HUYNH NGOC THIEN	1752051	

Class: CO1011

1 Introduction

This is big assigment for Programming Fundamentals class.
Our target is build a program for University system which the user can be:

- Admin
- Professor
- Student

Each type of user has different tasks and actions to work the system.
To finish the assignment, we use many technique and functions which learnt in the course.

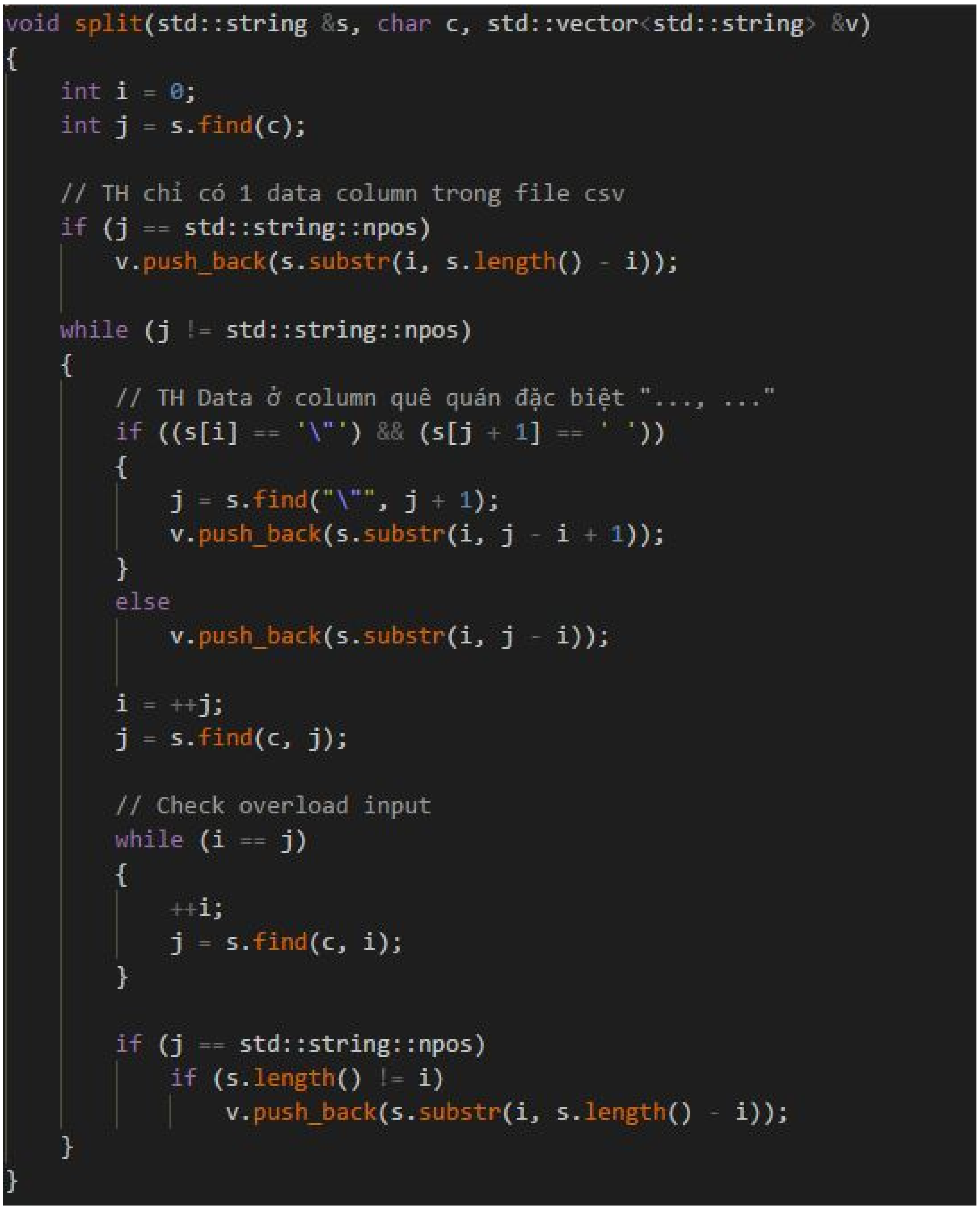
- File manipulate in **fstream** library.
- String processing with **string** library.
- Vector usage for storing data.
- Pass by reference technique.

Also, the program can work well with error - exception handling.
Overall, this program operates well with fully-equiped functions.

2 Main activity

2.1 Basic function

2.1.1 Void split



```cpp
void split(std::string &s, char c, std::vector<std::string> &v)
{
    int i = 0;
    int j = s.find(c);

    // TH chỉ có 1 data column trong file csv
    if (j == std::string::npos)
        v.push_back(s.substr(i, s.length() - i));

    while (j != std::string::npos)
    {
        // TH Data ở column quê quán đặc biệt "..., ..."
        if ((s[i] == '\"') && (s[j + 1] == ' '))
        {
            j = s.find("\"", j + 1);
            v.push_back(s.substr(i, j - i + 1));
        }
        else
            v.push_back(s.substr(i, j - i));

        i = ++j;
        j = s.find(c, j);

        // Check overload input
        while (i == j)
        {
            ++i;
            j = s.find(c, i);
        }

        if (j == std::string::npos)
            if (s.length() != i)
                v.push_back(s.substr(i, s.length() - i));
    }
}
```

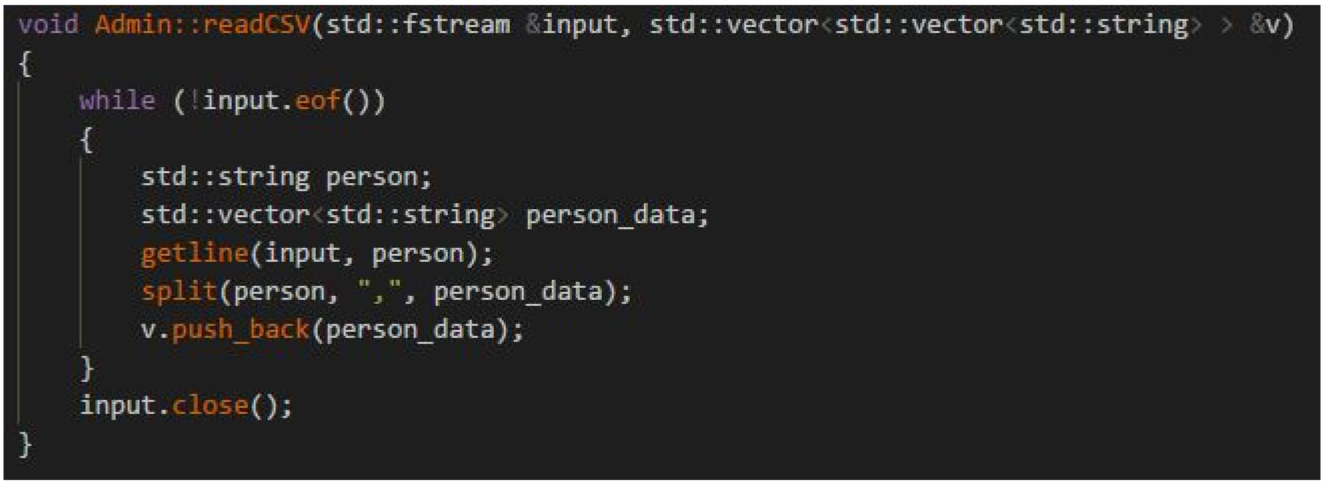
The **Split** function will take string which need to be split as argument. The function will split by character **c** into a vector to store. This function use in 2 case:

1. Split each line of CSV file into vector to store personal data.
2. Split input command into command keyword and requirement of command.

The **Split** function can find some special case to avoid error.

- Can define which file has only 1 column information.
- Can identify the comma in Address column to the comma in CSV file.
- Can truncate the redundant space in the input command.

2.1.2 Void ReadCSV

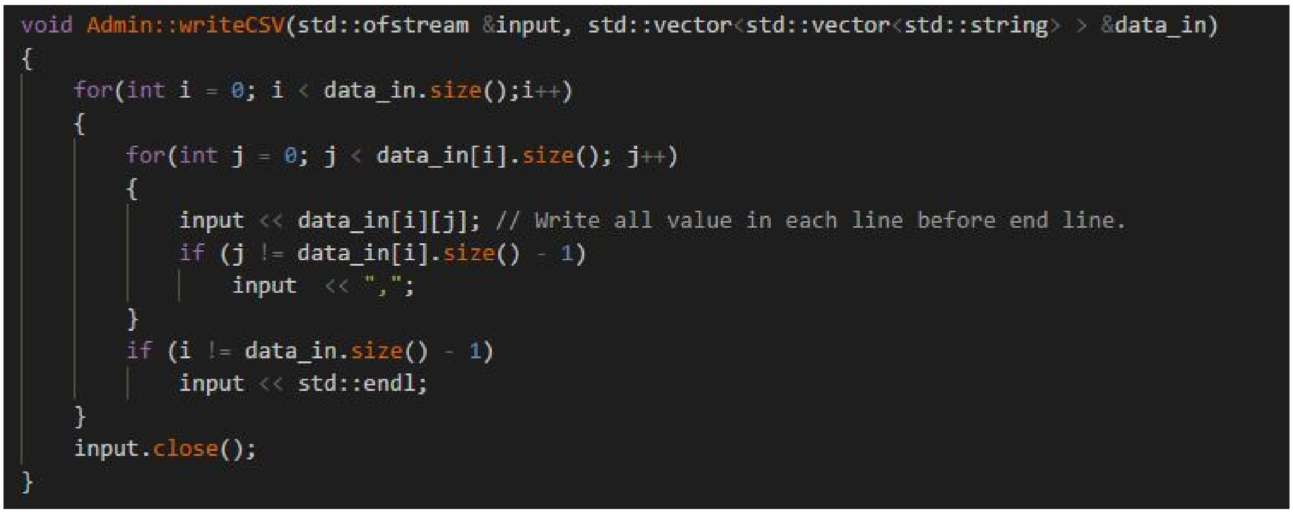


```
void Admin::readCSV(std::fstream &input, std::vector<std::vector<std::string> > &v)
{
    while (!input.eof())
    {
        std::string person;
        std::vector<std::string> person_data;
        getline(input, person);
        split(person, ",", person_data);
        v.push_back(person_data);
    }
    input.close();
}
```

The **readCSV** function is used to read all data from csv file into 1 single 2D vector.
Step to perform this action:

- The function take the file stream object as the input source and make change on 2D vector **v**.
- To read from the beginning to the end, the while loop identify the end of file. If not, it will read 1 single line and pass that string line into **split**.
- The **split** will split line by character ",".
- At the end of function, the vector person_data will received information of 1 line without comma.
- After that, that 1D vector push into at the end of 2D vector **v**.

2.1.3 Void WriteCSV

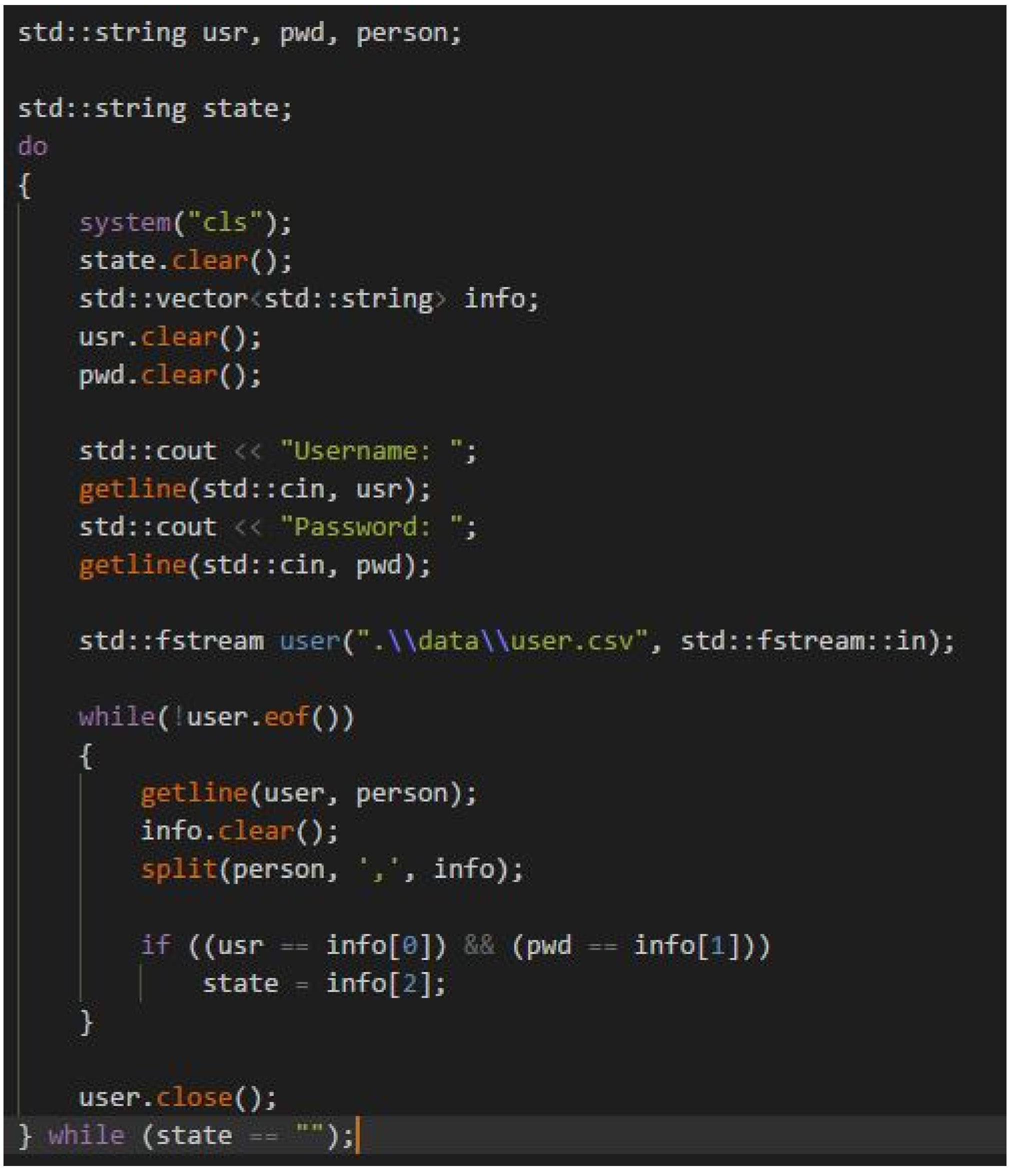


```
void Admin::writeCSV(std::ofstream &input, std::vector<std::vector<std::string> > &data_in)
{
    for(int i = 0; i < data_in.size();i++)
    {
        for(int j = 0; j < data_in[i].size(); j++)
        {
            input << data_in[i][j]; // Write all value in each line before end line.
            if (j != data_in[i].size() - 1)
                input  << ",";
        }
        if (i != data_in.size() - 1)
            input << std::endl;
    }
    input.close();
}
```

The **writeCSV** function is used to write 2D vector to csv file.
There are some notice in writing csv file:

1. On 1 line, after writing 1 information, must write down the comma to separate it with the next one.
2. After the final information on line, instead of writing comma, function must end of the line and go to the next line.
3. After the final line in csv file, there is no blank line.

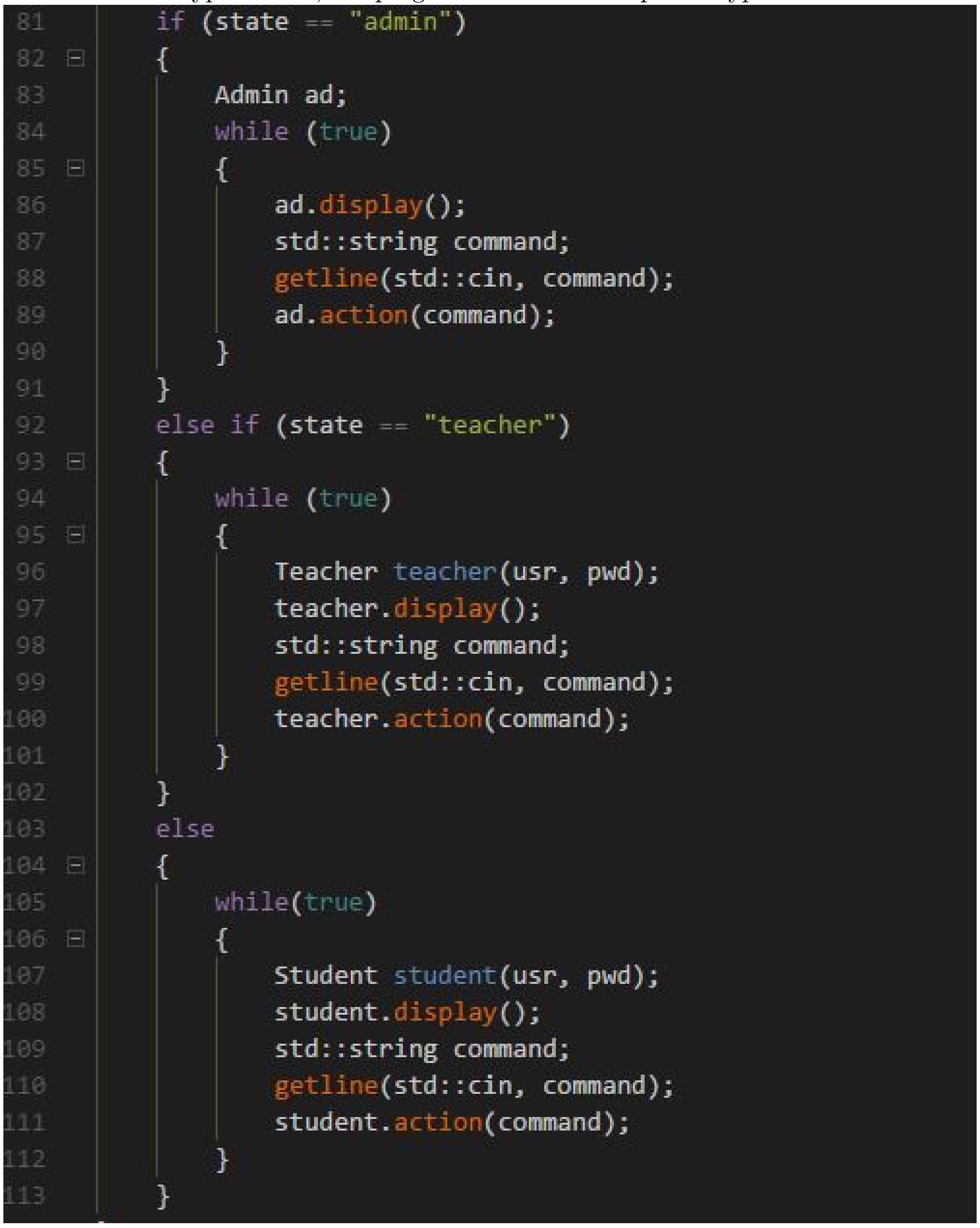
2.2 Login - Logout



```
std::string usr, pwd, person;

std::string state;
do
{
    system("cls");
    state.clear();
    std::vector<std::string> info;
    usr.clear();
    pwd.clear();

    std::cout << "Username: ";
    getline(std::cin, usr);
    std::cout << "Password: ";
    getline(std::cin, pwd);

    std::fstream user(".\\data\\user.csv", std::fstream::in);

    while(!user.eof())
    {
        getline(user, person);
        info.clear();
        split(person, ',', info);

        if ((usr == info[0]) && (pwd == info[1]))
            state = info[2];
    }

    user.close();
} while (state == "");
```

Every time the user try to login, the program will reload the user.csv file which is contain username, password and job title.
If login correctly, the variable **state** will note the type of the user which login the system.

Base on the type of user, the program will direct to specific type of action.



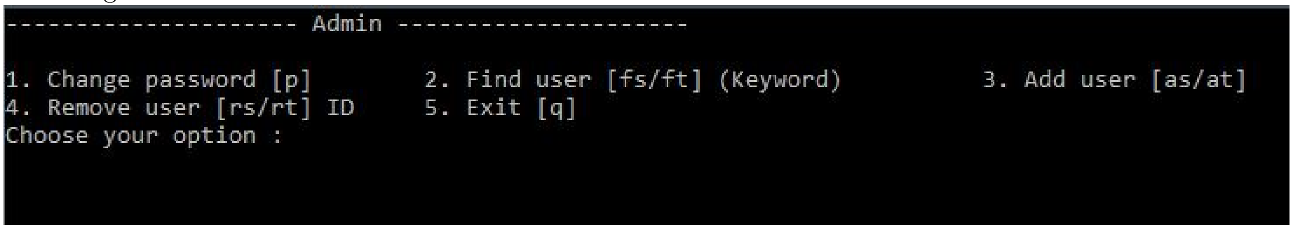
```
    if (state == "admin")
    {
        Admin ad;
        while (true)
        {
            ad.display();
            std::string command;
            getline(std::cin, command);
            ad.action(command);
        }
    }
    else if (state == "teacher")
    {
        while (true)
        {
            Teacher teacher(usr, pwd);
            teacher.display();
            std::string command;
            getline(std::cin, command);
            teacher.action(command);
        }
    }
    else
    {
        while(true)
        {
            Student student(usr, pwd);
            student.display();
            std::string command;
            getline(std::cin, command);
            student.action(command);
        }
    }
```

Step into specific loop of action, program will produce specific object to manipulated action.
Each users type only have 2 public function:

1. **Display** function which will print the menu of action for user.

2. **Action** function will take input argument from user to perform action.

To logout:



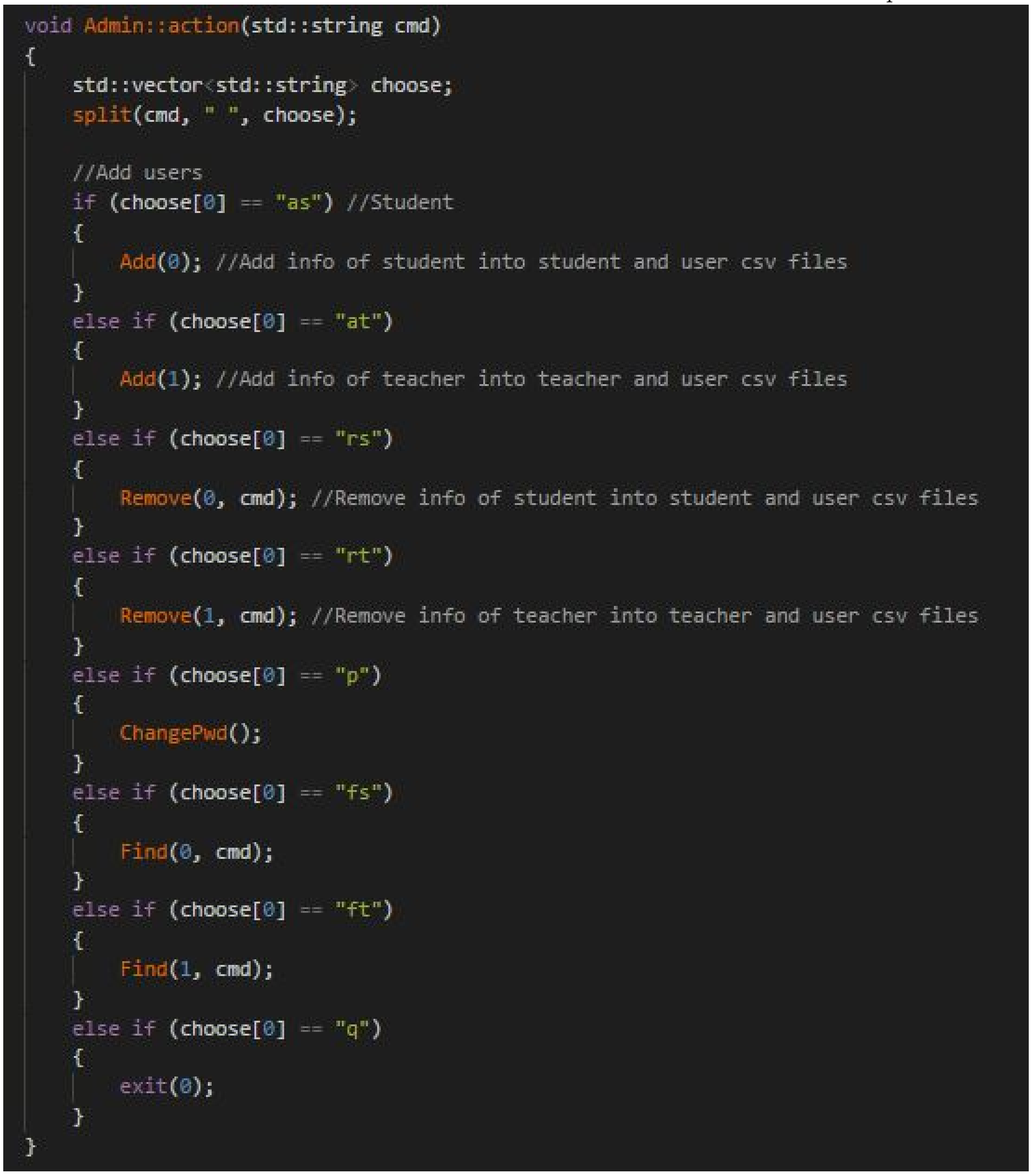
```
-------------------- Admin --------------------

1. Change password [p]       2. Find user [fs/ft] (Keyword)         3. Add user [as/at]
4. Remove user [rs/rt] ID    5. Exit [q]
Choose your option :
```

Base on the action menu, to logout user will type **q**. And the program will shutdown.

2.3 Admin

2.3.1 Action

This function will take in the command from user and direct to correct action to perform.



```
void Admin::action(std::string cmd)
{
    std::vector<std::string> choose;
    split(cmd, " ", choose);

    //Add users
    if (choose[0] == "as") //Student
    {
        Add(0); //Add info of student into student and user csv files
    }
    else if (choose[0] == "at")
    {
        Add(1); //Add info of teacher into teacher and user csv files
    }
    else if (choose[0] == "rs")
    {
        Remove(0, cmd); //Remove info of student into student and user csv files
    }
    else if (choose[0] == "rt")
    {
        Remove(1, cmd); //Remove info of teacher into teacher and user csv files
    }
    else if (choose[0] == "p")
    {
        ChangePwd();
    }
    else if (choose[0] == "fs")
    {
        Find(0, cmd);
    }
    else if (choose[0] == "ft")
    {
        Find(1, cmd);
    }
    else if (choose[0] == "q")
    {
        exit(0);
    }
}
```

This is the main function of Admin object to perform correct action.
In action Add, Remove and Find user, because the target can be teacher or student. So we have 2 type of command for each action.
But the basis of each action is the same, so that 3 functions take 0 or 1 to denote the type of target.

2.3.2 Change Password

Code

```cpp
void Admin::ChangePwd()
{
    std::vector<std::vector<std::string> > data_user;
    std::fstream user_file(".\\data\\user.csv", std::fstream::in);
    readCSV(user_file, data_user);
    std::string confirm, password;

    system("cls");
    std::cout << "Your new password: ";
    std::cin >> password;
    std::cout << "Confirm your password: ";
    std::cin >> confirm;
    if (password == confirm)
    {
        for (int i = 0; i < data_user.size(); i++)
        {
            if (data_user[i][0] == "admin")
            {
                data_user[i][1] = password;
                std::cout << "Change password successfully";

                std::ofstream write_user(".\\data\\user.csv", std::ofstream::out);
                writeCSV(write_user, data_user);
                break;
            }
        }
    }
    else std::cout << "Error: Wrong confirm password.";
    system("pause");
}
```

Open User.csv to read and save data to vector data_user

User must enter the new password two times to confirm.

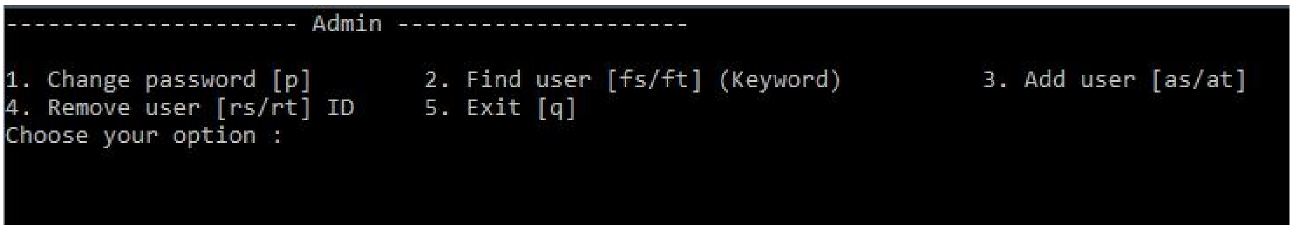
If two passwords are the same, it will change the value of vector where the old password is saved.

Otherwise, it will print out Error and return to the Menu.

How To Use

- When you want to change Admin's password, you enter "p" in the Menu section.
- The console will print out to tell you to enter your new password two times. You should enter it correctly to change the password.
- If you enter the confirm incorrectly, it will print out Error.
- After executing "print out" step (both Success and Error), it will return to selection Menu again.

2.3.3 Find user

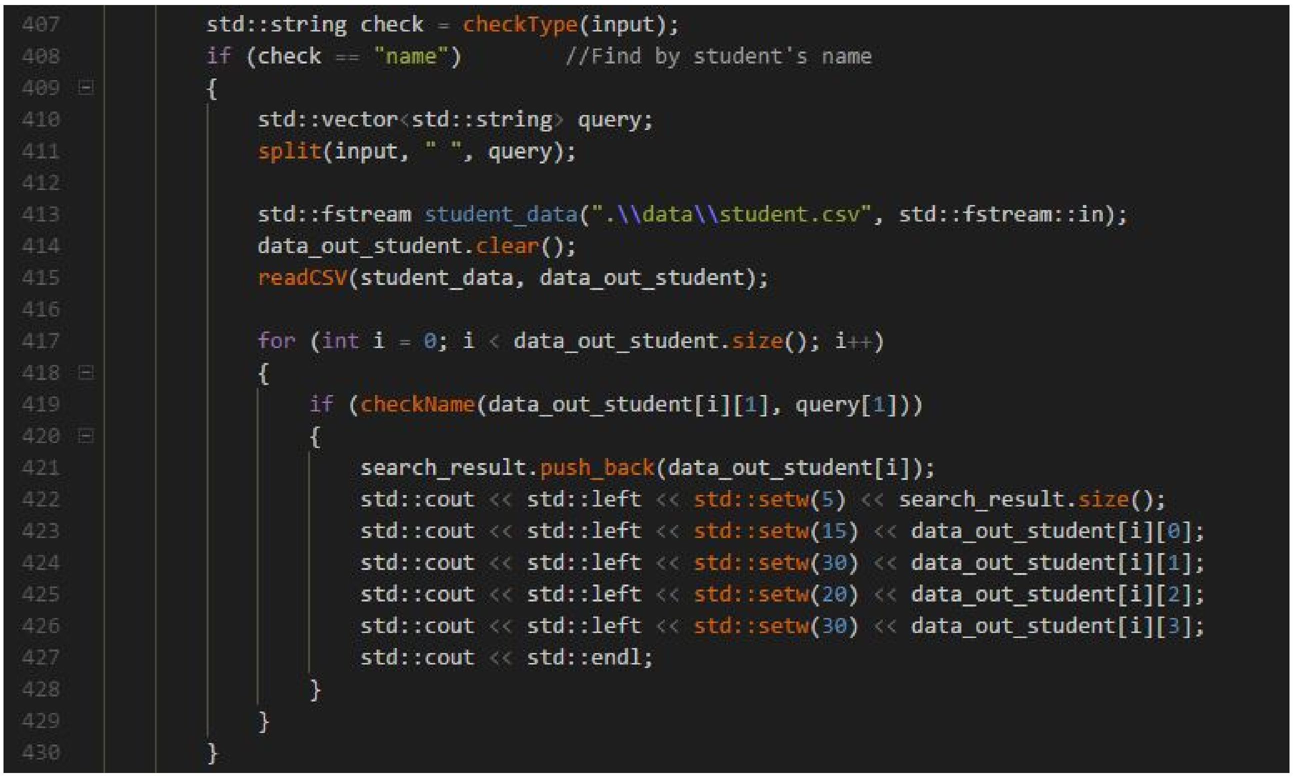


```
--------------------- Admin ---------------------

1. Change password [p]        2. Find user [fs/ft] (Keyword)          3. Add user [as/at]
4. Remove user [rs/rt] ID     5. Exit [q]
Choose your option :
```

If admin want to find student, the admin can find base on student's name or student's address. To use, admin type:

1. **fs NAME**: This command will find all student who have last name, middle name or family name correct to **NAME**.
   ex: **fs HANH**. This will find all student which HANH in their name.

2. **fs "ADDRESS"**: This command will find all student came from **ADDRESS**.
   ex: **fs "ThaiNguyen"**. This will find all student who came from Thai Nguyen.



```
407         std::string check = checkType(input);
408         if (check == "name")        //Find by student's name
409         {
410             std::vector<std::string> query;
411             split(input, " ", query);
412 
413             std::fstream student_data(".\\data\\student.csv", std::fstream::in);
414             data_out_student.clear();
415             readCSV(student_data, data_out_student);
416 
417             for (int i = 0; i < data_out_student.size(); i++)
418             {
419                 if (checkName(data_out_student[i][1], query[1]))
420                 {
421                     search_result.push_back(data_out_student[i]);
422                     std::cout << std::left << std::setw(5) << search_result.size();
423                     std::cout << std::left << std::setw(15) << data_out_student[i][0];
424                     std::cout << std::left << std::setw(30) << data_out_student[i][1];
425                     std::cout << std::left << std::setw(20) << data_out_student[i][2];
426                     std::cout << std::left << std::setw(30) << data_out_student[i][3];
427                     std::cout << std::endl;
428                 }
429             }
430         }
```

Line 407 is the most important line in this function. Function **checkType** return the type of query from user's input.
There are 3 values return from this function:

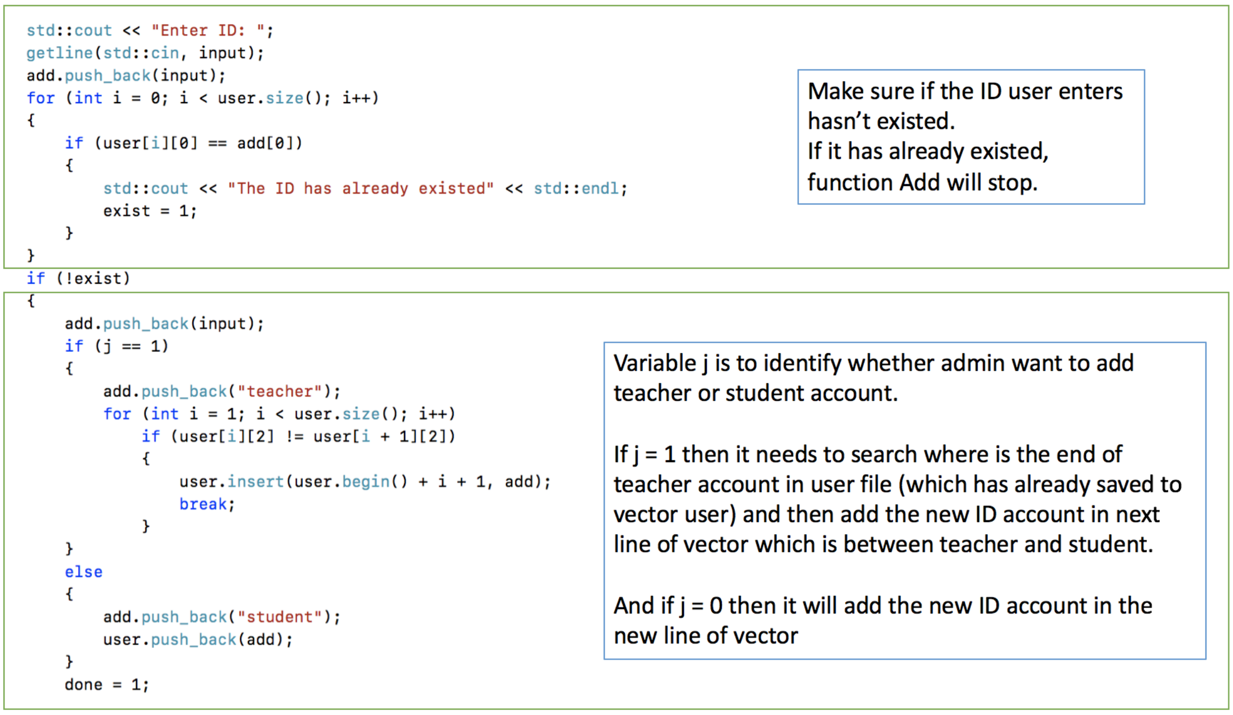
- "**name**".
- "**phone**" (Only in find teacher mode).
- "**address**".

From snippet code above, this part is used to find all student whose name correct from student.csv file.
And print out the console in table form.
Also, store correct student to a 2D vector for after usage (In function Remove).
All others finding mode have the same kind to this one. Note: to find teacher use command **ft** instead.

2.3.4 Add User

Code

At first we need to make sure it is not existed ID.



```
std::cout << "Enter ID: ";
getline(std::cin, input);
add.push_back(input);
for (int i = 0; i < user.size(); i++)
{
    if (user[i][0] == add[0])
    {
        std::cout << "The ID has already existed" << std::endl;
        exist = 1;
    }
}
if (!exist)
{
    add.push_back(input);
    if (j == 1)
    {
        add.push_back("teacher");
        for (int i = 1; i < user.size(); i++)
            if (user[i][2] != user[i + 1][2])
            {
                user.insert(user.begin() + i + 1, add);
                break;
            }
    }
    else
    {
        add.push_back("student");
        user.push_back(add);
    }
    done = 1;
```

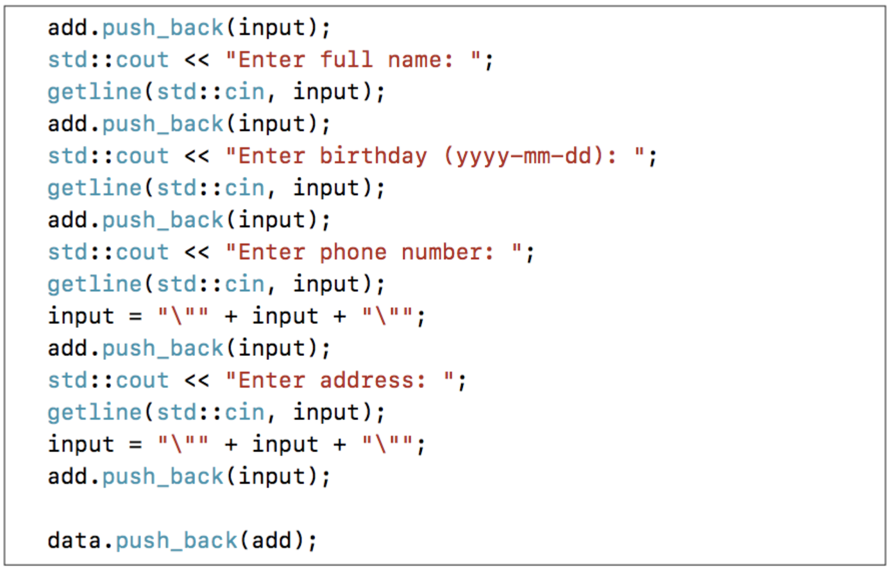
Make sure if the ID user enters hasn't existed.
If it has already existed, function Add will stop.

Variable j is to identify whether admin want to add teacher or student account.

If j = 1 then it needs to search where is the end of teacher account in user file (which has already saved to vector user) and then add the new ID account in next line of vector which is between teacher and student.

And if j = 0 then it will add the new ID account in the new line of vector

For the case when j = 1, we print out to ask the user to enter information (Name, Birthday, Phone number, Address).
After providing enough information, these data will be stored in the last row of vector (which has read and saved all the information from teacher.csv file)
Then the vector, which has already saved new data), will write over the teacher.csv file .



```
add.push_back(input);
std::cout << "Enter full name: ";
getline(std::cin, input);
add.push_back(input);
std::cout << "Enter birthday (yyyy-mm-dd): ";
getline(std::cin, input);
add.push_back(input);
std::cout << "Enter phone number: ";
getline(std::cin, input);
input = "\"" + input + "\"";
add.push_back(input);
std::cout << "Enter address: ";
getline(std::cin, input);
input = "\"" + input + "\"";
add.push_back(input);

data.push_back(add);
```

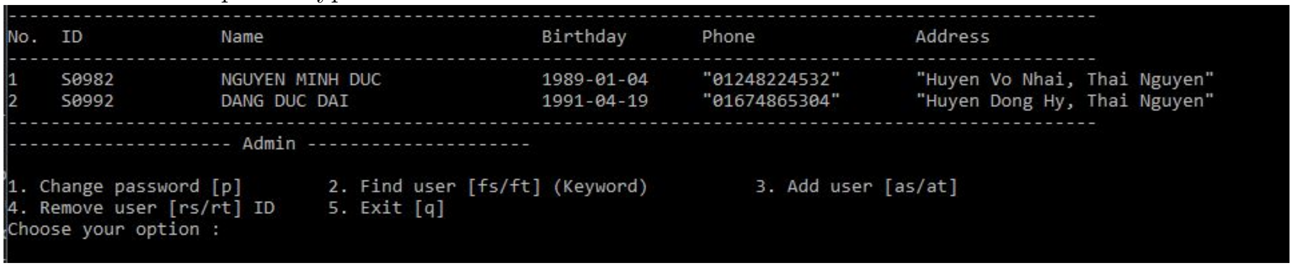
For the case when j = 0, it will be mostly the same as when j = 0, except it will work with student.csv file and user only needs to enter 3 information (Name, Birthday, Adress).

```
add.push_back(input);
std::cout << "Enter full name: ";
getline(std::cin, input);
add.push_back(input);
std::cout << "Enter birthday (yyyy-mm-dd): ";
getline(std::cin, input);
add.push_back(input);
std::cout << "Enter address: ";
getline(std::cin, input);
input = "\"" + input + "\"";
add.push_back(input);
data.push_back(add);
```

How To Use
- If you want to add a student account, enter ”as” in the selection Menu. Or if you want to add a teacher account, enter ”at”.
- The console will print out which information you need to provide.
(For the birthday information, please enter in form of yyyy-mm-dd, for example : 1999-08-18.)
- If you enter an existed ID, console will tell you and return to the selection Menu.

2.3.5 Remove user

After finding user, admin can remove user by No. in result form or by user’s ID.
This result come up after type: **ft DUC**.

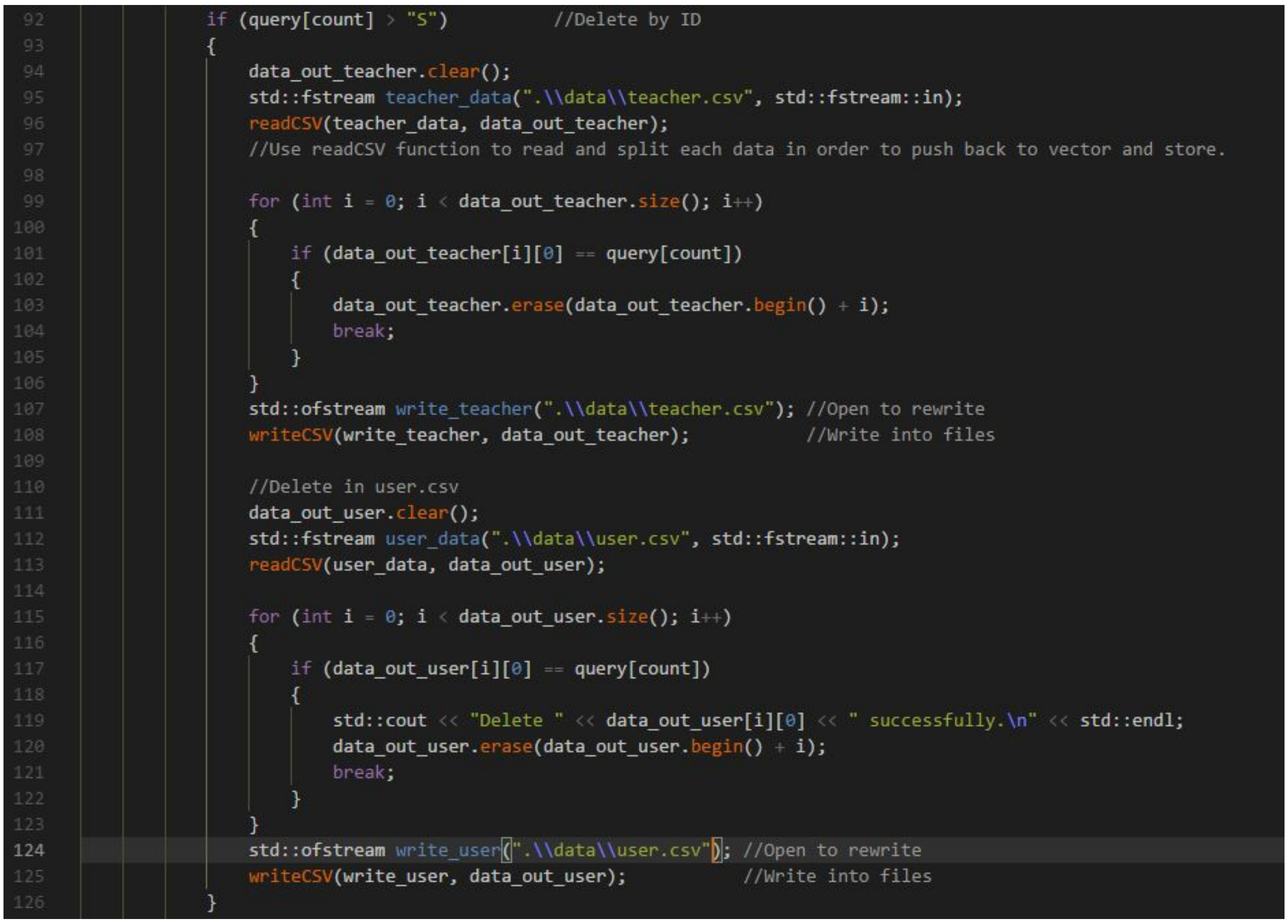


```
----------------------------------------------------------------------------------------------------------
No.  ID             Name                          Birthday       Phone                Address
----------------------------------------------------------------------------------------------------------
1    S0982          NGUYEN MINH DUC               1989-01-04     "01248224532"        "Huyen Vo Nhai, Thai Nguyen"
2    S0992          DANG DUC DAI                  1991-04-19     "01674865304"        "Huyen Dong Hy, Thai Nguyen"
----------------------------------------------------------------------------------------------------------
-------------------- Admin --------------------

1. Change password [p]       2. Find user [fs/ft] (Keyword)        3. Add user [as/at]
4. Remove user [rs/rt] ID    5. Exit [q]
Choose your option :
```

To remove, for example type:

- **rt 1**: This will remove ”NGUYEN MINH DUC” from database.
- **rt S0992**: This will remove ”DANG DUC DAI” from database.
- **rt 1 S0992**: This will remove 2 teachers from database. Can use to remove multiple user.
- Change **rt** to **rs** to remove student.

NOTE : After find teacher and want to remove, use correct command **rt**. And do the same with student.



```
            if (query[count] > "S")          //Delete by ID
            {
                data_out_teacher.clear();
                std::fstream teacher_data(".\\data\\teacher.csv", std::fstream::in);
                readCSV(teacher_data, data_out_teacher);
                //Use readCSV function to read and split each data in order to push back to vector and store.

                for (int i = 0; i < data_out_teacher.size(); i++)
                {
                    if (data_out_teacher[i][0] == query[count])
                    {
                        data_out_teacher.erase(data_out_teacher.begin() + i);
                        break;
                    }
                }
                std::ofstream write_teacher(".\\data\\teacher.csv"); //Open to rewrite
                writeCSV(write_teacher, data_out_teacher);           //Write into files

                //Delete in user.csv
                data_out_user.clear();
                std::fstream user_data(".\\data\\user.csv", std::fstream::in);
                readCSV(user_data, data_out_user);

                for (int i = 0; i < data_out_user.size(); i++)
                {
                    if (data_out_user[i][0] == query[count])
                    {
                        std::cout << "Delete " << data_out_user[i][0] << " successfully.\n" << std::endl;
                        data_out_user.erase(data_out_user.begin() + i);
                        break;
                    }
                }
                std::ofstream write_user(".\\data\\user.csv"); //Open to rewrite
                writeCSV(write_user, data_out_user);           //Write into files
            }
```

Each removed user, will be erased from user.csv and teacher.csv (student.csv for student).
To do this action on teacher for example, the function must go through these steps :

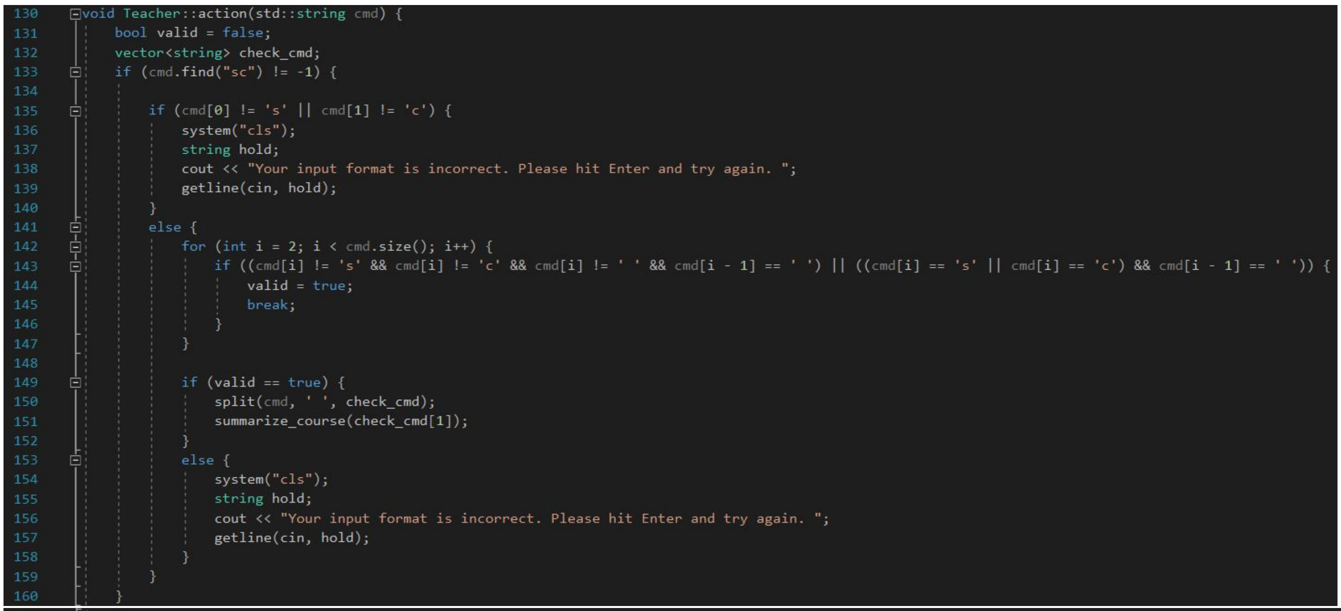
- Read all teacher data in teacher.csv and store.
- Iterate through the vector to find the teacher who has the same ID.
- Delete that entire row in vector and break the loop.
- Write that vector to teacher.csv.

These steps do the same in student.csv and user.csv.

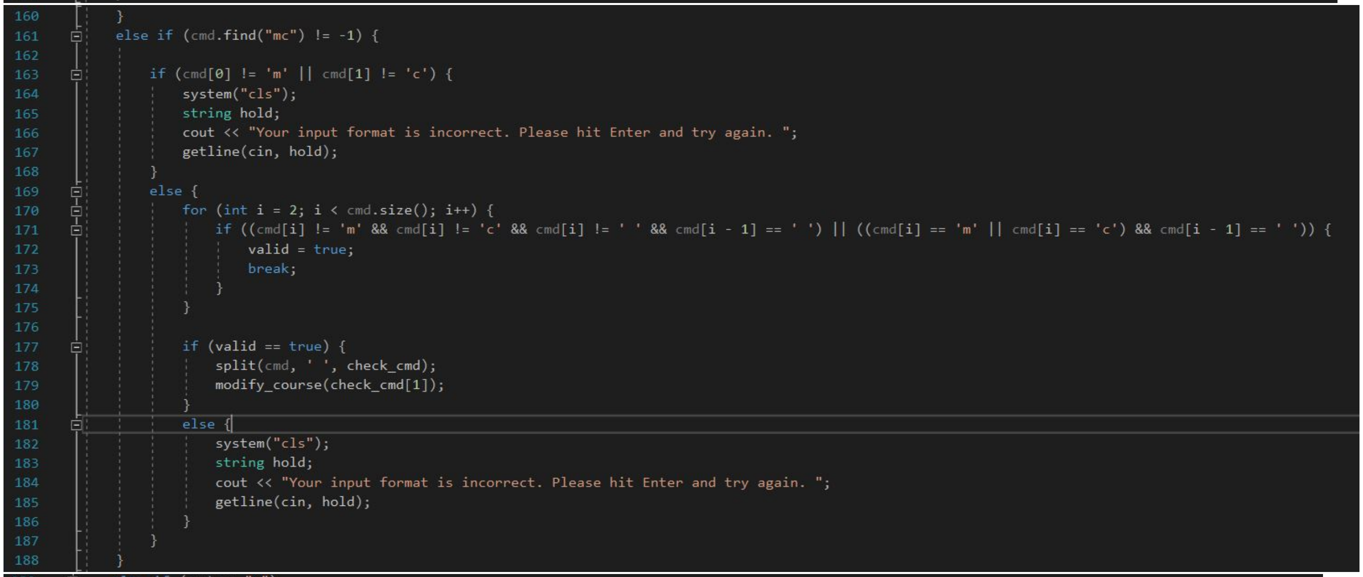
2.4 Professor

2.4.1 Action

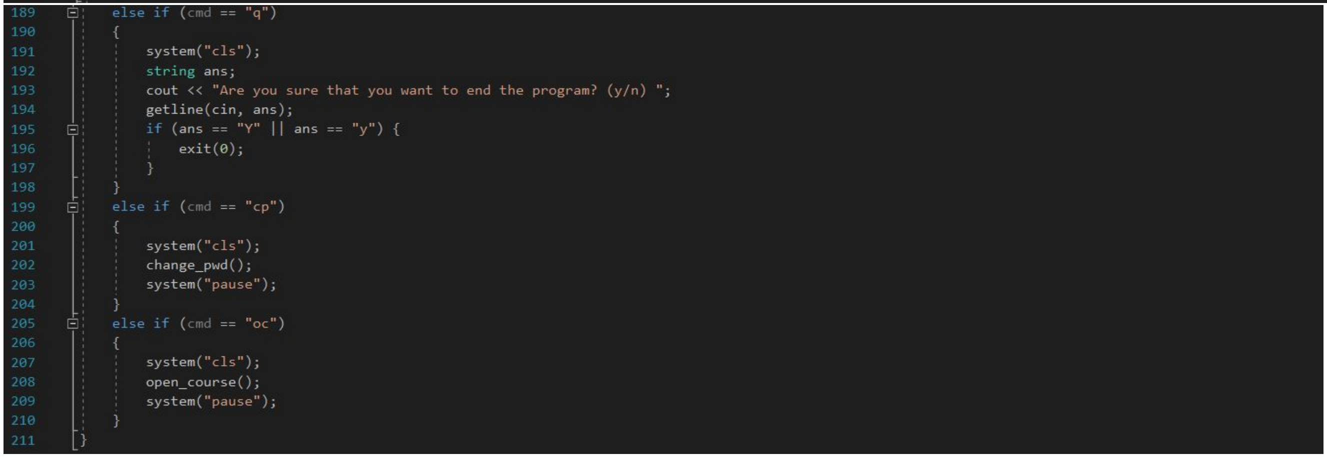
Source code:



```cpp
void Teacher::action(std::string cmd) {
    bool valid = false;
    vector<string> check_cmd;
    if (cmd.find("sc") != -1) {

        if (cmd[0] != 's' || cmd[1] != 'c') {
            system("cls");
            string hold;
            cout << "Your input format is incorrect. Please hit Enter and try again. ";
            getline(cin, hold);
        }
        else {
            for (int i = 2; i < cmd.size(); i++) {
                if ((cmd[i] != 's' && cmd[i] != 'c' && cmd[i] != ' ' && cmd[i - 1] == ' ') || ((cmd[i] == 's' || cmd[i] == 'c') && cmd[i - 1] == ' ')) {
                    valid = true;
                    break;
                }
            }

            if (valid == true) {
                split(cmd, ' ', check_cmd);
                summarize_course(check_cmd[1]);
            }
            else {
                system("cls");
                string hold;
                cout << "Your input format is incorrect. Please hit Enter and try again. ";
                getline(cin, hold);
            }
        }
    }
```



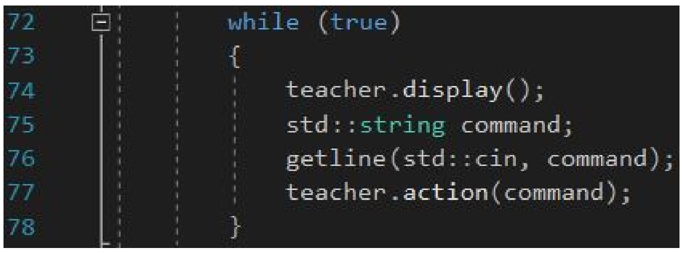
```cpp
    }
    else if (cmd.find("mc") != -1) {

        if (cmd[0] != 'm' || cmd[1] != 'c') {
            system("cls");
            string hold;
            cout << "Your input format is incorrect. Please hit Enter and try again. ";
            getline(cin, hold);
        }
        else {
            for (int i = 2; i < cmd.size(); i++) {
                if ((cmd[i] != 'm' && cmd[i] != 'c' && cmd[i] != ' ' && cmd[i - 1] == ' ') || ((cmd[i] == 'm' || cmd[i] == 'c') && cmd[i - 1] == ' ')) {
                    valid = true;
                    break;
                }
            }

            if (valid == true) {
                split(cmd, ' ', check_cmd);
                modify_course(check_cmd[1]);
            }
            else {
                system("cls");
                string hold;
                cout << "Your input format is incorrect. Please hit Enter and try again. ";
                getline(cin, hold);
            }
        }
    }
```



```cpp
    else if (cmd == "q")
    {
        system("cls");
        string ans;
        cout << "Are you sure that you want to end the program? (y/n) ";
        getline(cin, ans);
        if (ans == "Y" || ans == "y") {
            exit(0);
        }
    }
    else if (cmd == "cp")
    {
        system("cls");
        change_pwd();
        system("pause");
    }
    else if (cmd == "oc")
    {
        system("cls");
        open_course();
        system("pause");
    }
}
```

Documentation: • The function contains five different branches of conditions. If one out of the five conditions is satisfied, which means the user's input matches the syntax of one of the commands, the statements within the block of that condition will be executed ⇒ the desired functionality is activated. The other dissatisfied conditions and their blocks will be ignored.

• If none of the conditions is satisfied, which means the user's input does not match any command syntaxes.



```
while (true)
{
    teacher.display();
    std::string command;
    getline(std::cin, command);
    teacher.action(command);
}
```

The "action" function call will end. The "display" function will be called and the user will be prompted to input the command again because everything is placed within a **while(true)** loop. By this way, the appropriate functionality will only be activated when the user inputs a command with correct syntax.

• The "sc [Course code]" command (line 133 to 160):

+ Basing on the first condition, if the string "sc" is found within the inputted command (string object "cmd"), the block of this **if** statement will be executed.

+ Types of input syntax errors the user can create:

1. One or multiple white-space characters ' ' before the string "sc".
   For example: "   sc MT1005"
2. The string "sc" is not placed at the front.
   For example: "MT1005 sc".
3. Only the string "sc" **without** any [Course code].
   For example: "sc".
4. Only the string "sc" and one or multiple white-space characters ' ' behind it, **without** any [Course code].
   For example: "sc   ".
5. Missing the white-space character(s) (one or many are both accepted) between the string "sc" and [Course code].
   For example: "scMT1005"

+ **To resolve error number 1 and 2**, we have the next **if** statement with the condition "(cmd[0] != 's' || cmd[1] != 'c')". If the first and second letters of "cmd" are not respectively 's' and 'c', the screen will be cleared and a warning ("Your input format is incorrect. Please hit Enter and try again. ") will be showed.

The purpose of the **getline** function is only to wait for the user to hit Enter.

When the user hits Enter, the "action" function call will end. The "display" function will be called and the user will be prompted to input the command again.

+ **To resolve error number 3, 4 and 5**, we have the next **else** statement.
The **for** loop scans the whole string "cmd". This loop starts at the third character (index i = 2) of "cmd" since the first and the second characters are now surely 's' and 'c'. If there exists a character within "cmd" that is: different from 's', 'c' and the character to its left is a whitespace ' ' **OR** the same as 's' or 'c' and the character to its left is a whitespace ' ', the **bool** variable "valid" will turn **true** and we immediately break out of the loop.

In the next **if** statement, we check "valid". If "valid" is still **false**, which means one of the three errors 3, 4 or 5 has been triggered, we skip the **if** statement and jump into **else** to show the user the warning and return to the command menu. Otherwise, if "valid" has turned **true**, the function "split" will be called to split up the string "cmd" by the whitespace(s) and **push_back** the string "sc" and the [Course code] into the string vector "check_cmd". [Course code] will be the second element of this vector and it will be passed to the function "summarize_course(string code)".

- The "mc [Course code]" command (line 161 to 188):

⇒ Exactly the same as the procedure we have taken for the command "sc [Course code]". This time, however, instead of having the letter 's' in the "cmd" string and calling the function "summarize_course(string code)" when no input error is triggered, we have the letter 'm' replacing the position of 's'; and when no input error is triggered, the function "modify_course(string code)" will be called.

- The "q" command (line 189 to 198):

  + When the user inputs "q", only this block of code will be executed, the rest will be ignored.

  + The screen is cleared and the following question is printed to the console. "Are you sure that you want to end the program? (y/n) "

  + The user is prompted to input his/her answer and this answer will be stored in the string object "ans". If "ans" is anything other than "Y" or "y", the "action" function call will end. The "display" function will be called and the user will be prompted to input the command again. Otherwise, if "ans" is "Y" or "y", the condition of the **if** statement is evaluated to be **true** and the program will end immediately because of **exit(0);**.

- The "cp" command (line 199 to 204):

  + When the user inputs "cp", only this block of code will be executed, the rest will be ignored.

  + The screen is cleared and the "change_pwd()" function is called to execute the changing password functionality of the program.

  + When the "change_pwd()" function call ends, the "display" function will be called and the user will be prompted to input the command again.

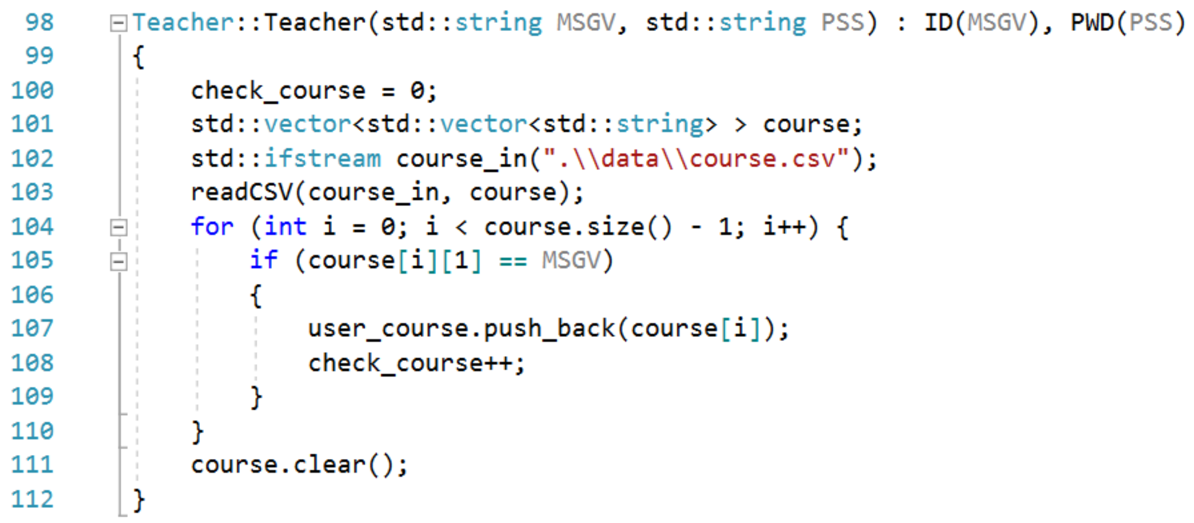
- The "oc" command (line 205 to 210):

  + When the user inputs "oc", only this block of code will be executed, the rest will be ignored.

  + The screen is cleared and the "open_course()" function is called to execute the opening course functionality of the program.

  + When the "open_course()" function call ends, the "display" function will be called and the user will be prompted to input the command again.

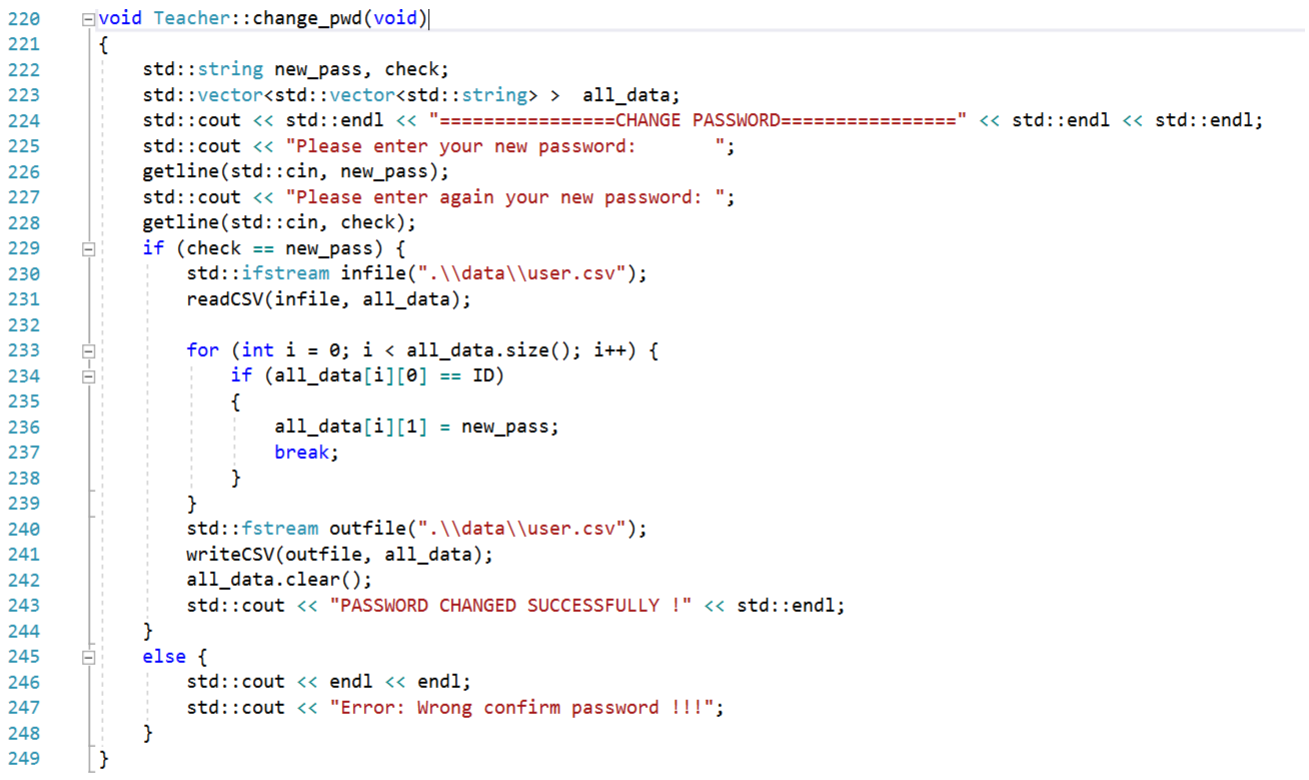
2.4.2 Constructor



```
98   Teacher::Teacher(std::string MSGV, std::string PSS) : ID(MSGV), PWD(PSS)
99   {
100      check_course = 0;
101      std::vector<std::vector<std::string> > course;
102      std::ifstream course_in(".\\data\\course.csv");
103      readCSV(course_in, course);
104      for (int i = 0; i < course.size() - 1; i++) {
105          if (course[i][1] == MSGV)
106          {
107              user_course.push_back(course[i]);
108              check_course++;
109          }
110      }
111      course.clear();
112  }
```

-(98) :
MSGV and PSS are assigned respectively to private variables ID and PWD.
-(100) :
Set private variable "check_course" is 0. This variable is the number of course which teacher has in the course.csv file.
-(101) → (110) :
First, we open the course.csv file and read it through "readCSV" function. Then, in the for loop, we find the course of the login teacher and save it to 2D vector "user_course" (declared in private of Teacher class). Beside that, the "check_course" variable also count up when the function find out the courses' teacher.
-(111) :
Clear 2D vector "course" that is declared in line (101).

2.4.3 Change Password



```
220  void Teacher::change_pwd(void)
221  {
222      std::string new_pass, check;
223      std::vector<std::vector<std::string> >  all_data;
224      std::cout << std::endl << "================CHANGE PASSWORD================" << std::endl << std::endl;
225      std::cout << "Please enter your new password:       ";
226      getline(std::cin, new_pass);
227      std::cout << "Please enter again your new password: ";
228      getline(std::cin, check);
229      if (check == new_pass) {
230          std::ifstream infile(".\\data\\user.csv");
231          readCSV(infile, all_data);
232
233          for (int i = 0; i < all_data.size(); i++) {
234              if (all_data[i][0] == ID)
235              {
236                  all_data[i][1] = new_pass;
237                  break;
238              }
239          }
240          std::fstream outfile(".\\data\\user.csv");
241          writeCSV(outfile, all_data);
242          all_data.clear();
243          std::cout << "PASSWORD CHANGED SUCCESSFULLY !" << std::endl;
244      }
245      else {
246          std::cout << endl << endl;
247          std::cout << "Error: Wrong confirm password !!!";
248      }
249  }
```

-(222) → (223):
Declaring variables: string "new_pass" and "check", 2D vector "all_data".
-(224) → (228):
Ask the user to enter and confirm new password and store them in two string variables: "new_pass" and "check".
-(229) → (244):

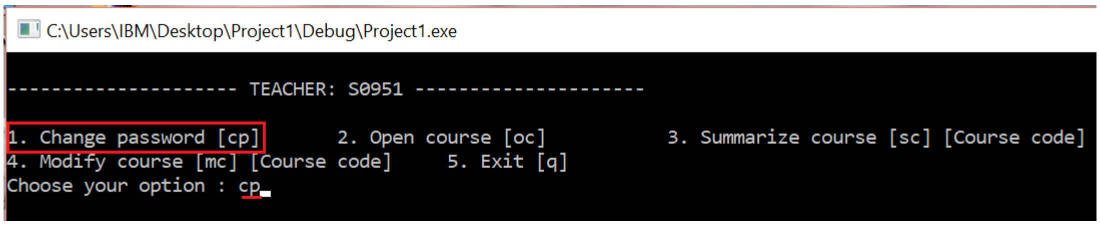
Here, the program will check if the user confirm the right new password or not. If it is right, the "user.csv" file will be opened and read. Then, in the for loop, the function will find ID of the teacher to assign the new password to the 2D vector. After that, we open the file "user.csv" again to write the changed 2D vector to it and print out the screen: "PASSWORD CHANGED SUCCESSFULLY !".

-(245) → (248):

If the confirm password is different from the new password, the screen will show: "Error: Wrong confirm password !!!".

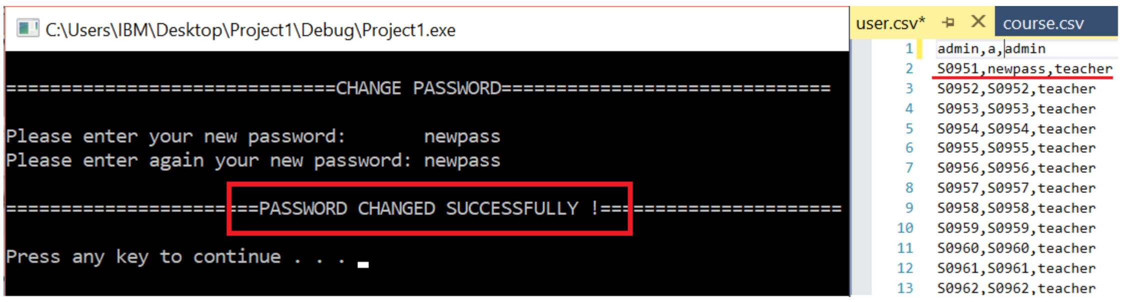
- **Console Screen:**

1.After login, to change password, you have to type "cp":



```
C:\Users\IBM\Desktop\Project1\Debug\Project1.exe

-------------------- TEACHER: S0951 --------------------

1. Change password [cp]       2. Open course [oc]          3. Summarize course [sc] [Course code]
4. Modify course [mc] [Course code]     5. Exit [q]
Choose your option : cp
```

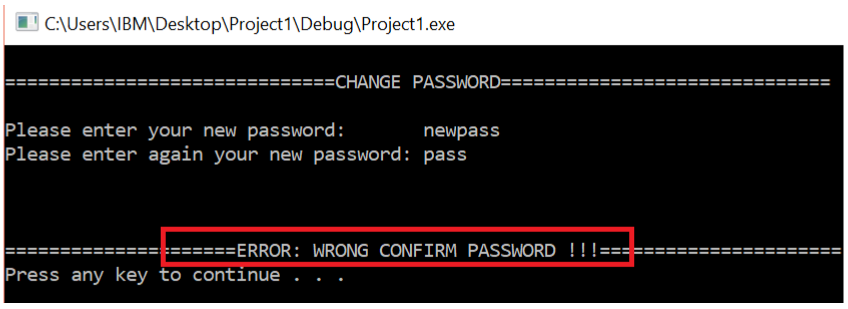
2.Respectively type your new password and cofirm it right:



```
C:\Users\IBM\Desktop\Project1\Debug\Project1.exe

=============================CHANGE PASSWORD=============================

Please enter your new password:       newpass
Please enter again your new password: newpass

===================PASSWORD CHANGED SUCCESSFULLY !===================

Press any key to continue . . .
```

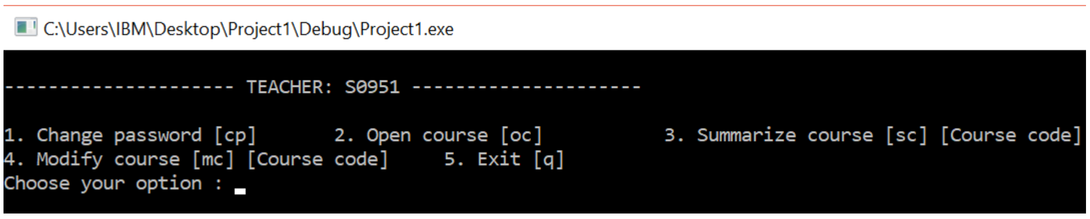
```
user.csv*    course.csv
admin,a,admin
S0951,newpass,teacher
S0952,S0952,teacher
S0953,S0953,teacher
S0954,S0954,teacher
S0955,S0955,teacher
S0956,S0956,teacher
S0957,S0957,teacher
S0958,S0958,teacher
S0959,S0959,teacher
S0960,S0960,teacher
S0961,S0961,teacher
S0962,S0962,teacher
```

3.Respectively type your new password and cofirm it wrong:



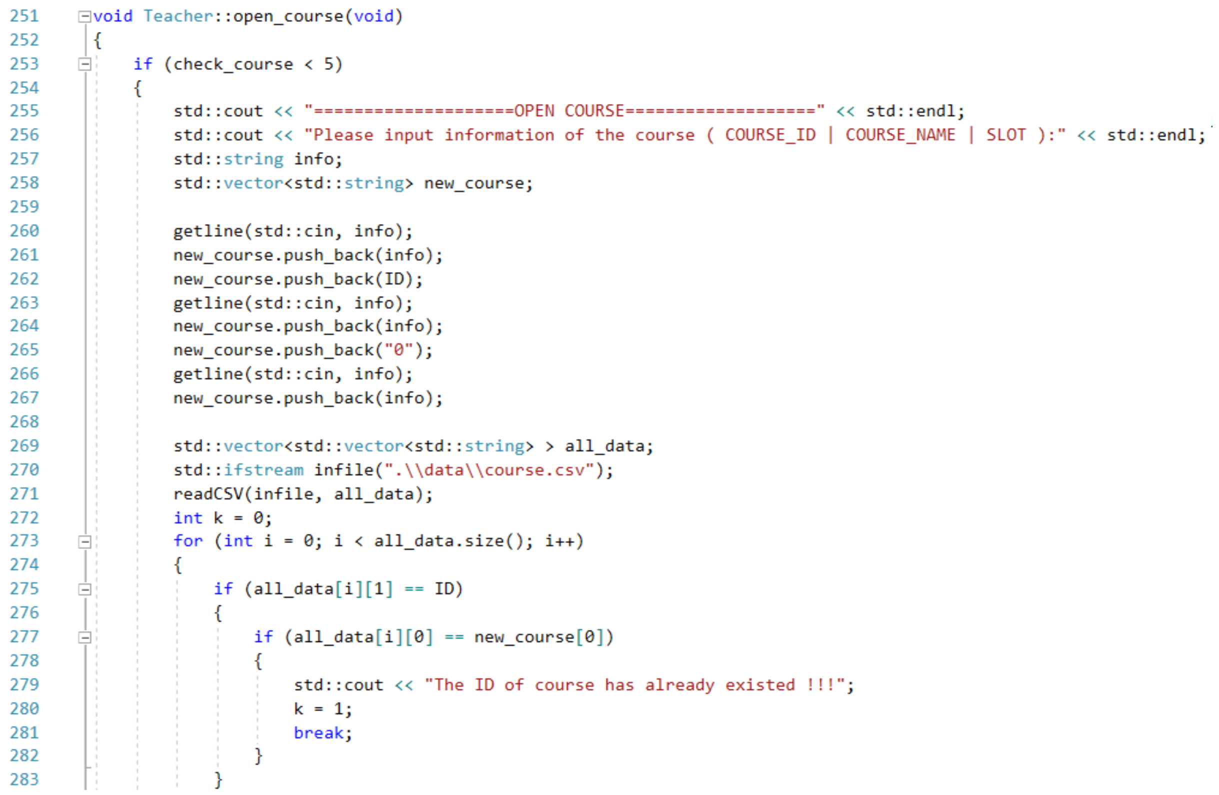
```
C:\Users\IBM\Desktop\Project1\Debug\Project1.exe

=============================CHANGE PASSWORD=============================

Please enter your new password:       newpass
Please enter again your new password: pass


===================ERROR: WRONG CONFIRM PASSWORD !!!===================
Press any key to continue . . .
```

5.Press any key to come back the option:



```
C:\Users\IBM\Desktop\Project1\Debug\Project1.exe

-------------------- TEACHER: S0951 --------------------

1. Change password [cp]       2. Open course [oc]          3. Summarize course [sc] [Course code]
4. Modify course [mc] [Course code]     5. Exit [q]
Choose your option :
```

2.4.4 Open Course



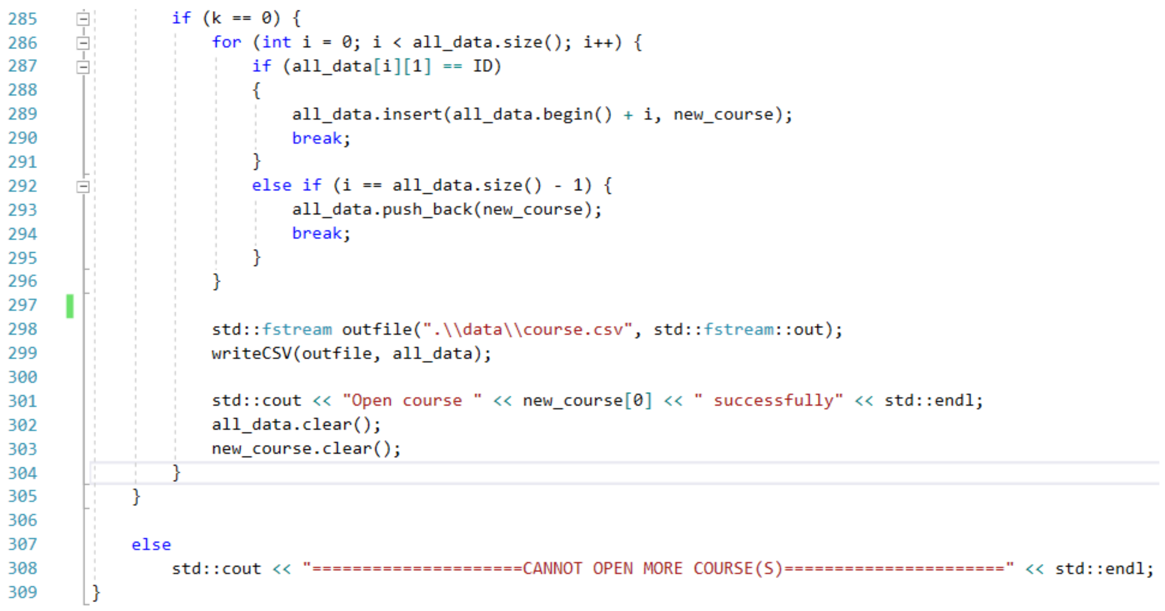
```
251  void Teacher::open_course(void)
252  {
253      if (check_course < 5)
254      {
255          std::cout << "===================OPEN COURSE==================" << std::endl;
256          std::cout << "Please input information of the course ( COURSE_ID | COURSE_NAME | SLOT ):" << std::endl;
257          std::string info;
258          std::vector<std::string> new_course;
259  
260          getline(std::cin, info);
261          new_course.push_back(info);
262          new_course.push_back(ID);
263          getline(std::cin, info);
264          new_course.push_back(info);
265          new_course.push_back("0");
266          getline(std::cin, info);
267          new_course.push_back(info);
268  
269          std::vector<std::vector<std::string> > all_data;
270          std::ifstream infile(".\\data\\course.csv");
271          readCSV(infile, all_data);
272          int k = 0;
273          for (int i = 0; i < all_data.size(); i++)
274          {
275              if (all_data[i][1] == ID)
276              {
277                  if (all_data[i][0] == new_course[0])
278                  {
279                      std::cout << "The ID of course has already existed !!!";
280                      k = 1;
281                      break;
282                  }
283              }
```

-(253) → (267):

First, the fuction will check if the teacher has already had 5 course or not. If not, this teacher can open more course so the screen will print out the form of open course and guide teacher to input information. Then, function will read and push back all the information into 1D vector "new_course" (line : (259) → (267)).

-(269) → (283):

File "course.csv" will be opened and read by "readCSV" function. Then, variable "k" = 0 is declared to check the duplication of the ID of the course. In the first for loop, we will find the teacher who has already had courses, and then we will check the duplication of ID of courses. Whenever, the new ID of the course is duplicated, the screen will print out error and now the variable "k" = 1.



```
285          if (k == 0) {
286              for (int i = 0; i < all_data.size(); i++) {
287                  if (all_data[i][1] == ID)
288                  {
289                      all_data.insert(all_data.begin() + i, new_course);
290                      break;
291                  }
292                  else if (i == all_data.size() - 1) {
293                      all_data.push_back(new_course);
294                      break;
295                  }
296              }
297  
298              std::fstream outfile(".\\data\\course.csv", std::fstream::out);
299              writeCSV(outfile, all_data);
300  
301              std::cout << "Open course " << new_course[0] << " successfully" << std::endl;
302              all_data.clear();
303              new_course.clear();
304          }
305      }
306  
307      else
308          std::cout << "====================CANNOT OPEN MORE COURSE(S)=====================" << std::endl;
309  }
```

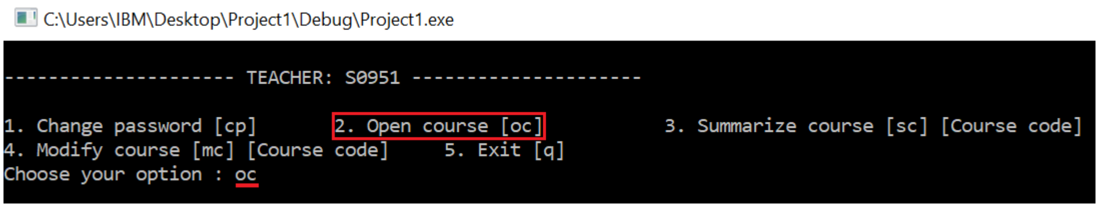
-(285) → (296):

Here, the situation: the ID of the course is not duplicated (k=0). In this second for loop, we will find the first position of ID of teacher to insert the new course if this teacher has already had course(s). If not, the new course will be push back to the end of the 2D vector "all_data". Then, we will write 2D vector to the "course.csv" file by "writeCSV" function. Finally, print out the screen the success and clear all the vector.

-(307) and (308):
If the teacher has equal or more than five courses, he/she cannot open any more: print out error.

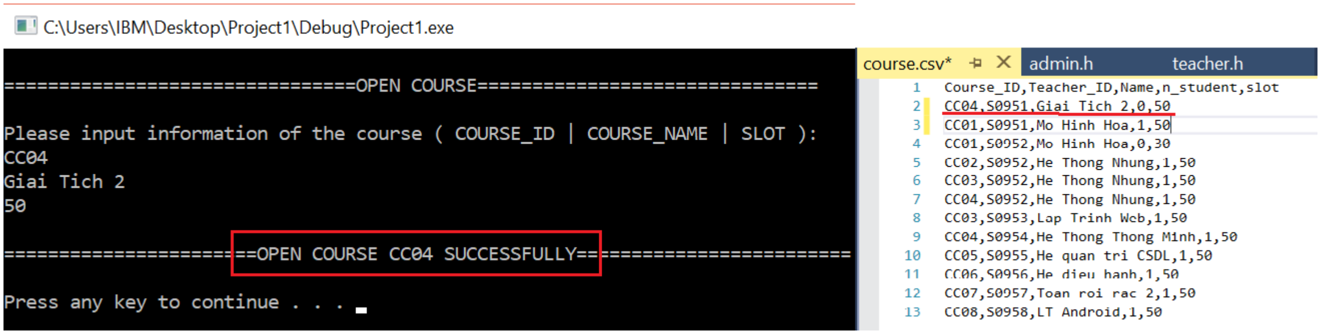
- **Console Screen:**

1.After login, to open course, you have to type "oc":



```
C:\Users\IBM\Desktop\Project1\Debug\Project1.exe

-------------------- TEACHER: S0951 --------------------

1. Change password [cp]        2. Open course [oc]          3. Summarize course [sc] [Course code]
4. Modify course [mc] [Course code]      5. Exit [q]
Choose your option : oc
```

2.Respectively type ID course, Name course and Slot. Success:

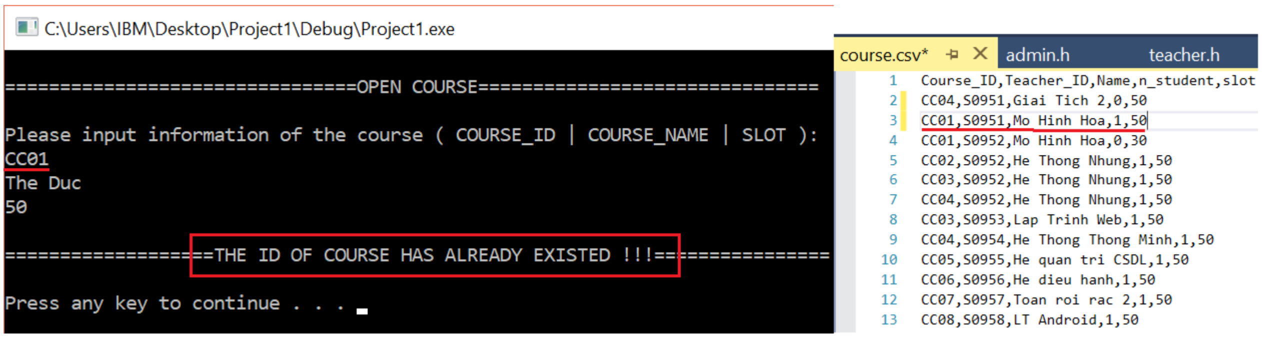


```
C:\Users\IBM\Desktop\Project1\Debug\Project1.exe

================================OPEN COURSE================================

Please input information of the course ( COURSE_ID | COURSE_NAME | SLOT ):
CC04
Giai Tich 2
50

====================OPEN COURSE CC04 SUCCESSFULLY======================

Press any key to continue . . .
```

course.csv* admin.h teacher.h

```
Course_ID,Teacher_ID,Name,n_student,slot
CC04,S0951,Giai Tich 2,0,50
CC01,S0951,Mo Hinh Hoa,1,50
CC01,S0952,Mo Hinh Hoa,0,30
CC02,S0952,He Thong Nhung,1,50
CC03,S0952,He Thong Nhung,1,50
CC04,S0952,He Thong Nhung,1,50
CC03,S0953,Lap Trinh Web,1,50
CC04,S0954,He Thong Thong Minh,1,50
CC05,S0955,He quan tri CSDL,1,50
CC06,S0956,He dieu hanh,1,50
CC07,S0957,Toan roi rac 2,1,50
CC08,S0958,LT Android,1,50
```

3.Respectively type ID course, Name course and Slot. Duplication ID course:

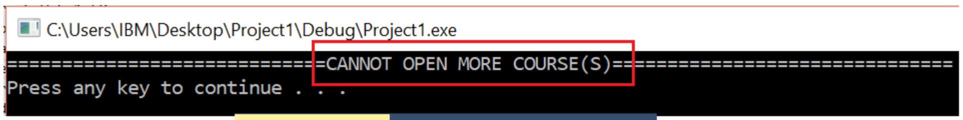


```
C:\Users\IBM\Desktop\Project1\Debug\Project1.exe

================================OPEN COURSE================================

Please input information of the course ( COURSE_ID | COURSE_NAME | SLOT ):
CC01
The Duc
50

=================THE ID OF COURSE HAS ALREADY EXISTED !!!===============

Press any key to continue . . .
```

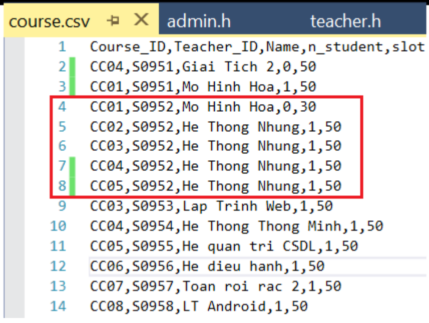
course.csv* admin.h teacher.h

```
Course_ID,Teacher_ID,Name,n_student,slot
CC04,S0951,Giai Tich 2,0,50
CC01,S0951,Mo Hinh Hoa,1,50
CC01,S0952,Mo Hinh Hoa,0,30
CC02,S0952,He Thong Nhung,1,50
CC03,S0952,He Thong Nhung,1,50
CC04,S0952,He Thong Nhung,1,50
CC03,S0953,Lap Trinh Web,1,50
CC04,S0954,He Thong Thong Minh,1,50
CC05,S0955,He quan tri CSDL,1,50
CC06,S0956,He dieu hanh,1,50
CC07,S0957,Toan roi rac 2,1,50
CC08,S0958,LT Android,1,50
```

4.Teacher who has equal or more than 5 courses:

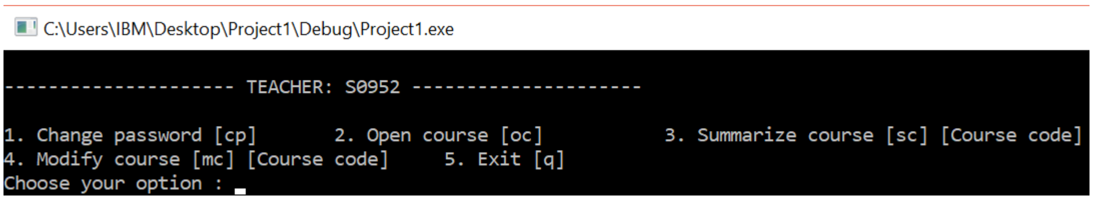


```
C:\Users\IBM\Desktop\Project1\Debug\Project1.exe
=============================CANNOT OPEN MORE COURSE(S)==============================
Press any key to continue . . .
```



```
course.csv   admin.h   teacher.h
Course_ID,Teacher_ID,Name,n_student,slot
CC04,S0951,Giai Tich 2,0,50
CC01,S0951,Mo Hinh Hoa,1,50
CC01,S0952,Mo Hinh Hoa,0,30
CC02,S0952,He Thong Nhung,1,50
CC03,S0952,He Thong Nhung,1,50
CC04,S0952,He Thong Nhung,1,50
CC05,S0952,He Thong Nhung,1,50
CC03,S0953,Lap Trinh Web,1,50
CC04,S0954,He Thong Thong Minh,1,50
CC05,S0955,He quan tri CSDL,1,50
CC06,S0956,He dieu hanh,1,50
CC07,S0957,Toan roi rac 2,1,50
CC08,S0958,LT Android,1,50
```

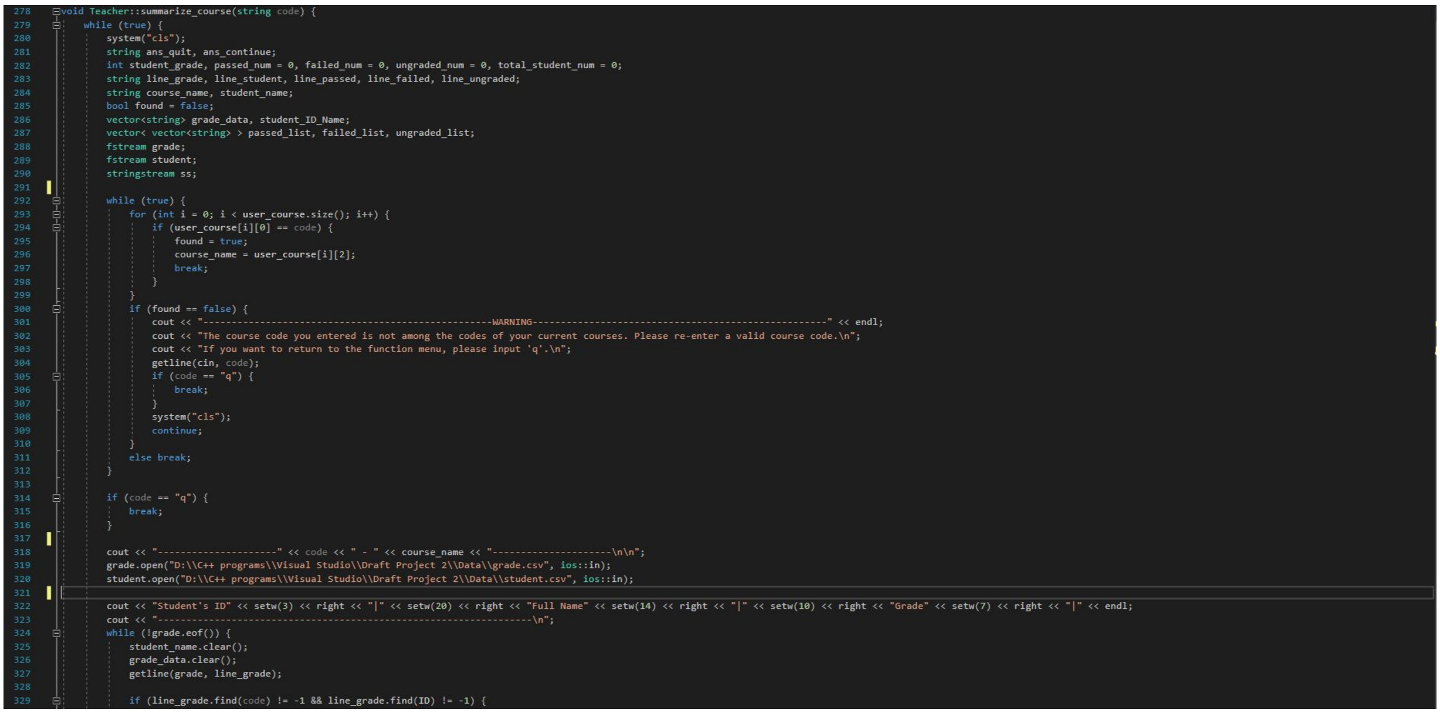
5.Press any key to come back the option:



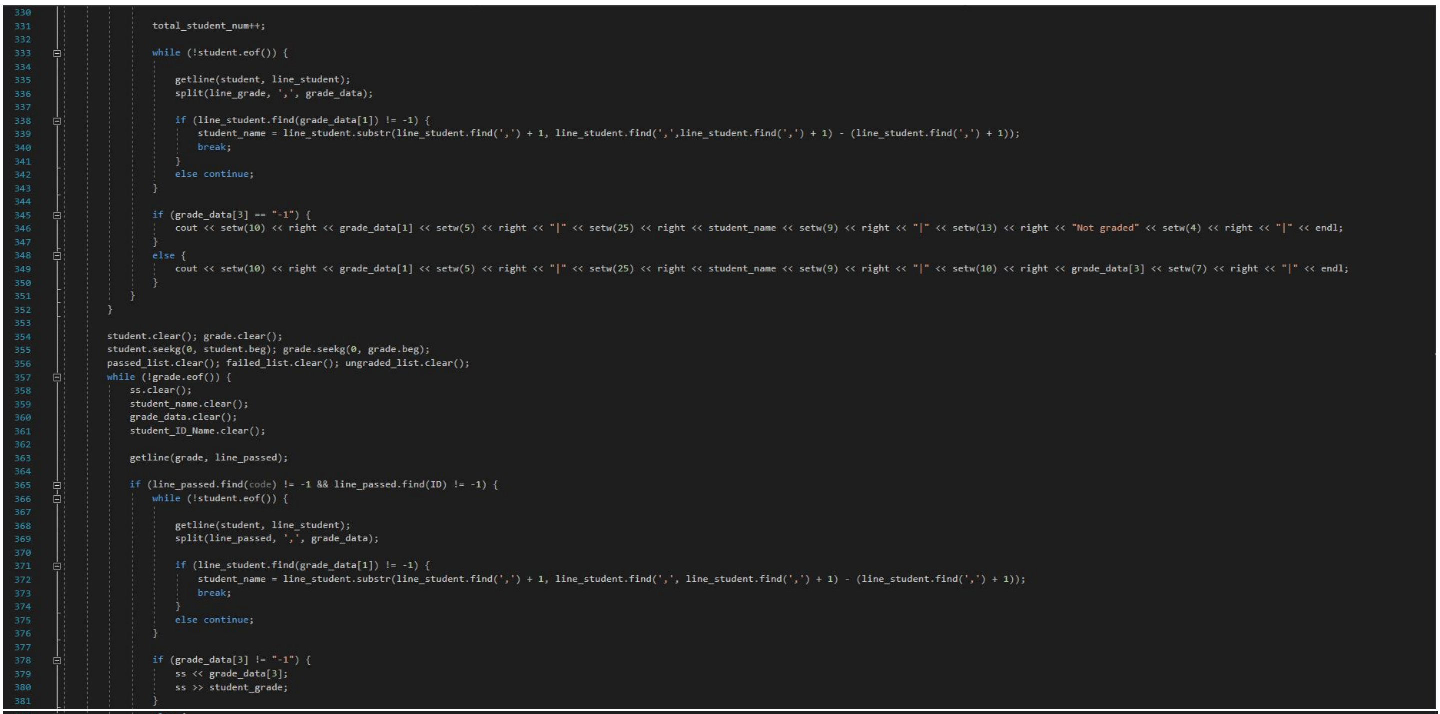
```
C:\Users\IBM\Desktop\Project1\Debug\Project1.exe

-------------------- TEACHER: S0952 --------------------

1. Change password [cp]        2. Open course [oc]          3. Summarize course [sc] [Course code]
4. Modify course [mc] [Course code]      5. Exit [q]
Choose your option : _
```

2.4.5 Summarize course

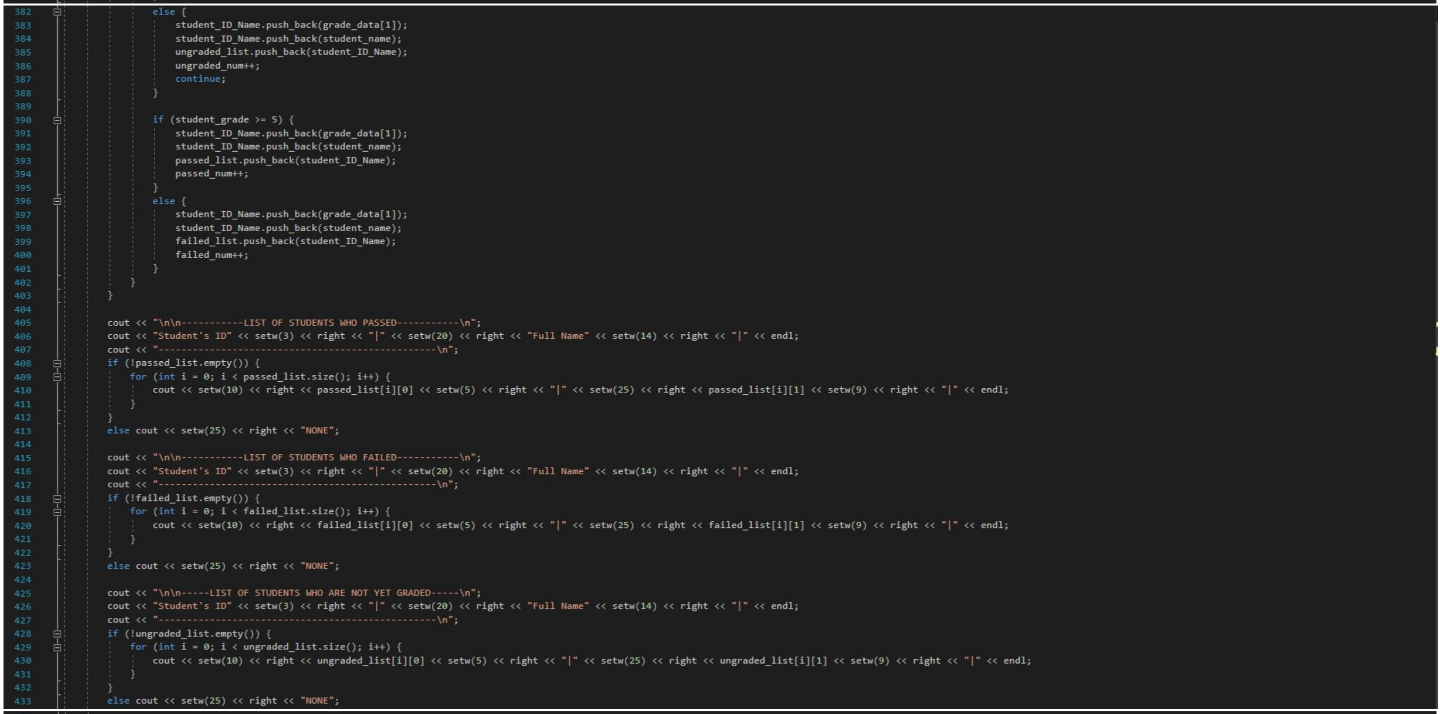
Source code:



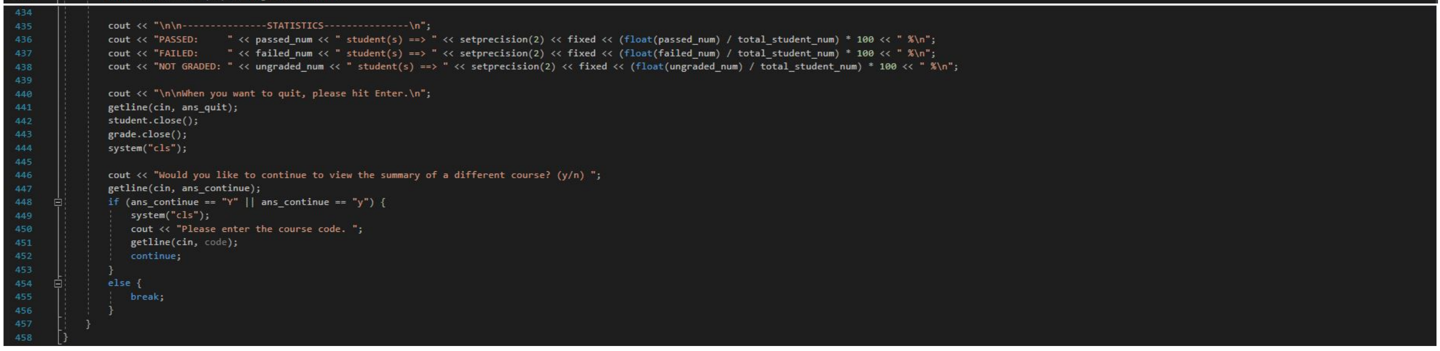
```cpp
void Teacher::summarize_course(string code) {
    while (true) {
        system("cls");
        string ans_quit, ans_continue;
        int student_grade, passed_num = 0, failed_num = 0, ungraded_num = 0, total_student_num = 0;
        string line_grade, line_student, line_passed, line_failed, line_ungraded;
        string course_name, student_name;
        bool found = false;
        vector<string> grade_data, student_ID_Name;
        vector< vector<string> > passed_list, failed_list, ungraded_list;
        fstream grade;
        fstream student;
        stringstream ss;

        while (true) {
            for (int i = 0; i < user_course.size(); i++) {
                if (user_course[i][0] == code) {
                    found = true;
                    course_name = user_course[i][2];
                    break;
                }
            }
            if (found == false) {
                cout << "--------------------------------------------------WARNING----------------------------------------------------" << endl;
                cout << "The course code you entered is not among the codes of your current courses. Please re-enter a valid course code.\n";
                cout << "If you want to return to the function menu, please input 'q'.\n";
                getline(cin, code);
                if (code == "q") {
                    break;
                }
                system("cls");
                continue;
            }
            else break;
        }

        if (code == "q") {
            break;
        }

        cout << "--------------------" << code << " - " << course_name << "--------------------\n\n";
        grade.open("D:\\C++ programs\\Visual Studio\\Draft Project 2\\Data\\grade.csv", ios::in);
        student.open("D:\\C++ programs\\Visual Studio\\Draft Project 2\\Data\\student.csv", ios::in);

        cout << "Student's ID" << setw(3) << right << "|" << setw(20) << right << "Full Name" << setw(14) << right << "|" << setw(10) << right << "Grade" << setw(7) << right << "|" << endl;
        cout << "----------------------------------------------------------------\n";
        while (!grade.eof()) {
            student_name.clear();
            grade_data.clear();
            getline(grade, line_grade);

            if (line_grade.find(code) != -1 && line_grade.find(ID) != -1) {
```



```cpp

                total_student_num++;

                while (!student.eof()) {

                    getline(student, line_student);
                    split(line_grade, ',', grade_data);

                    if (line_student.find(grade_data[1]) != -1) {
                        student_name = line_student.substr(line_student.find(',') + 1, line_student.find(',',line_student.find(',') + 1) - (line_student.find(',') + 1));
                        break;
                    }
                    else continue;
                }

                if (grade_data[3] == "-1") {
                    cout << setw(10) << right << grade_data[1] << setw(5) << right << "|" << setw(25) << right << student_name << setw(9) << right << "|" << setw(13) << right << "Not graded" << setw(4) << right << "|" << endl;
                }
                else {
                    cout << setw(10) << right << grade_data[1] << setw(5) << right << "|" << setw(25) << right << student_name << setw(9) << right << "|" << setw(10) << right << grade_data[3] << setw(7) << right << "|" << endl;
                }
            }
        }

        student.clear(); grade.clear();
        student.seekg(0, student.beg); grade.seekg(0, grade.beg);
        passed_list.clear(); failed_list.clear(); ungraded_list.clear();
        while (!grade.eof()) {
            ss.clear();
            student_name.clear();
            grade_data.clear();
            student_ID_Name.clear();

            getline(grade, line_passed);

            if (line_passed.find(code) != -1 && line_passed.find(ID) != -1) {
                while (!student.eof()) {

                    getline(student, line_student);
                    split(line_passed, ',', grade_data);

                    if (line_student.find(grade_data[1]) != -1) {
                        student_name = line_student.substr(line_student.find(',') + 1, line_student.find(',', line_student.find(',') + 1) - (line_student.find(',') + 1));
                        break;
                    }
                    else continue;
                }

                if (grade_data[3] != "-1") {
                    ss << grade_data[3];
                    ss >> student_grade;
                }
```



```cpp
                else {
                    student_ID_Name.push_back(grade_data[1]);
                    student_ID_Name.push_back(student_name);
                    ungraded_list.push_back(student_ID_Name);
                    ungraded_num++;
                    continue;
                }

                if (student_grade >= 5) {
                    student_ID_Name.push_back(grade_data[1]);
                    student_ID_Name.push_back(student_name);
                    passed_list.push_back(student_ID_Name);
                    passed_num++;
                }
                else {
                    student_ID_Name.push_back(grade_data[1]);
                    student_ID_Name.push_back(student_name);
                    failed_list.push_back(student_ID_Name);
                    failed_num++;
                }
            }
        }

        cout << "\n\n-----------LIST OF STUDENTS WHO PASSED-----------\n";
        cout << "Student's ID" << setw(3) << right << "|" << setw(20) << right << "Full Name" << setw(14) << right << "|" << endl;
        cout << "------------------------------------------------\n";
        if (!passed_list.empty()) {
            for (int i = 0; i < passed_list.size(); i++) {
                cout << setw(10) << right << passed_list[i][0] << setw(5) << right << "|" << setw(25) << right << passed_list[i][1] << setw(9) << right << "|" << endl;
            }
        }
        else cout << setw(25) << right << "NONE";

        cout << "\n\n-----------LIST OF STUDENTS WHO FAILED-----------\n";
        cout << "Student's ID" << setw(3) << right << "|" << setw(20) << right << "Full Name" << setw(14) << right << "|" << endl;
        cout << "------------------------------------------------\n";
        if (!failed_list.empty()) {
            for (int i = 0; i < failed_list.size(); i++) {
                cout << setw(10) << right << failed_list[i][0] << setw(5) << right << "|" << setw(25) << right << failed_list[i][1] << setw(9) << right << "|" << endl;
            }
        }
        else cout << setw(25) << right << "NONE";

        cout << "\n\n-----LIST OF STUDENTS WHO ARE NOT YET GRADED-----\n";
        cout << "Student's ID" << setw(3) << right << "|" << setw(20) << right << "Full Name" << setw(14) << right << "|" << endl;
        cout << "------------------------------------------------\n";
        if (!ungraded_list.empty()) {
            for (int i = 0; i < ungraded_list.size(); i++) {
                cout << setw(10) << right << ungraded_list[i][0] << setw(5) << right << "|" << setw(25) << right << ungraded_list[i][1] << setw(9) << right << "|" << endl;
            }
        }
        else cout << setw(25) << right << "NONE";
```



```cpp

        cout << "\n\n---------------STATISTICS---------------\n";
        cout << "PASSED:     " << passed_num << " student(s) ==> " << setprecision(2) << fixed << (float(passed_num) / total_student_num) * 100 << " %\n";
        cout << "FAILED:     " << failed_num << " student(s) ==> " << setprecision(2) << fixed << (float(failed_num) / total_student_num) * 100 << " %\n";
        cout << "NOT GRADED: " << ungraded_num << " student(s) ==> " << setprecision(2) << fixed << (float(ungraded_num) / total_student_num) * 100 << " %\n";

        cout << "\n\nWhen you want to quit, please hit Enter.\n";
        getline(cin, ans_quit);
        student.close();
        grade.close();
        system("cls");

        cout << "Would you like to continue to view the summary of a different course? (y/n) ";
        getline(cin, ans_continue);
        if (ans_continue == "Y" || ans_continue == "y") {
            system("cls");
            cout << "Please enter the course code. ";
            getline(cin, code);
            continue;
        }
        else {
            break;
        }
    }
}
```

Documentation: • The first **while(true)** loop is used to check if the lecturer/teacher has the course with the code that he/she has inputted:

+ The first elements of every sub-vector of the 2D vector "user_course" are all the codes of the courses that the user currently has.

Therefore, we will compare the inputted course code (stored in the string object "code") with these elements. If a match is found, which means the inputted course code is valid, the **bool** variable "found" will turn **true**, the course name (the third element of the current sub-vector) will be assigned into the string object "course_name" and we immediately break out of the loop. Otherwise, "found" will stay **false** and the loop will run to the end.

+ The next **if** statement checks the value of "found". If "found" is still **false**, which means the inputted course code is invalid, a warning (line 301 to 303) will be shown and the user will be asked to input the course code again. Otherwise, if "found" has turned **true**, we immediately break out of the **while(true)** loop.

+ Because everything is placed within a **while(true)** loop, it is made sure that we only break out of this loop when the user has successfully inputted a valid course code.

+ When the wrong course code warning shows up, the user also has the option to quit. If the user inputs "q" instead of a valid course code, we immediately break out of the **while(true)** loop because of the **if** statement from line 305 to 307. And then, directly beneath the **while(true)** loop, another **if** statement (line 314 to 316) breaks us out of the function "summarize_course" and return to the command menu. If the user inputs anything other than "q", this whole process will be ignored.

• From line 318 to 352, we scan the two .csv files **grade.csv** and **student.csv** to print each student's ID Number, Name and Grade to the console:

+ Line 318: Printing the course code (stored in the string object "code") and the name of the course (stored in the string object "course_name") to the console.

+ Lines 319 and 320: Opening the two .csv files "grade" and "student" in **input mode** (Read-Only mode) to obtain the student's ID Number, Grade and Name.

+ Lines 322 and 323: Printing the titles of the columns of the Grade table. There are three columns in total: Student's ID, Full Name and Grade.

+ In the **while(!grade.eof())** loop from line 324 to 352, we will get each line out of the **grade.csv** file, extract the students' IDs and Grades out of these lines. And with the extracted students' IDs, we can identify the students' Names and extract them from the **student.csv** file.

The **if** statement at line 329 checks the current line (stored in the string object "line_grade") to see if this line has the correct course code (stored in the string object "code") and the correct teacher's ID (stored in the string object "ID"). The reason we must have these two constraints is because one course may be taught by many teachers and one teacher may teach many different courses). If the line does not have "code" or "ID", the condition is evaluated to be **false** and we return to the beginning of the loop to get the next line. Otherwise, we continue into the block of this **if** statement.

"total_student_num" is an integer variable used to count the number of students who are currently in the course of this teacher. This variable will be used later for more statistics.

In the **while(!student.eof())** loop from line 333 to 343, we will get each line out of the **student.csv** file and extract the student's Name out of the correct line, based on the current student's ID we have. By calling the function "split", we split up and **push_back** the student's course code, ID Number, Teacher's ID and Grade respectively as the first, second, third and fourth element of the string vector "grade_data". Then, the **if** statement at line 338

if (line_grade.find(code) != -1 && line_grade.find(ID) != -1)

will check if the current line has the correct student's ID (second element of the "grade_data" vector = grade_data[1]). If the line does not have a student's ID that matches, we ignore the **if** statement block and continue getting the next line from **student.csv**. Otherwise, if the matching student's ID is found in that line, we will extract the student's Name by using the **substr()** function on the string object "line_student" and assigning that sub-string into the string object "student_name". After that, we immediately break out of the **while(!student.eof())** loop.

The last pair of **if** and **else** statements are for printing the student's ID, Name and Grade in that order. In the **grade.csv** file, any students who are not yet graded will have the string "-1" as their current grades. Therefore, if the student's current grade is "-1", which means the fourth element of the vector "grade_data" ("grade_data"[3]) is "-1", the **if** statement block will be executed and that student's Grade slot will display "Not graded"; otherwise, if the student's current grade is any value from "0" to "10", the **else** statement block will be executed and that student's Grade slot will display his/her current grade.

Finally, we return to the beginning of the **while(!grade.eof())** loop. If **!grade.eof()** still evaluates to **true**, which means the end of the file has not been reached yet, we clear the content of the string object "student_name" and string vector "grade_data" for storing new data and continue getting lines from the files.

• Lines 354 and 355: Clearing the **eof** (end-of-file) flag and return the cursor to the beginning of both files **grade.csv** and **student.csv**. This step is necessary for will scan everything of these two files again in the next **while()** loop.

• From line 357 to 403, we scan the two .csv files **grade.csv** and **student.csv** again. But this time, instead of printing the Grade Table to the console, we sort all students into three groups: Passed, Failed and Not Graded.

+ From line 359 to 376: The same procedure as before, the student's course code, ID Number, Teacher's ID and Grade are **push_back**ed into the string vector "grade_data" in that order by the "split" function. The student's Name is assigned into the string object "student_name".

+ With the first pair of **if** and **else** statements, we check if the student's current grade is the same as the string "-1" or not. If the grade is not "-1", we stream it (which is also the fourth element of "grade_data" = grade_data[3]) into the **sstream** object "ss", then we stream from "ss" into the float variable "student_grade" for later numerical comparisons. Otherwise, if grade_data[3] is "-1", the **else** statement block will be executed: we **push_back** the student's ID and Name **in that order** into the string vector "student_ID_Name";then, we **push_back** the vector "student_ID_Name" into the 2D string vector "ungraded_list" and let "ungraded_num" (the integer variable to count the number of students who are not graded) increase by one.

+ With the second pair of **if** and **else** statements, now we already have the numerical form of the student's grade stored in "student_grade", we need to compare it with 5 to see if the student passed or failed. If "student_grade" >= 5, we **push_back** the student's ID and Name **in that order** into the string vector "student_ID_Name";then, we **push_back** the vector "student_ID_Name" into the 2D string vector "passed_list" and let "passed_num" (the integer variable to count the number of students who are passed) increase by one. Otherwise, if "student_grade" < 5, we carry out the same procedure as above, but now with the 2D vector "failed_list" and the counting integer variable "failed_num".

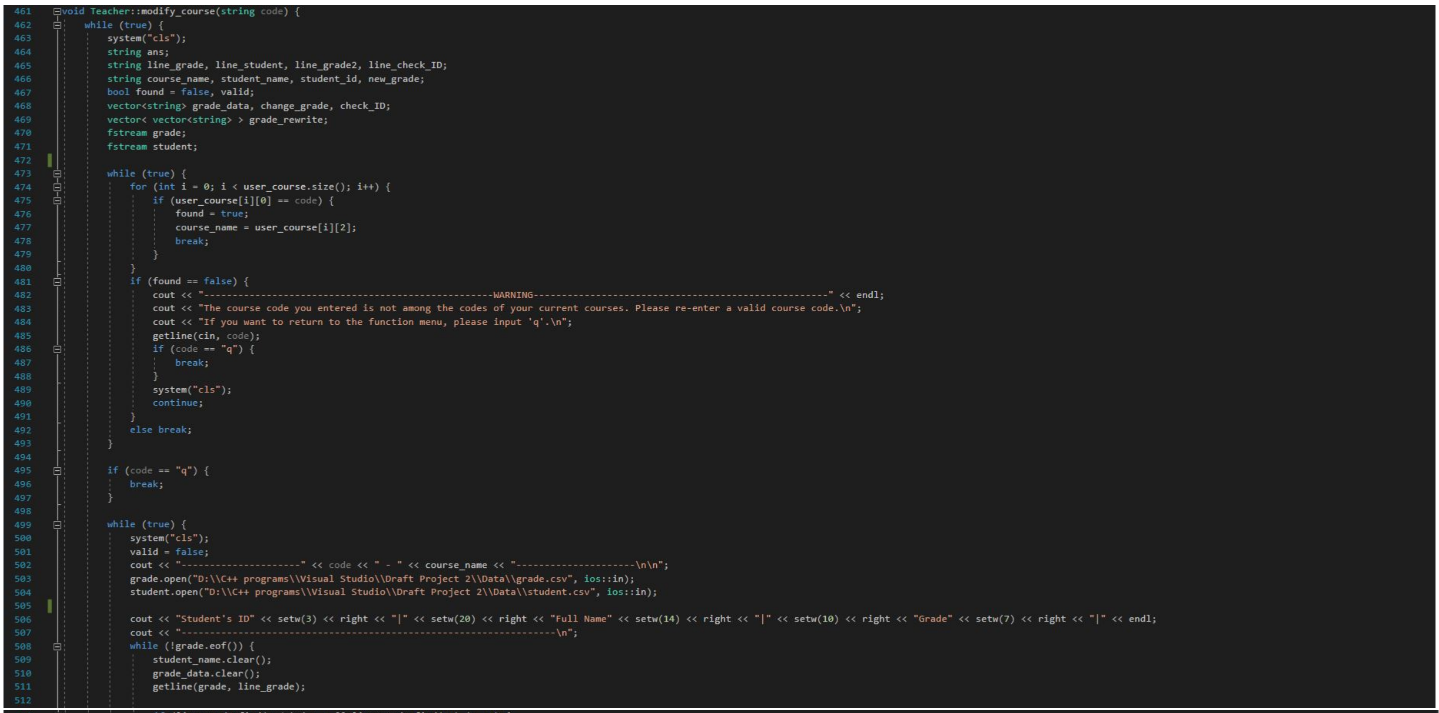
• From line 406 to 439: Printing to the console the list of students who passed, list of students who failed and list of student who are not graded using the three 2D string vectors: "passed_list", "failed_list" and "ungraded_list". Moreover, we use the the variables "passed_num" (number of students who passed), "failed_num" (number of students who failed), "ungraded_num" (number of students who are not graded) and "total_student_num" (total number of students of the course) to calculate the percentage that each group occupies.

• From line 441 to 445, we give the user the option to quit the current session. If the user hits Enter, both files **student.h** and **grade.h** will be closed and the screen will be cleared.

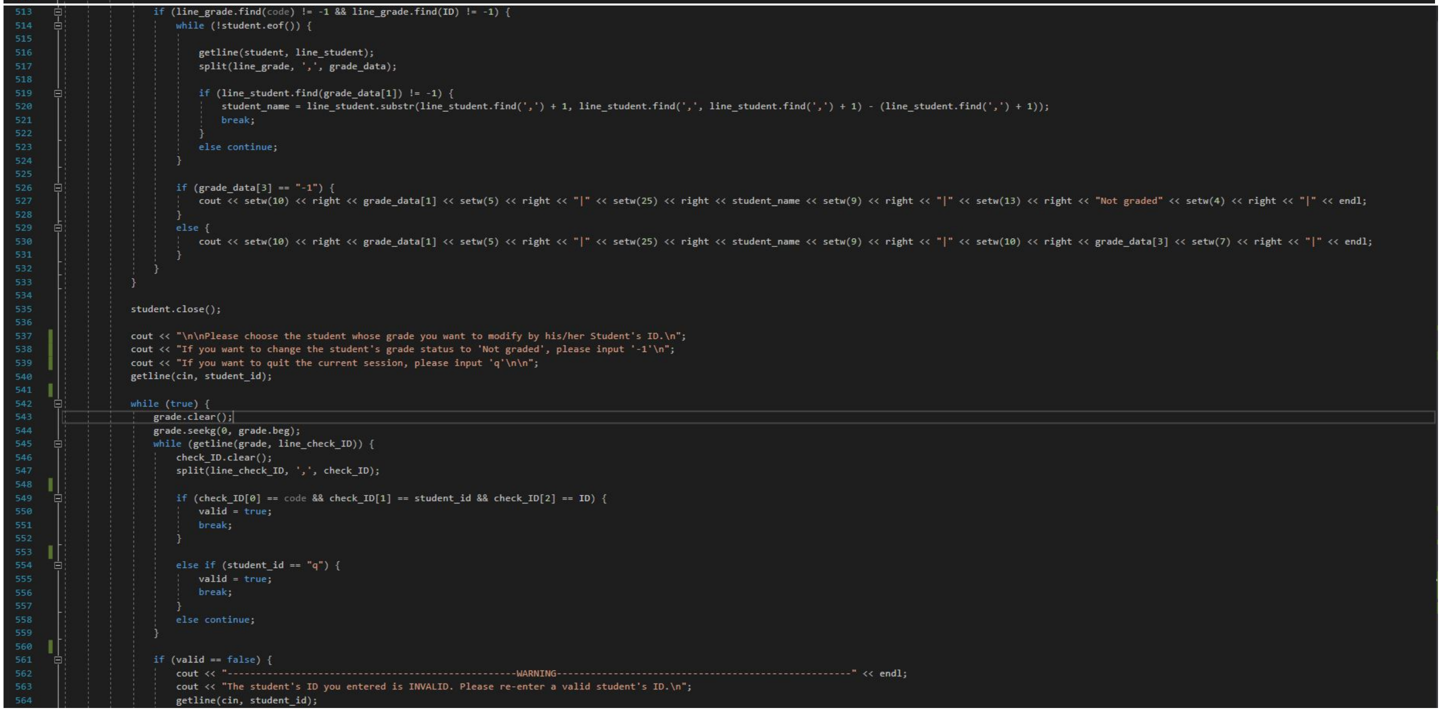
• From line 447 to 457, we ask the user if he/she wants to continue to view the summary of a different course. If the user inputs ”Y” or ”y”, the screen will be cleared and the user will be asked to input a new course code into the variable ”code”. After inputting, the line **continue;** return the program flow to the **first while(true)** loop that contains everything and the whole procedure starts again. Otherwise, if the user input anything or than ”y” or ”Y”, we immediately break out of the **while(true)** loop and return to the command menu.

2.4.6 Modify course

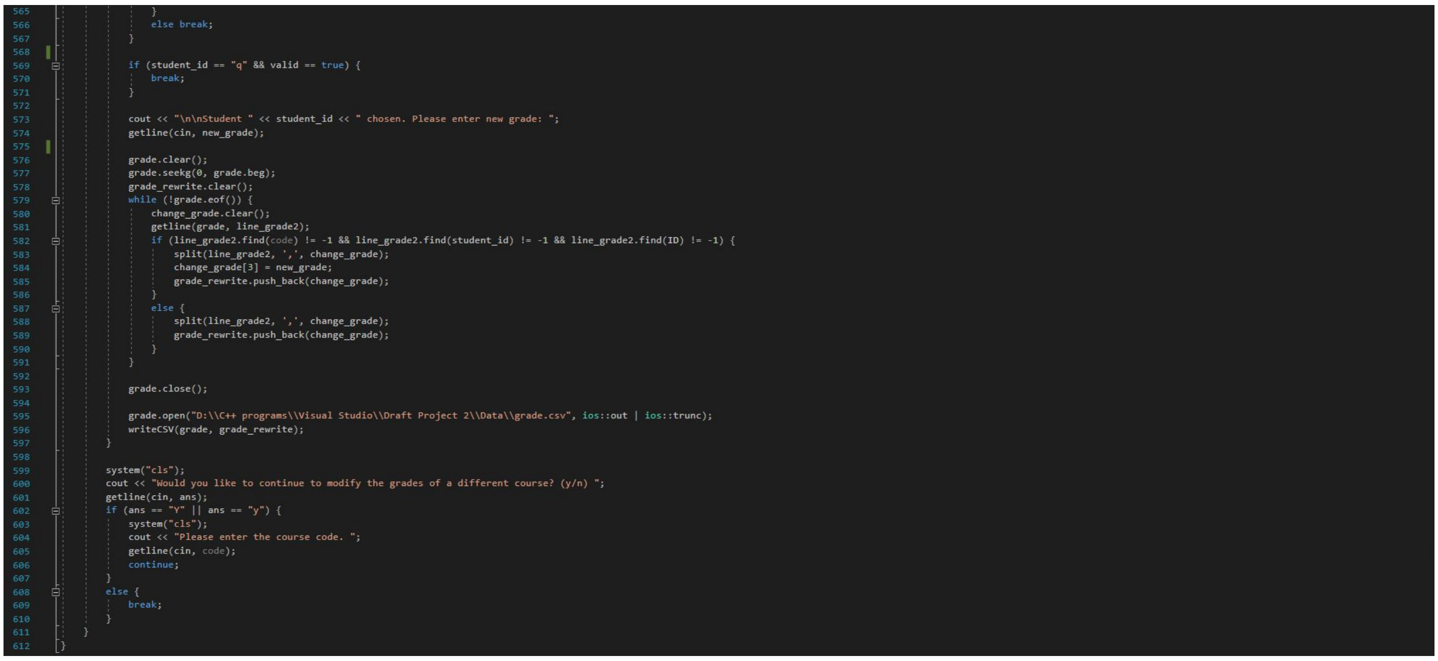
Source code:



```cpp
void Teacher::modify_course(string code) {
    while (true) {
        system("cls");
        string ans;
        string line_grade, line_student, line_grade2, line_check_ID;
        string course_name, student_name, student_id, new_grade;
        bool found = false, valid;
        vector<string> grade_data, change_grade, check_ID;
        vector< vector<string> > grade_rewrite;
        fstream grade;
        fstream student;

        while (true) {
            for (int i = 0; i < user_course.size(); i++) {
                if (user_course[i][0] == code) {
                    found = true;
                    course_name = user_course[i][2];
                    break;
                }
            }
            if (found == false) {
                cout << "--------------------------------------------------WARNING---------------------------------------------------" << endl;
                cout << "The course code you entered is not among the codes of your current courses. Please re-enter a valid course code.\n";
                cout << "If you want to return to the function menu, please input 'q'.\n";
                getline(cin, code);
                if (code == "q") {
                    break;
                }
                system("cls");
                continue;
            }
            else break;
        }

        if (code == "q") {
            break;
        }

        while (true) {
            system("cls");
            valid = false;
            cout << "--------------------" << code << " - " << course_name << "--------------------\n\n";
            grade.open("D:\\C++ programs\\Visual Studio\\Draft Project 2\\Data\\grade.csv", ios::in);
            student.open("D:\\C++ programs\\Visual Studio\\Draft Project 2\\Data\\student.csv", ios::in);

            cout << "Student's ID" << setw(3) << right << "|" << setw(20) << right << "Full Name" << setw(14) << right << "|" << setw(10) << right << "Grade" << setw(7) << right << "|" << endl;
            cout << "----------------------------------------------------------------\n";
            while (!grade.eof()) {
                student_name.clear();
                grade_data.clear();
                getline(grade, line_grade);

```



```cpp
                if (line_grade.find(code) != -1 && line_grade.find(ID) != -1) {
                    while (!student.eof()) {

                        getline(student, line_student);
                        split(line_grade, ',', grade_data);

                        if (line_student.find(grade_data[1]) != -1) {
                            student_name = line_student.substr(line_student.find(',') + 1, line_student.find(',', line_student.find(',') + 1) - (line_student.find(',') + 1));
                            break;
                        }
                        else continue;
                    }

                    if (grade_data[3] == "-1") {
                        cout << setw(10) << right << grade_data[1] << setw(5) << right << "|" << setw(25) << right << student_name << setw(9) << right << "|" << setw(13) << right << "Not graded" << setw(4) << right << "|" << endl;
                    }
                    else {
                        cout << setw(10) << right << grade_data[1] << setw(5) << right << "|" << setw(25) << right << student_name << setw(9) << right << "|" << setw(10) << right << grade_data[3] << setw(7) << right << "|" << endl;
                    }
                }
            }

            student.close();

            cout << "\n\nPlease choose the student whose grade you want to modify by his/her Student's ID.\n";
            cout << "If you want to change the student's grade status to 'Not graded', please input '-1'\n";
            cout << "If you want to quit the current session, please input 'q'\n\n";
            getline(cin, student_id);

            while (true) {
                grade.clear();
                grade.seekg(0, grade.beg);
                while (getline(grade, line_check_ID)) {
                    check_ID.clear();
                    split(line_check_ID, ',', check_ID);

                    if (check_ID[0] == code && check_ID[1] == student_id && check_ID[2] == ID) {
                        valid = true;
                        break;
                    }

                    else if (student_id == "q") {
                        valid = true;
                        break;
                    }
                    else continue;
                }

                if (valid == false) {
                    cout << "--------------------------------------------------WARNING---------------------------------------------------" << endl;
                    cout << "The student's ID you entered is INVALID. Please re-enter a valid student's ID.\n";
                    getline(cin, student_id);
```



```
565                 }
566                 else break;
567             }
568 
569             if (student_id == "q" && valid == true) {
570                 break;
571             }
572 
573             cout << "\n\nStudent " << student_id << " chosen. Please enter new grade: ";
574             getline(cin, new_grade);
575 
576             grade.clear();
577             grade.seekg(0, grade.beg);
578             grade_rewrite.clear();
579             while (!grade.eof()) {
580                 change_grade.clear();
581                 getline(grade, line_grade2);
582                 if (line_grade2.find(code) != -1 && line_grade2.find(student_id) != -1 && line_grade2.find(ID) != -1) {
583                     split(line_grade2, ',', change_grade);
584                     change_grade[3] = new_grade;
585                     grade_rewrite.push_back(change_grade);
586                 }
587                 else {
588                     split(line_grade2, ',', change_grade);
589                     grade_rewrite.push_back(change_grade);
590                 }
591             }
592 
593             grade.close();
594 
595             grade.open("D:\\C++ programs\\Visual Studio\\Draft Project 2\\Data\\grade.csv", ios::out | ios::trunc);
596             writeCSV(grade, grade_rewrite);
597         }
598 
599         system("cls");
600         cout << "Would you like to continue to modify the grades of a different course? (y/n) ";
601         getline(cin, ans);
602         if (ans == "Y" || ans == "y") {
603             system("cls");
604             cout << "Please enter the course code. ";
605             getline(cin, code);
606             continue;
607         }
608         else {
609             break;
610         }
611     }
612 }
```

Documentation: • From line 463 to 536: the same as the part from line 293 to line 353 of the "summarize_course" function. This part checks the validity of the course code inputted by the user. If the course code is valid, the Grade Table containing the ID Number, Name and Grade of each student will be displayed.

• From line 538 to 597, this part will let the user enter the ID of the student whose grade he/she wants to modify. Then, the user will be prompted to enter a new grade for the chosen student. After that, the whole **grade.csv** file will be re-written to save the new modification: + Line 538 to 541: this part prints all the usage instructions and then prompts the user to
input the student's ID.

+ From line 543 to 572:
The **while(true)** loop is used to check the validity of the inputted student's ID. The **while (getline(grade, line_check_ID))** loop gets each line out of the file **grade.csv** and stores it in the string object "line_check_ID". The function "split" is called to **push_back** the student's course code, ID Number and Teacher's ID into the string vector "check_ID" as the first, second and third elements. The **if** statement:

if (check_ID[0] == code && check_ID[1] == student_id && check_ID[2] == ID)

will check if the student's course code (check_ID[0]) matches "code"; ID Number (check_ID[1]) matches "student_id" (the input of the user) and Teacher's ID (check_ID[2]) matches "ID". If the condition evaluates to be **true**, the **bool** variable "valid" will turn **true** and we break out of the loop. Otherwise, this **if** statement is simply skipped.

If the user inputs "q", which means he/she wants to quit the current session, the next **else if** statement:

else if (student_id == "q")

will evaluate to **true**. The **bool** variable "valid" will turn **true** and we break out of the loop.

If none of the above conditions is **true**, the loop continues till the end of the file **grade.csv**.

The **if** statement:

if (valid == false)

checks the value of "valid". If "valid" is still **false**, this means **NO** student has the ID Number that the user has inputted. And for that reason, a warning will be shown, telling the user about the invalidity of the input and asking the user to re-input another student's ID.

Back to the case when the user inputs "q" to quit, "valid" will turn **true** and we break out of the ID checking loop. The first thing we will meet is the **if** statement:

if (student_id == "q" && valid == true)

There are only two cases when we can break out of the ID checking loop: when we input a correct student's ID and when we input "q". Therefore, this **if** statement checks again to determine which is the current case. If the condition evaluates to **true**, which means the user really did input "q" to quit, we will break out of the biggest **while(true)** loop ⇒ Quit the current session.

+ Line 574 to 597:
When the user has inputted a valid student's ID, line 574 and 575 will be executed to obtain the student's new grade.

+ Line 577 to 597:
We clear the **eof** flag and return the cursor to the beginning of the file **grade.csv**. We clear the content of the 2D string vector "grade_rewrite".

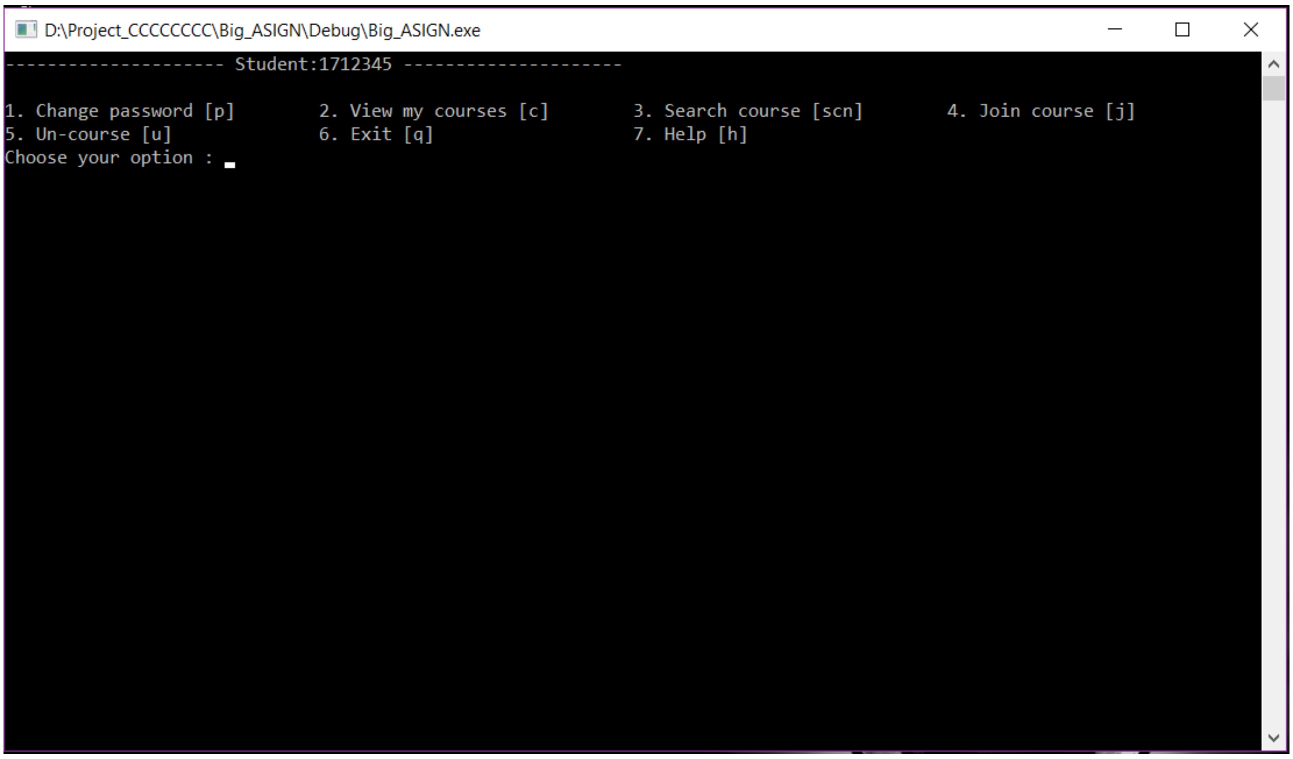
The **while(!grade.eof())** loop gets each line out of the file **grade.csv**. If the line has the correct course code, student's ID and Teacher's ID (same as "code", "student_id" and "ID"), we will call the function "split" to add all data on that line into the string vector "change_grade"; then, we assign the old grade - fourth element of "change_grade" (change_grade[3]) - with the new grade (stored in the string object "new_grade") and **push_back** "change_grade" into the 2D string vector "grade_rewrite". Otherwise, the function "split" will also be called for other unchanged lines, **push_back** the data of these lines into "change_grade" and **push_back** "change_grade" into "grade_rewrite" in every iteration.

Once the loop is over, **grade.csv** will be closed and re-opened immediately in **output** and **truncated** mode (Delete all current content and Write-Only). The function "writeCSV" is called to rewrite everything, including the change in grade of the chosen student, from the 2D string vector "grade_rewrite" into the **grade.csv** file.

Because everything is placed within a **while(true)** loop, when the call to "writeCSV" function is done, the Grade Table will be displayed again, but this time, with the change the user has made. Everything happens so fast that it feels like we have instantly changed the grade.

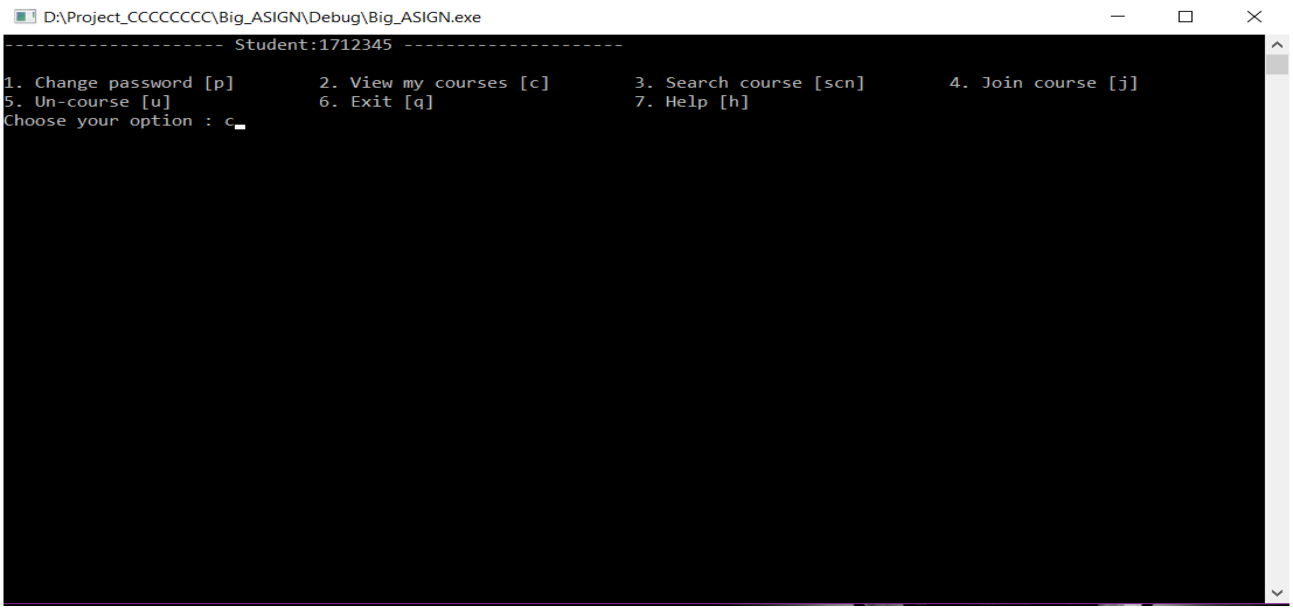
+ From line 600 to 611: This part runs when the user has inputted "q" to quit the session. The user will be asked if he/she wants to continue to modify the grades of a different course. If the user inputs "y" or "Y", he/she will be prompted to input the new course code. Otherwise, we break out of the "modify_course" function and return to the command menu.

2.5 Student



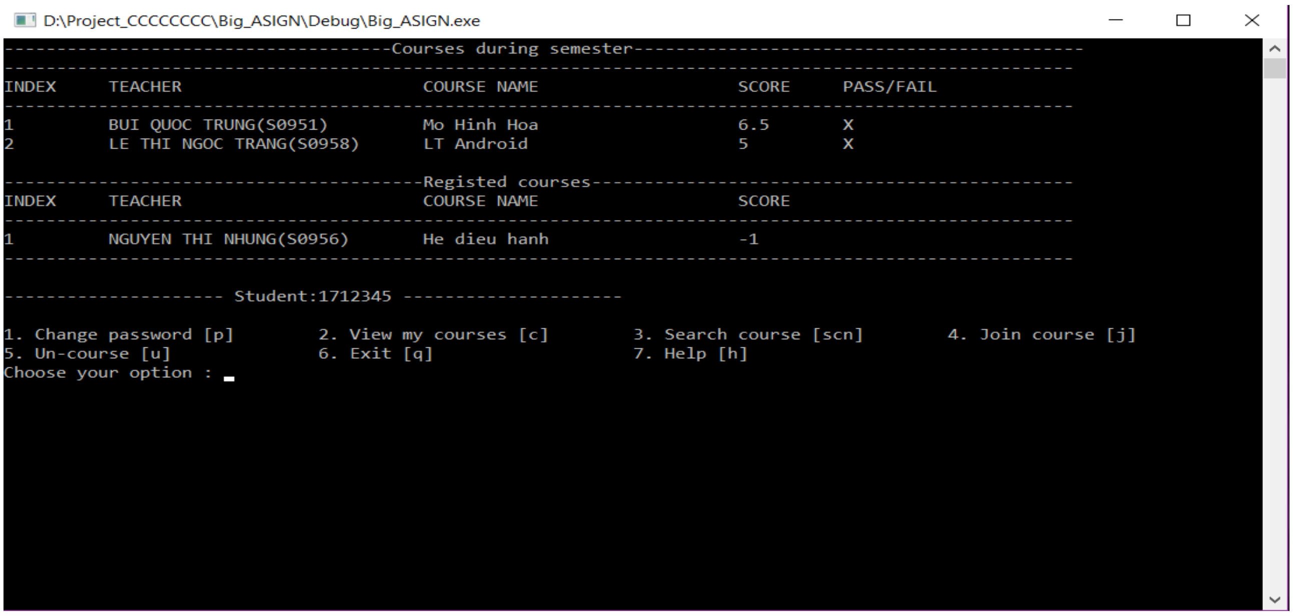
```
D:\Project_CCCCCCCC\Big_ASIGN\Debug\Big_ASIGN.exe
-------------------- Student:1712345 --------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option : 
```

2.5.1 Action

- Use the command is read from in main and redirect to suitable function in student header
- Read command lead to view course function
- 

```
D:\Project_CCCCCCCC\Big_ASIGN\Debug\Big_ASIGN.exe
-------------------- Student:1712345 --------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option : c
```

- Redirect to view course function

- 

```
D:\Project_CCCCCCCC\Big_ASIGN\Debug\Big_ASIGN.exe
-----------------------------------Courses during semester----------------------------------------
------------------------------------------------------------------------------------------------
INDEX     TEACHER                        COURSE NAME                    SCORE     PASS/FAIL
------------------------------------------------------------------------------------------------
1         BUI QUOC TRUNG(S0951)          Mo Hinh Hoa                    6.5       X
2         LE THI NGOC TRANG(S0958)       LT Android                     5         X

----------------------------------------Registed courses-----------------------------------------
INDEX     TEACHER                        COURSE NAME                    SCORE
------------------------------------------------------------------------------------------------
1         NGUYEN THI NHUNG(S0956)        He dieu hanh                   -1
------------------------------------------------------------------------------------------------

-------------------- Student:1712345 --------------------

1. Change password [p]         2. View my courses [c]         3. Search course [scn]         4. Join course [j]
5. Un-course [u]               6. Exit [q]                    7. Help [h]
Choose your option : _
```

2.5.2 Change password

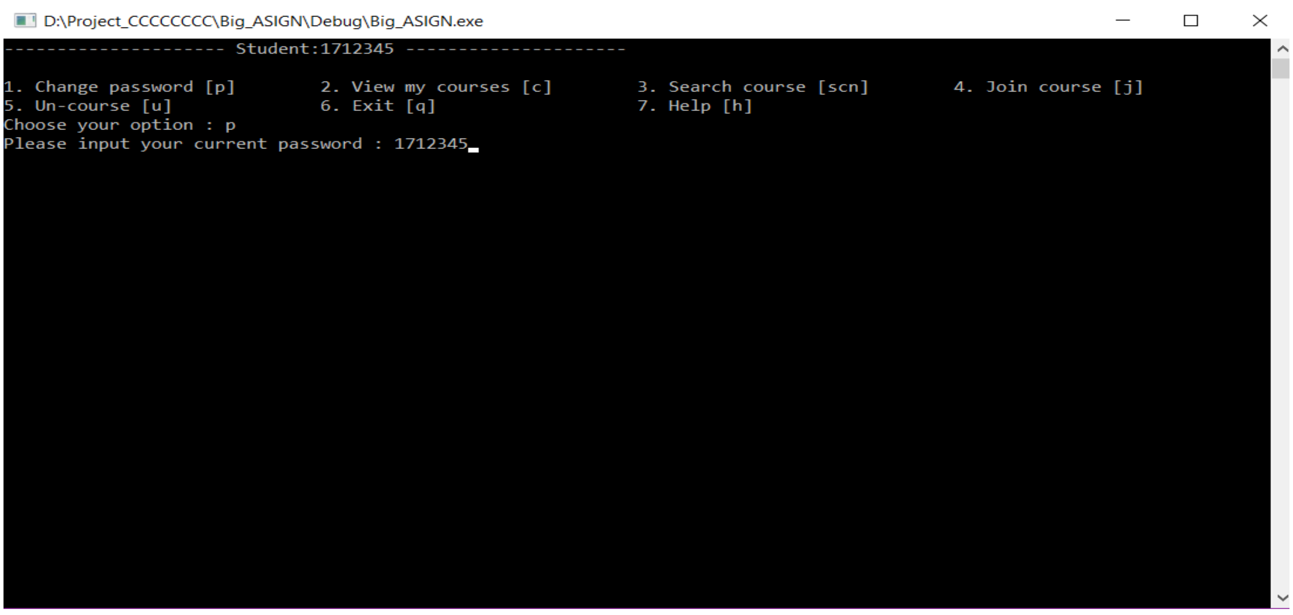
Code

1. Read user.csv into data vector
2. Read current password from keyboard
3. Check current password
4. Read new password from keyboard
5. Loop through data vector and change password
6. Write data vector to user.csv

```
void Student::Change_password(void)
{
    std::vector<std::vector<std::string> > data;
    std::string new_password,old_password;  //Khởi tạo mảng password mới
    std::ifstream user_in(".\\data\\user.csv"); //Khởi tạo biến input user.csv
    readCSV(user_in, data); //Đọc file và ghi thông tin vào mảng data[][]
    std::cout << "Please input your current password : ";
    getline(std::cin, old_password); //Đọc password mới từ bàn phím
    if (old_password != PWD)
    {
        std::cout << "Your current password is wrong !!\n";
        system("pause");
        system("cls");
        return;
    }
    std::cout << "Please input your new password : ";
    getline(std::cin, new_password); //Đọc password mới từ bàn phím
    for (int i = 0; i < data.size(); i++) //Chạy vòng lặp dò từ đầu đến cuối
    {
        if (data[i][0] == ID) //Nếu thông tin trùng với ID
        {
            data[i][1] = new_password; //Thay đổi password
        }
    }
    std::ofstream user_out(".\\data\\user.csv", std::ofstream::trunc); //Khởi tạo biến output user.csv
    writeCSV(user_out, data); //Ghi đè mảng data[][] và file user.csv
    data.clear(); //Làm trống mảng data[][]
    std::cout << "CHANGE PASSWORD SUCCESSFULLY !!\n";
    system("pause");
    exit(0);
```
7. }

User

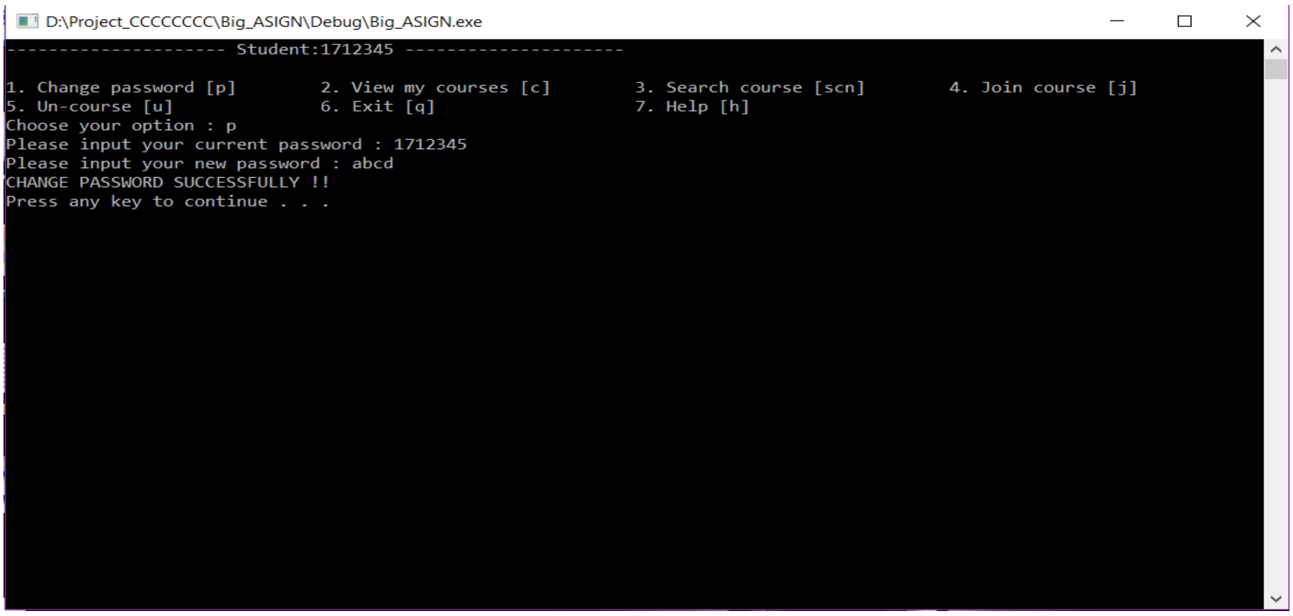
1. Check current password



```
D:\Project_CCCCCCCC\Big_ASIGN\Debug\Big_ASIGN.exe
-------------------- Student:1712345 --------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option : p
Please input your current password : 1712345
```

2.

3. Read new password from keyboard

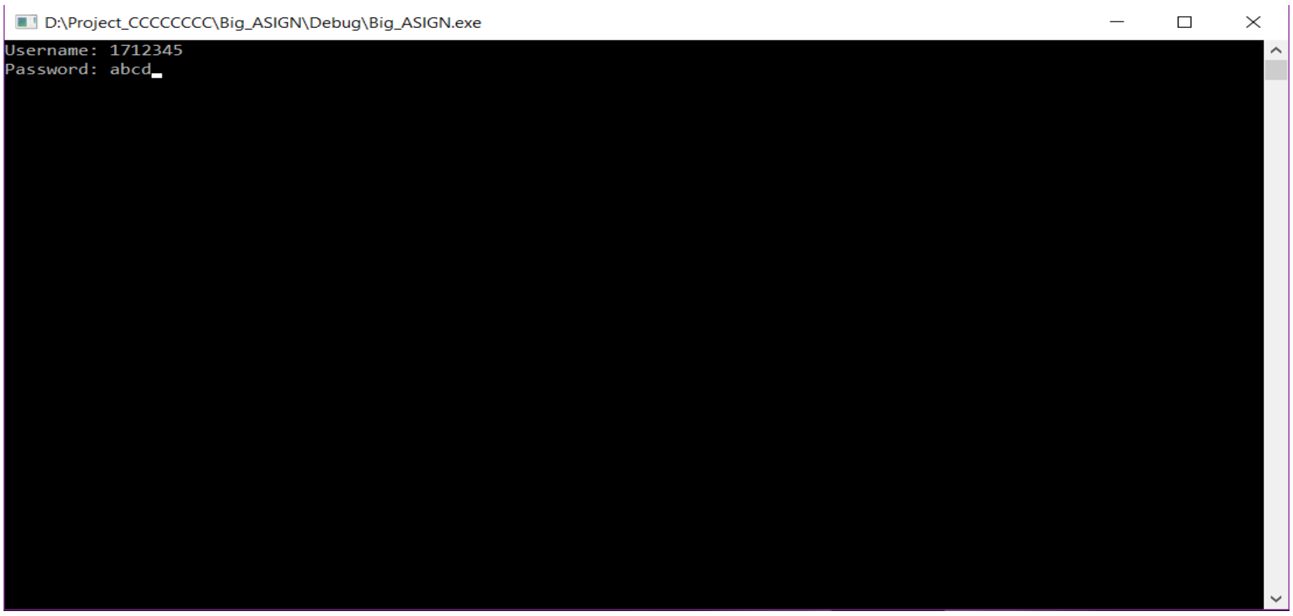


```
D:\Project_CCCCCCCC\Big_ASIGN\Debug\Big_ASIGN.exe
-------------------- Student:1712345 --------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option : p
Please input your current password : 1712345
Please input your new password : abcd
CHANGE PASSWORD SUCCESSFULLY !!
Press any key to continue . . .
```

4.

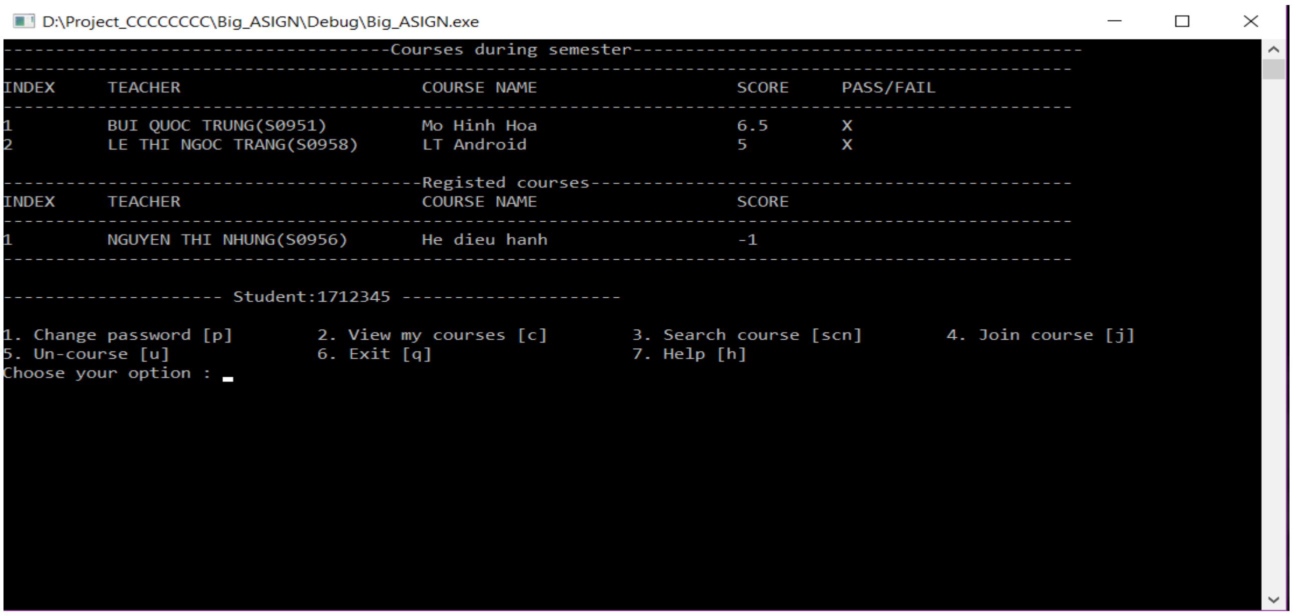
5. Change password successfully and auto quit

6. Login with new password

7. 

```
D:\Project_CCCCCCCC\Big_ASIGN\Debug\Big_ASIGN.exe
Username: 1712345
Password: abcd
```

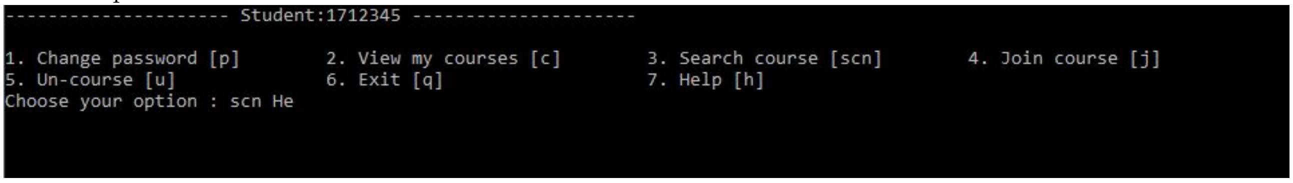
2.5.3 View course

- Show all information about courses that student is studying or registered
- 

```
D:\Project_CCCCCCCC\Big_ASIGN\Debug\Big_ASIGN.exe
-------------------------------------Courses during semester-------------------------------------------
--------------------------------------------------------------------------------------------------------
INDEX     TEACHER                        COURSE NAME                    SCORE     PASS/FAIL
--------------------------------------------------------------------------------------------------------
1         BUI QUOC TRUNG(S0951)          Mo Hinh Hoa                    6.5       X
2         LE THI NGOC TRANG(S0958)       LT Android                     5         X

--------------------------------------Registed courses------------------------------------------
INDEX     TEACHER                        COURSE NAME                    SCORE
--------------------------------------------------------------------------------------------------------
1         NGUYEN THI NHUNG(S0956)        He dieu hanh                   -1
--------------------------------------------------------------------------------------------------------

-------------------- Student:1712345 --------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option :
```

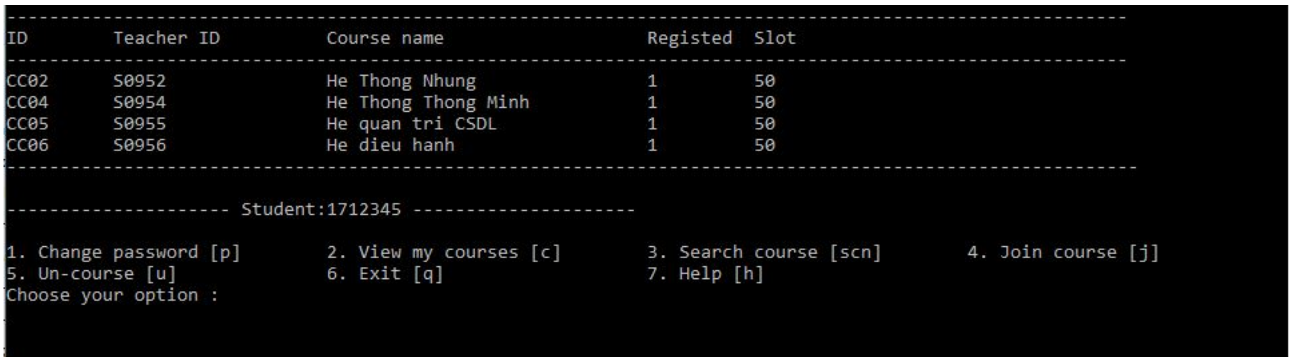
2.5.4 Search course by name

To use function find course, user type command as **scn NAME**.
For example:



```
-------------------- Student:1712345 --------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option : scn He
```

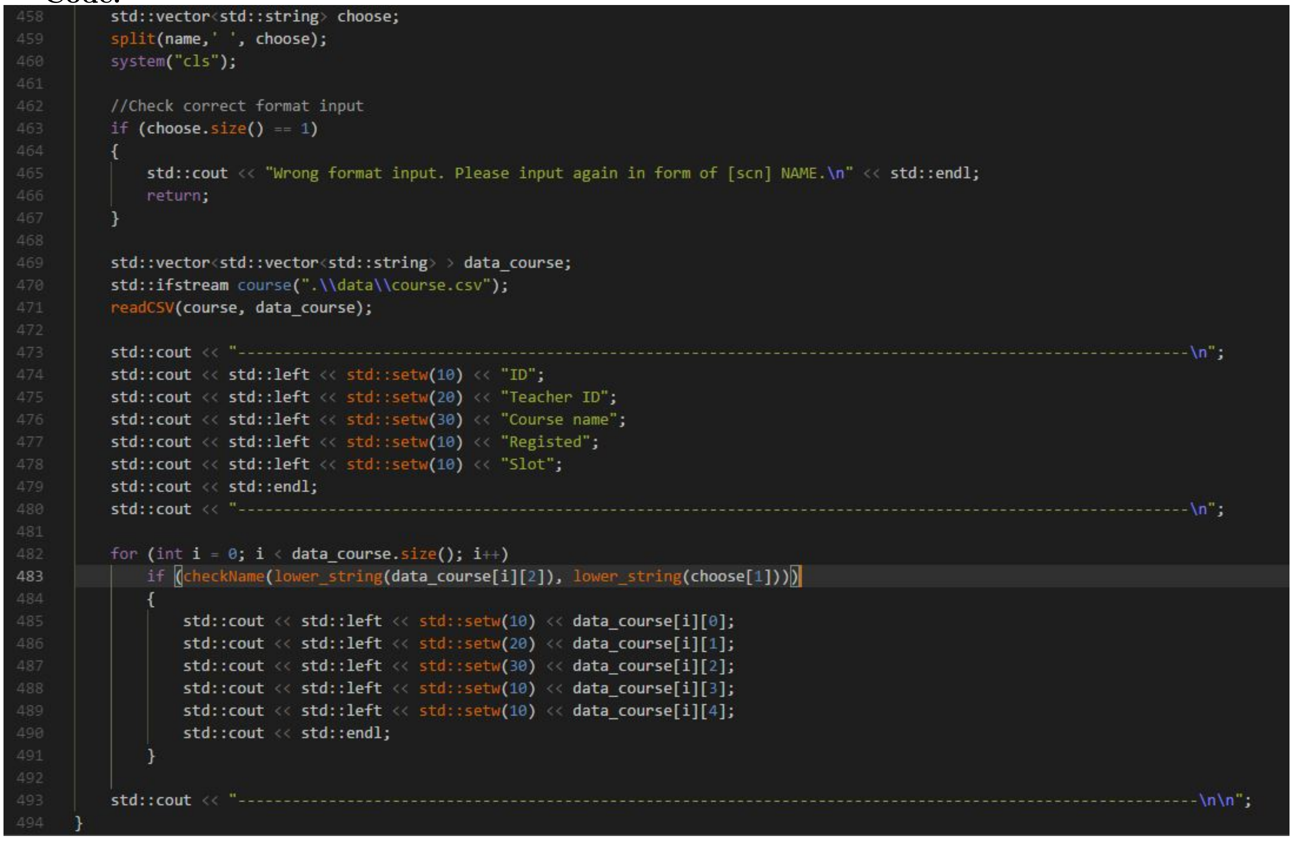
And student will have result as:



```
----------------------------------------------------------------------------------------------------------
ID        Teacher ID          Course name                   Registed  Slot
----------------------------------------------------------------------------------------------------------
CC02      S0952               He Thong Nhung                1         50
CC04      S0954               He Thong Thong Minh           1         50
CC05      S0955               He quan tri CSDL              1         50
CC06      S0956               He dieu hanh                  1         50
-----------------------------------------------------------------------------------------------------------

-------------------- Student:1712345 --------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option :
```

NOTE: user ONLY input 1 search query, DO NOT type more than 1.

Code:



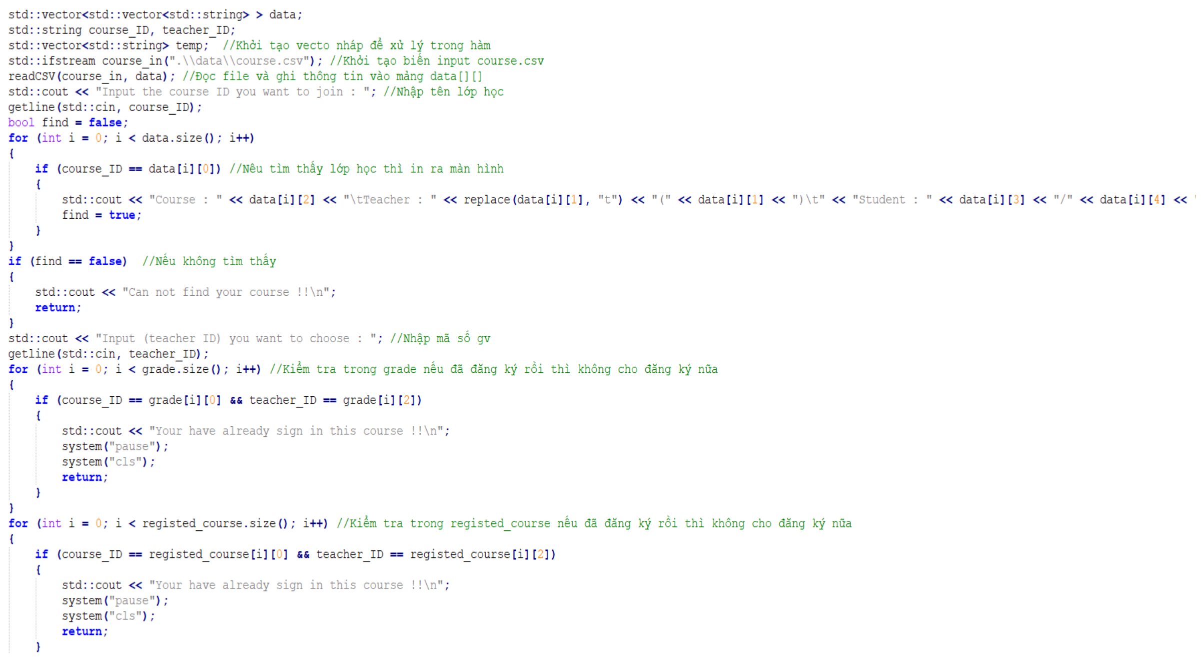
```cpp
    std::vector<std::string> choose;
    split(name,' ', choose);
    system("cls");

    //Check correct format input
    if (choose.size() == 1)
    {
        std::cout << "Wrong format input. Please input again in form of [scn] NAME.\n" << std::endl;
        return;
    }

    std::vector<std::vector<std::string> > data_course;
    std::ifstream course(".\\data\\course.csv");
    readCSV(course, data_course);

    std::cout << "----------------------------------------------------------------------------------------------------------\n";
    std::cout << std::left << std::setw(10) << "ID";
    std::cout << std::left << std::setw(20) << "Teacher ID";
    std::cout << std::left << std::setw(30) << "Course name";
    std::cout << std::left << std::setw(10) << "Registed";
    std::cout << std::left << std::setw(10) << "Slot";
    std::cout << std::endl;
    std::cout << "----------------------------------------------------------------------------------------------------------\n";

    for (int i = 0; i < data_course.size(); i++)
        if (checkName(lower_string(data_course[i][2]), lower_string(choose[1])))
        {
            std::cout << std::left << std::setw(10) << data_course[i][0];
            std::cout << std::left << std::setw(20) << data_course[i][1];
            std::cout << std::left << std::setw(30) << data_course[i][2];
            std::cout << std::left << std::setw(10) << data_course[i][3];
            std::cout << std::left << std::setw(10) << data_course[i][4];
            std::cout << std::endl;
        }

    std::cout << "----------------------------------------------------------------------------------------------------------\n\n";
}
```

- The function open file course.csv in read-only mode.
- The search in course name column to find suitable course.
- To make sure the different in uppercase-lowercase don't cause function miss courses. All names will be lowered.

2.5.5 Join course

Code

1. Read course.csv into data vector
2. Loop through data vector and show the courses suitable
3. Read teacher ID from keyboard
4. Check if this course already registered ?



```cpp
std::vector<std::vector<std::string> > data;
std::string course_ID, teacher_ID;
std::vector<std::string> temp;  //Khởi tạo vecto nháp để xử lý trong hàm
std::ifstream course_in(".\\data\\course.csv"); //Khởi tạo biến input course.csv
readCSV(course_in, data); //Đọc file và ghi thông tin vào mảng data[][]
std::cout << "Input the course ID you want to join : "; //Nhập tên lớp học
getline(std::cin, course_ID);
bool find = false;
for (int i = 0; i < data.size(); i++)
{
    if (course_ID == data[i][0]) //Nêu tìm thấy lớp học thì in ra màn hình
    {
        std::cout << "Course : " << data[i][2] << "\tTeacher : " << replace(data[i][1], "t") << "(" << data[i][1] << ")\t" << "Student : " << data[i][3] << "/" << data[i][4] <<
        find = true;
    }
}
if (find == false)  //Nếu không tìm thấy
{
    std::cout << "Can not find your course !!\n";
    return;
}
std::cout << "Input (teacher ID) you want to choose : "; //Nhập mã số gv
getline(std::cin, teacher_ID);
for (int i = 0; i < grade.size(); i++) //Kiểm tra trong grade nếu đã đăng ký rồi thì không cho đăng ký nữa
{
    if (course_ID == grade[i][0] && teacher_ID == grade[i][2])
    {
        std::cout << "Your have already sign in this course !!\n";
        system("pause");
        system("cls");
        return;
    }
}
for (int i = 0; i < registed_course.size(); i++) //Kiểm tra trong registed_course nếu đã đăng ký rồi thì không cho đăng ký nữa
{
    if (course_ID == registed_course[i][0] && teacher_ID == registed_course[i][2])
    {
        std::cout << "Your have already sign in this course !!\n";
        system("pause");
        system("cls");
        return;
    }
```

5.

6. Check if this course is full ?

7. Increase student number in this course

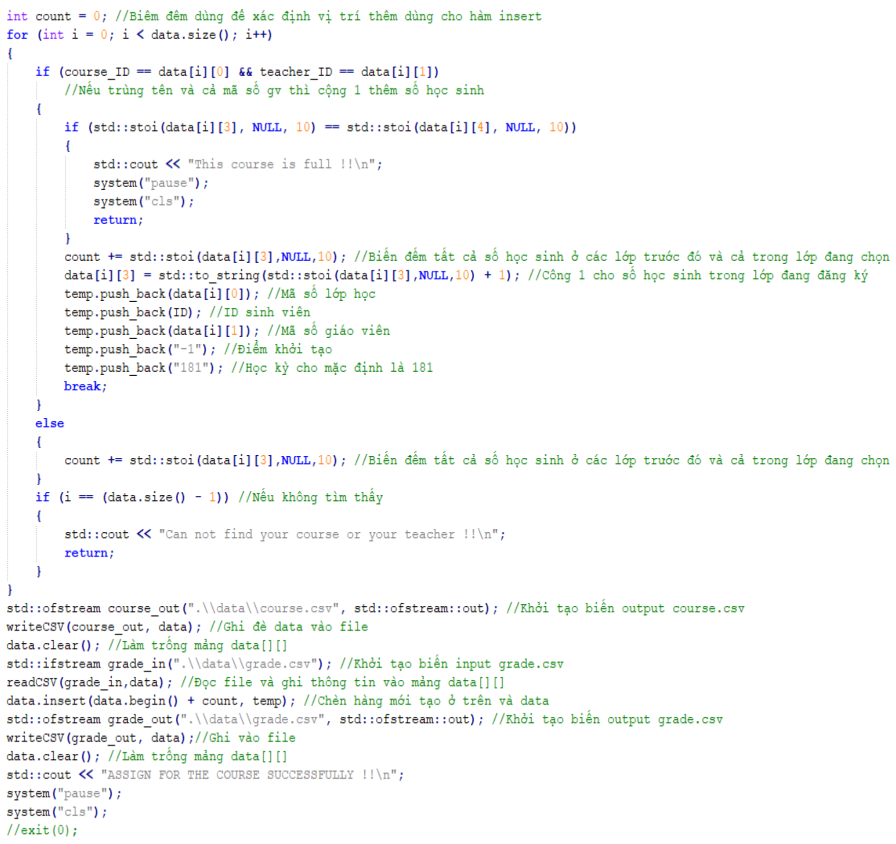
8. Use temp vector contain new student information

9. Write data vector to course.csv

10. Read grade.csv into data vector

11. Insert temp vector into data vector

12. Write data vector to grade.csv

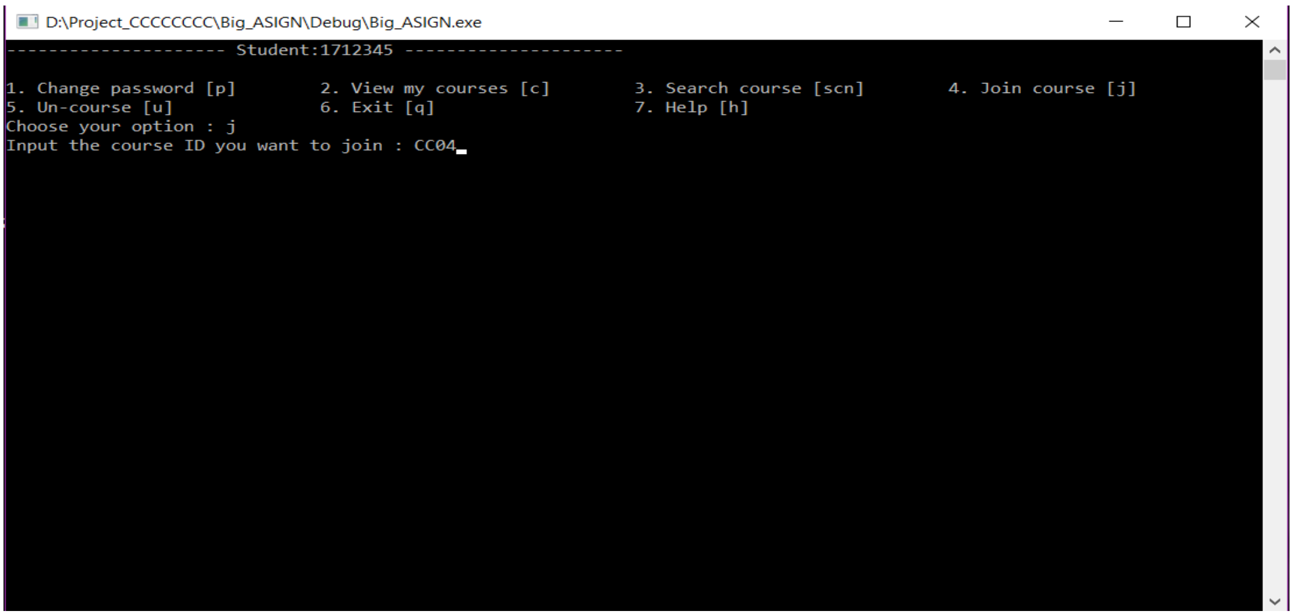


```cpp
int count = 0; //Biêm đêm dùng để xác định vị trí thêm dùng cho hàm insert
for (int i = 0; i < data.size(); i++)
{
    if (course_ID == data[i][0] && teacher_ID == data[i][1])
        //Nếu trùng tên và cả mã số gv thì cộng 1 thêm số học sinh
    {
        if (std::stoi(data[i][3], NULL, 10) == std::stoi(data[i][4], NULL, 10))
        {
            std::cout << "This course is full !!\n";
            system("pause");
            system("cls");
            return;
        }
        count += std::stoi(data[i][3],NULL,10); //Biến đếm tất cả số học sinh ở các lớp trước đó và cả trong lớp đang chọn
        data[i][3] = std::to_string(std::stoi(data[i][3],NULL,10) + 1); //Cộng 1 cho số học sinh trong lớp đang đăng ký
        temp.push_back(data[i][0]); //Mã số lớp học
        temp.push_back(ID); //ID sinh viên
        temp.push_back(data[i][1]); //Mã số giáo viên
        temp.push_back("-1"); //Điểm khởi tạo
        temp.push_back("181"); //Học kỳ cho mặc định là 181
        break;
    }
    else
    {
        count += std::stoi(data[i][3],NULL,10); //Biến đếm tất cả số học sinh ở các lớp trước đó và cả trong lớp đang chọn
    }
    if (i == (data.size() - 1)) //Nếu không tìm thấy
    {
        std::cout << "Can not find your course or your teacher !!\n";
        return;
    }
}
std::ofstream course_out(".\\data\\course.csv", std::ofstream::out); //Khởi tạo biến output course.csv
writeCSV(course_out, data); //Ghi đè data vào file
data.clear(); //Làm trống mảng data[][]
std::ifstream grade_in(".\\data\\grade.csv"); //Khởi tạo biến input grade.csv
readCSV(grade_in,data); //Đọc file và ghi thông tin vào mảng data[][]
data.insert(data.begin() + count, temp); //Chèn hàng mới tạo ở trên và data
std::ofstream grade_out(".\\data\\grade.csv", std::ofstream::out); //Khởi tạo biến output grade.csv
writeCSV(grade_out, data);//Ghi vào file
data.clear(); //Làm trống mảng data[][]
std::cout << "ASSIGN FOR THE COURSE SUCCESSFULLY !!\n";
system("pause");
system("cls");
//exit(0);
```

13.

User

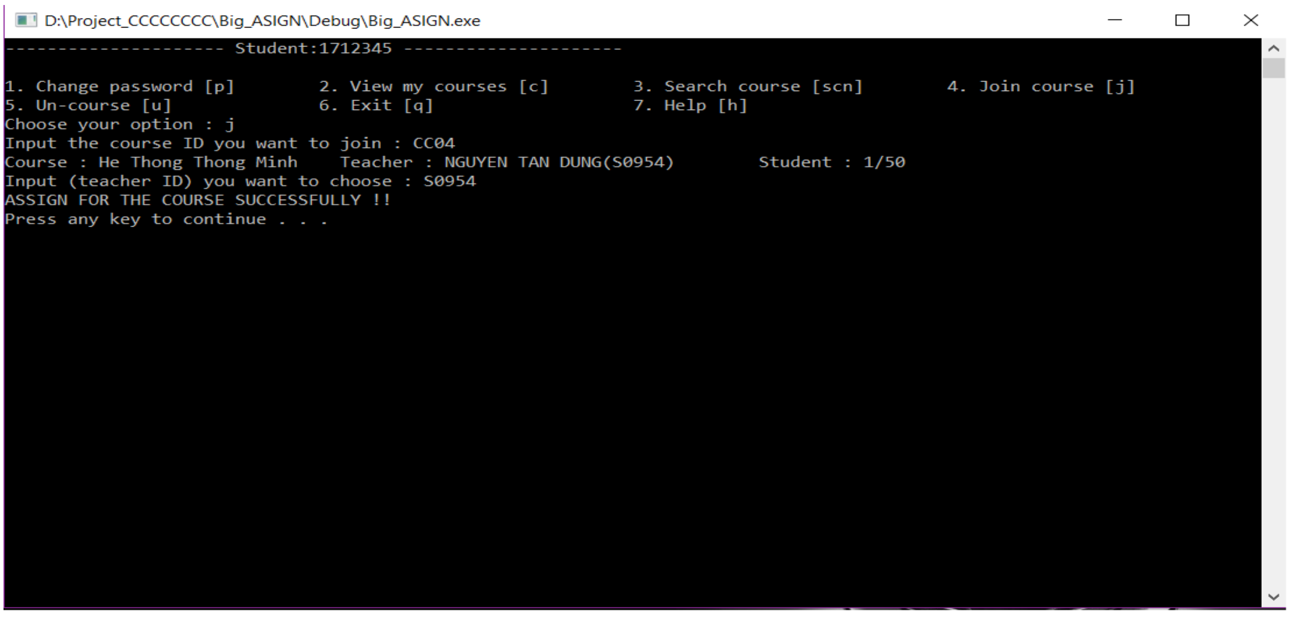
1. Read course ID from keyboard

2. 

```
--------------------- Student:1712345 ---------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option : j
Input the course ID you want to join : CC04_
```

3. Show all courses that have same ID

4. Read teacher ID who you want to choose from keyboard



D:\Project_CCCCCCCC\Big_ASIGN\Debug\Big_ASIGN.exe

```
-------------------- Student:1712345 --------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option : j
Input the course ID you want to join : CC04
Course : He Thong Thong Minh     Teacher : NGUYEN TAN DUNG(S0954)        Student : 1/50
Input (teacher ID) you want to choose : S0954
ASSIGN FOR THE COURSE SUCCESSFULLY !!
Press any key to continue . . .
```

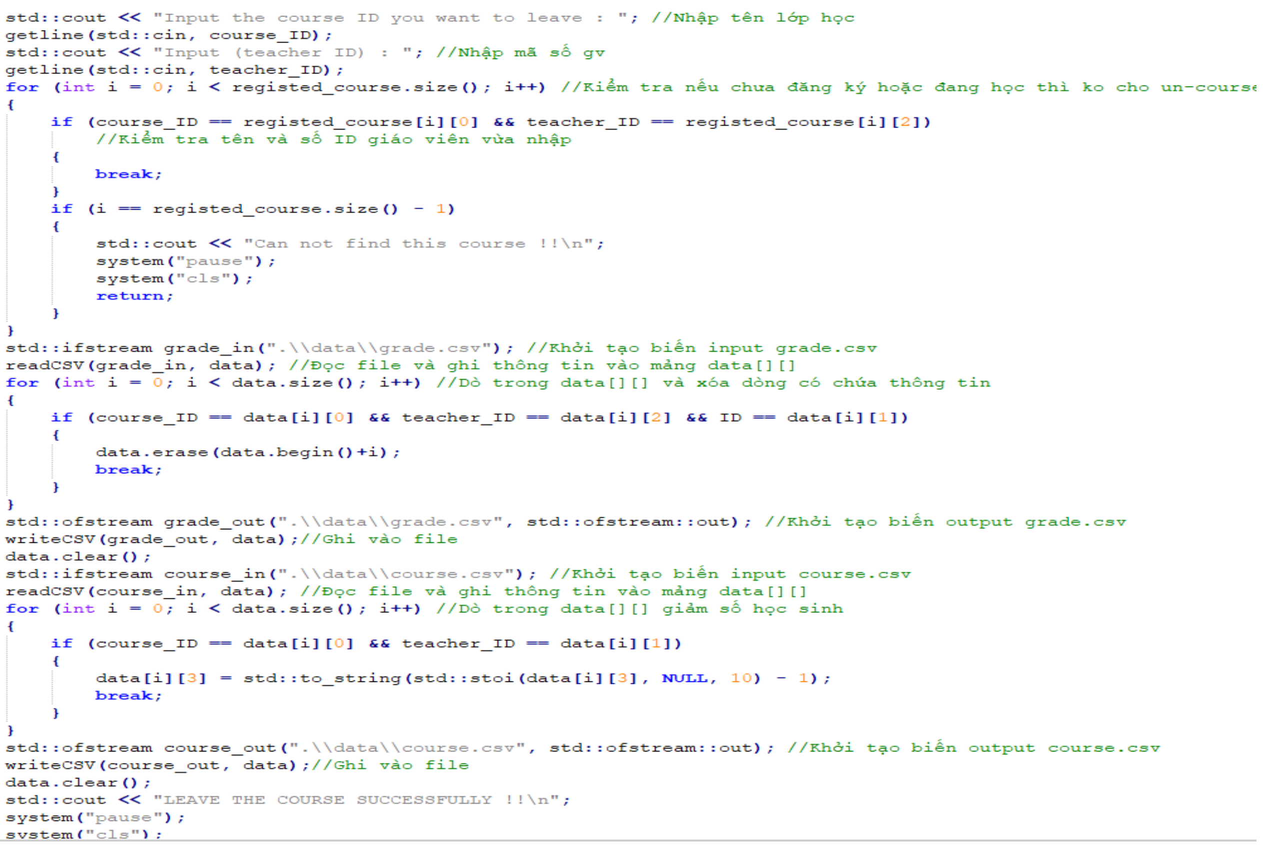
5.

6. Assign the course successfully

2.5.6 Un-course

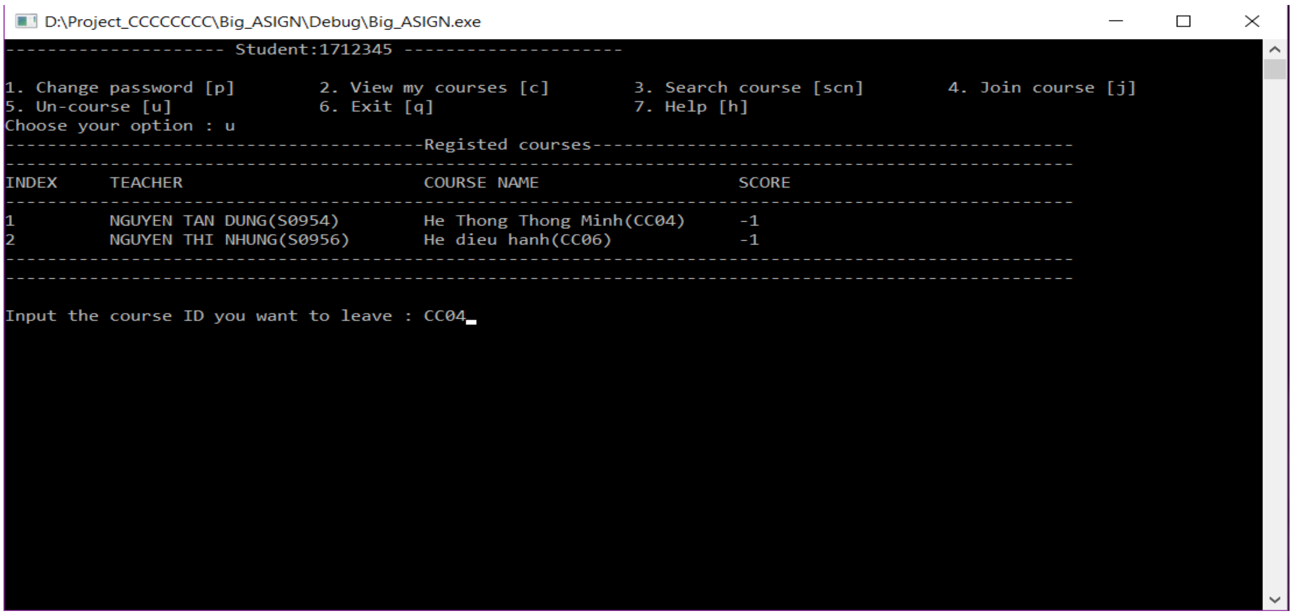
Code

1. Show all the course registered
2. Read course ID from keyboard
3. Read teacher ID from keyboard
4. Check if this course was registered or being studied?
5. Read grade.csv into data vector
6. Loop through data vector and erase row suitable
7. Write data vector to grade.csv
8. Read course.csv into data vector
9. Decrease student number in this course
10. Write data vector to course.csv



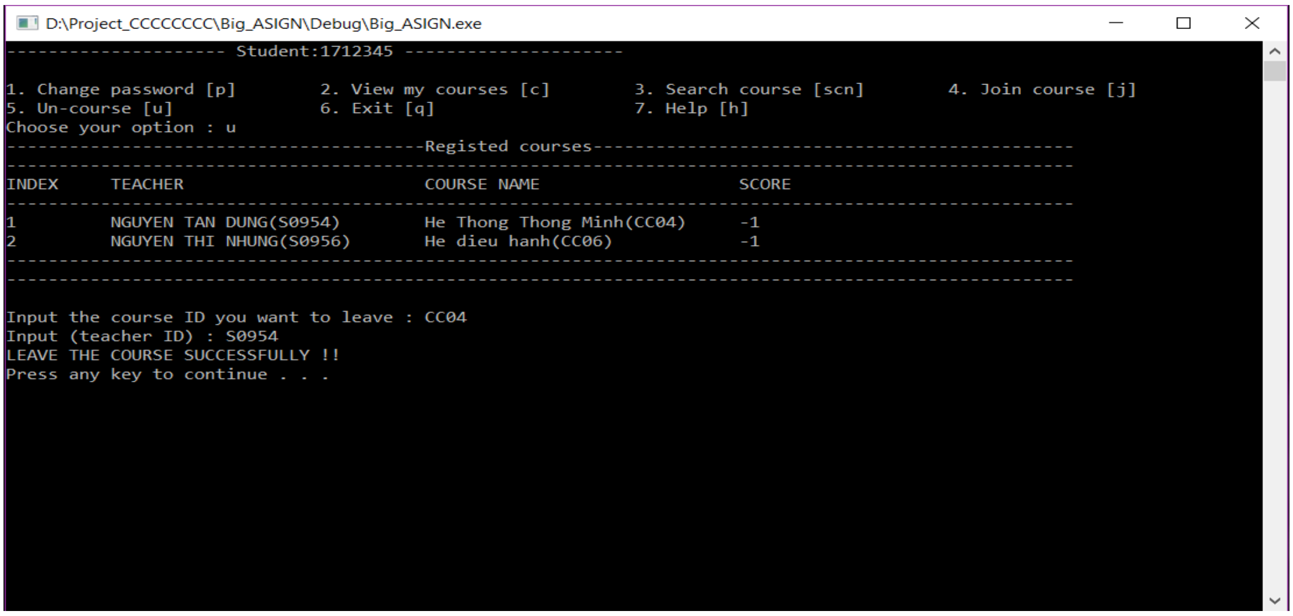
11.
```
std::cout << "Input the course ID you want to leave : "; //Nhập tên lớp học
getline(std::cin, course_ID);
std::cout << "Input (teacher ID) : "; //Nhập mã số gv
getline(std::cin, teacher_ID);
for (int i = 0; i < registed_course.size(); i++) //Kiểm tra nếu chưa đăng ký hoặc đang học thì ko cho un-cours
{
    if (course_ID == registed_course[i][0] && teacher_ID == registed_course[i][2])
        //Kiểm tra tên và số ID giáo viên vừa nhập
    {
        break;
    }
    if (i == registed_course.size() - 1)
    {
        std::cout << "Can not find this course !!\n";
        system("pause");
        system("cls");
        return;
    }
}
std::ifstream grade_in(".\\data\\grade.csv"); //Khởi tạo biến input grade.csv
readCSV(grade_in, data); //Đọc file và ghi thông tin vào mảng data[][]
for (int i = 0; i < data.size(); i++) //Dò trong data[][] và xóa dòng có chứa thông tin
{
    if (course_ID == data[i][0] && teacher_ID == data[i][2] && ID == data[i][1])
    {
        data.erase(data.begin()+i);
        break;
    }
}
std::ofstream grade_out(".\\data\\grade.csv", std::ofstream::out); //Khởi tạo biến output grade.csv
writeCSV(grade_out, data);//Ghi vào file
data.clear();
std::ifstream course_in(".\\data\\course.csv"); //Khởi tạo biến input course.csv
readCSV(course_in, data); //Đọc file và ghi thông tin vào mảng data[][]
for (int i = 0; i < data.size(); i++) //Dò trong data[][] giảm số học sinh
{
    if (course_ID == data[i][0] && teacher_ID == data[i][1])
    {
        data[i][3] = std::to_string(std::stoi(data[i][3], NULL, 10) - 1);
        break;
    }
}
std::ofstream course_out(".\\data\\course.csv", std::ofstream::out); //Khởi tạo biến output course.csv
writeCSV(course_out, data);//Ghi vào file
data.clear();
std::cout << "LEAVE THE COURSE SUCCESSFULLY !!\n";
system("pause");
system("cls");
```

User

1. Show all the courses you registered
2. Read course ID which you want to un-course form keyboard
3. 

```
-------------------- Student:1712345 --------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option : u
----------------------------------------Registed courses----------------------------------------
-------------------------------------------------------------------------------------------------
INDEX     TEACHER                        COURSE NAME                    SCORE
-------------------------------------------------------------------------------------------------
1         NGUYEN TAN DUNG(S0954)         He Thong Thong Minh(CC04)      -1
2         NGUYEN THI NHUNG(S0956)        He dieu hanh(CC06)             -1
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------

Input the course ID you want to leave : CC04
```

4. Read teacher ID who you want to choose from keyboard
5. Un-course successfully

6. 

D:\Project_CCCCCCCC\Big_ASIGN\Debug\Big_ASIGN.exe

```
-------------------- Student:1712345 --------------------

1. Change password [p]        2. View my courses [c]        3. Search course [scn]        4. Join course [j]
5. Un-course [u]              6. Exit [q]                   7. Help [h]
Choose your option : u
---------------------------------------Registed courses---------------------------------------------
----------------------------------------------------------------------------------------------------
INDEX     TEACHER                       COURSE NAME                  SCORE
----------------------------------------------------------------------------------------------------
1         NGUYEN TAN DUNG(S0954)        He Thong Thong Minh(CC04)     -1
2         NGUYEN THI NHUNG(S0956)       He dieu hanh(CC06)            -1
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------

Input the course ID you want to leave : CC04
Input (teacher ID) : S0954
LEAVE THE COURSE SUCCESSFULLY !!
Press any key to continue . . .
```