**Contents**

1 Specification………………………………………………………………….2

2 Analysis………………………………………………………………………..3

2.1 Input……………………………………………………………………..4

2.2 Output……………………………………………………………………5

3 Algorithm design……………………………………………………………7

3.1 Initialization……………………………………………………………8

3.2 Get information……………………………………………………….8

3.2.1 From the keyboard…………………………………………..9

3.3 Login……………………………………………………………………..9

3.3.1 For administrator……………………………………………..9

3.3.2 For students……………………………………………………9

3.4 Operation chosen…………………………………………………….9

3.4.1 Realize functions set for administrator……………..10

3.4.2 Realize functions set for students…………………….10

3.5 Read and write files……………………………………………….10

3.6 Branch………………………………………………………………….10

4 Test and debug……………………………………………………………11

4.1 Administrator………………………………………………………..11

4.2 Students……………………………………………………………….22

**1 Specification**

1.1 The aim of this project is to design a management system for a university to provide the administrator and students with functionality to use. The system users are differentiated with three kinds of identity which are administrator, undergraduate and graduate students. In addition, this system should be able to store the information of all the students and their courses respectively. To give a further explication, the administrator has the higher authority to edit the course information and student information including:

(1) Browse, modify and delete the course information and student information (the modification of students’ selections of certain courses is excluded)

(2) Search and display specific information as required

(3) Register a new student information and ensure the student ID is not repeated

(4) Add a new course information and similarly ensure the course name and ID are unique.

While students have the ability to search for their own information (personal information and past marks) and choose new courses for next semester or the courses which need to be relearned (some courses are only available for graduate students)

1.2 Firstly, four head files should be should be constructed to build the basic functions of the program. They are specifically summarized as the following statements

(1) “stdc++”: a header file containing all the C++ header files that could be used for convenience.

(2) “Course.h”: to define a Course class to record courses’ information which are classified into private for students to refer and public for browsing, and achieve the previously mentioned functions of courses in section 1 after initialization the member functions.

(3) “Student.h”: to define a Student class with two self-defined sub classes undergraduate and graduate. The class aims to store the students’ information which are divided into protected for them to research and public for browsing. Furthermore, it implements the functions of students mentioned in part 1 after initialization the member functions.

(4) “global\_path.h”: to define two paths towards certain files to store the information of students and courses respectively.

Second, for the source file designing, the system should primarily provide a surface for the users to choose whether to log in as an administrator or a student. If the user logs into the system as an administrator, a choice will be immediately shown to let the person select his or her operation object, that is students or courses. Subsequently, a more detailed menu will be printed on the screen offering the operation type which are browsing, adding, modifying, deleting and choosing courses. Moreover, a slogan should be displayed every time when the user finishes a single operation to continue this system until the “logout” button is pressed.

1.3 The execution of this program will be stuck if an input error happens. Being illegal inputs may be defined as follows:

Input the number that not provided in the menu or the form of the input is not included in the desired variety.

The wrong input of password will lead to the fail of logging in, and the program will back to the main menu, as a result, users should carry out the log in process again.

When searching for students or courses which do not exist, no information will be displayed. Similarly, the system should display appropriate output. For instance, when the administrator browses all courses, there should be a chat printed on screen showing the information of a certain one.

1.4 The program exits.

**2 Analysis**

In the following chapters, we will analyze the input, output and additional requirements or constraints of the solution.

Input:

Firstly, it is necessary to give users a main list to let user choose which operation they want to choose. The list should include:

1. Administrator
2. Student
3. Logout

Users need to enter different numbers to enter different branches or exit the program. Users need to enter different numbers to enter different branches or exit the program. When the user selects 1, he will enter the administrator interface for administrator operation; when the user selects 2, he will enter the student interface and enter 0 to exit the program.

When the user enters the administrator interface, the program provides the following options.

1.Add a student

2.Browse all students’ information;

3.Search a student;

4.Browse a student information;

5.Delete a student;

6.Add a course;

7.Browse all courses information;

8.Search a course;

9.Browse a course information;

10.Delete a course;

0.Logout;

This option page is to provide function selection for administrators. When input 1, he can add students. Requirements will appear on the screen. Users can add students by entering new ID, student name, student level and password as required. The data is stored in a file. Input the number 2, the user will be able to browse all the existing student information, but can't modify the student information; when input the number 3, the user will enter the search interface, and can find the student by entering the student ID; when input the number 4, the user can modify the student information, but can't modify the student selection results; when input the number 5, the user can delete The student information stored in the file; when the number 6 is entered, the user can add courses for students to choose; when the number 7 is entered, the user can view all courses available for students to choose, but cannot make any changes to the courses; when the user enters the number 8, he can search for a course; when the user enters the number 9, he can carry out the course information Modify; when the user enters the number 10, the existing course can be deleted. Enter the number 10 and the user will exit the program.

If the user enters 2 in the first selection interface, he will enter the student interface, and then he needs to enter the grade, student number, type name and password. The program then provides the following options.

1. Choice a new course
2. Browse my selected courses
3. Browse my personal information
4. Logout

If the user enters the number 1, he can add a course (the course is provided by the administrator); when the user enters the number 2, he can browse all the courses he has selected; if he enters the number 3, he can browse his information, such as his ID, name, grade and password; if the user enters the number 0, he will exit the program.

Output:

The screen will print out a general catalog, and the administrator or student will select what they want to do from the following three options

Welcome to the University Student Management System

Please select your identity

1 Administrator

2 Student

0 Logout

Each legal input corresponds to a function to be selected by the user. When "0" is selected, the program will exit. Similarly, two other menus will be displayed on the screen. Students and administrators determine what they want to do next when the activity is complete. The following is the administrator's output menu:

Welcome administrator:“\*\*\*\*\*”

Please select your identity

1.Add a student

2.Browse all students’ information;

3.Search a student;

4.Browse a student information;

5.Delete a student;

6.Add a course;

7.Browse all courses information;

8.Search a course;

9.Browse a course information;

10.Delete a course;

0.Logout;

Enter your choice:

Each administrator's input will correspond to a specific output. When "0" is entered, the program will exit.

In the student's program branch, the screen will display such output.

Welcome student:“\*\*\*\*\*”

Please select your identity

1 Choice a new course

2 Browse my selected courses

3 Browse my personal information

1. Logout

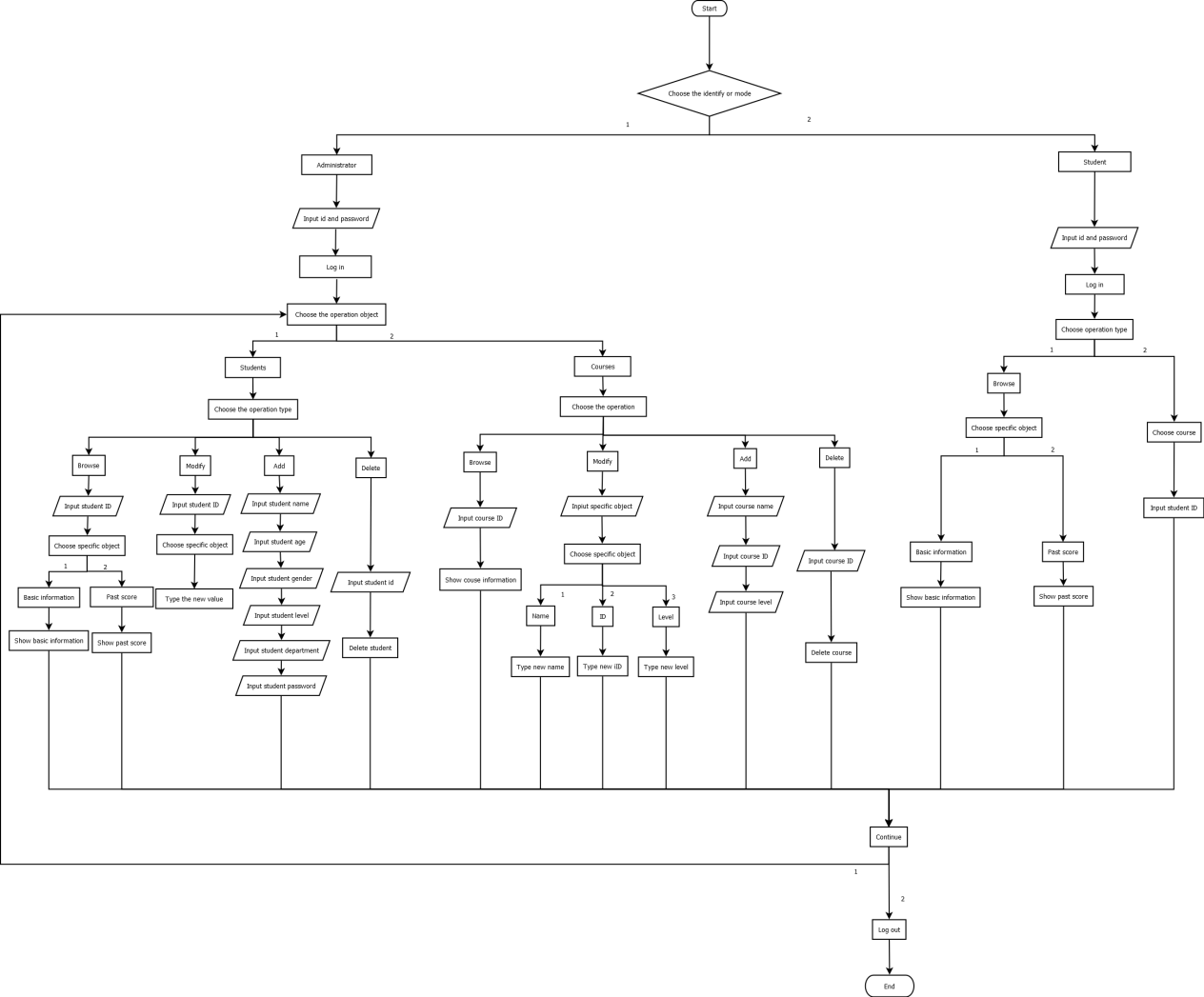
Enter your choice:

Select the program according to the prompts. Different options correspond to different functions. Select according to your own needs. When you enter "0", you will exit the program.

Additional requirements or constraints on the solution

During function selection, only the options given on the screen can be entered, and other numbers are invalid; during ID input, the ID size within the valid range shall be entered. The program should be able to provide the output of each option.

**3. Algorithm design:**



The first thing is separating the task into parts by means of the functions or characters. Thus, we designed Course.h and Student.h to record their information and achieve specific functions, respectively. And we also defined stdc++.h which is an implementation file for a precompiled header. Then we added test.cpp to run the tests. In test.cpp file, we first initialized the variables for later use, and realized the functions for this project.

3.1 Initialization

We started the program with the initialization. To begin with, we defined class Student and class Course with constructors that initialize the detailed information. Next, we set up student and course lists in Student.h and Course.h files, respectively and set both lists to 0. Then we set up lists for id numbers, one stores the used id numbers, and another stores the id numbers that haven’t been used.

In test.cpp file, we first set up *void people\_initialize()*, and *void courses\_initialize()*, in these two functions, we designed to let the program read from files. Take people initialization as an example, we applied json to realize the reading function. More specifically, we used a finished json object neamed students\_json to handle the data file and access data members. The reason we applied json is that it allow us to store data dynamically. Then we applied vector to store all students’ id numbers and their past results, and pass them in different types of varibles. For initializing the courses, the logic is the same.

3.2 Get information

After initialization, we designed to first get all the information form users and courses for later use. Take ‘student’ as an example, we applied set functions to validate their arguments and get member functions to present the data in different forms for both students and administrators. After that, we set up *get\_info()* functions to present the information. Also for students, we set up *get\_score()* function and *choose\_course()* function. To emphasize, the students should only choose the courses that are allowed for them. Thus, we use if statement to compare the selected course level and the student level, if it is the same, then the operation is allowed, otherwise, not allowed. As for getting information of courses, the strategies are basically alike.

3.2.1 Get from the keyboard

Then, in test.cpp file, we created functions called *string read\_a\_string()* to read from the keyboard and returns the result. Also we created a function to convert a string to digits, which is *int string\_to\_int(string str)*. In this function, *atoi(str. stringName())* is applied to convert the type.

3.3 Login

Now, this part is in fact the start of the program. We first initialized some variables for various datatype. Next, we applied do…while statement with nested if statement inside to get users’ identities and guide them. The program will show the choice for identity choosing. 1 represents administrator, and 2 represents student.

3.3.1 Login for admin

Firstly, to compare the input and the set password, we applied *strcmp(string1, string2)*, if string1 equals to string2, the function returns 0. Thus, we can compare the return value with 0 to verify if a user input the correct password and then the user can log in as an administrator.

3.3.2 Login for student

After choosing student mode, their student id will be checked to see if it is among the set range. We use if…else if statement to perform three situations, illegal student id, legal id number but no corresponding person, and the correct one. Then we need *bool* *check\_student\_id\_int\_no\_use(int id\_int)* function to make sure the id numbers are not deplicated.For the correct situation, we then verify the password just like what we did in admin part.

3.4 Operation chosen

After logging in, we called *branch(string id)* function to lead people with different identities to different operation panels. We applied *int find\_operation\_type()* to detect and return the results. To be informed, this branch function will be explicated at the end.

3.4.1 Realize functions set for admin

For administrators, they can browse, modify, add, delete both courses and students, and they can be divided into two parts by means of how we realized them. One part takes in the browsing and modifying functions. The program leads users to type in the specific object they want to browse or change of a student or a course. After that, there are different branches, for example, if one chooses to modify the name of a student, then we created a specific function for this operation. Take modifying student id number as an example, we initialized string id\_new and int\_id\_new\_int to store the data from the input by *read\_a\_string()* function mentioned before. Then we used if statement to check whether the id is within scope and ensure that id numbers are not duplicated, this strategy is also used for modifying course id number.

The other part includes adding and deleting functions, this part requires to insert the new data into the responding lists by using *insert* function. As for deleting operation, we set the selected id numbers of student or course to NULL, and release the memory.

3.4.2 Realize functions set for students

For students, they can browse their information and choose courses. Browsing function is already realized in administrator part. To choose courses, we set up *choose\_course()* function in Student.h file, and in this function, we applied for statement to loop through all the students with two if statements implemented the verification for the course level and courses existences, respectively. finally, in test.cpp file, calling choose\_course() function by a specific student object in *student\_choose\_course(string id)* function to realize choosing courses.

3.5 Read and write files

This part is the most challenging. There are students and courses needed to be stored, we first looped through all the students or courses, and then use created json objects to receive data via *get* member functions

3.6. Branch

This is the last part but it is not the end of the program. We applied do…while statement with an if…else statement inside to operate three functions, continue, log-in with other account, and log-out. Continue means run the program with present identity. Thus, we applied *continue* statementwhich will automatically jump to the while statement which leads to the operation chosen part. And the rest will be all the same as previous. For changing account, we called *login()* function to make the user log in with new identity. For logging out, the program will simply stop.

**4 Test and debug**

1. Administrator ：

When the user chooses 1 in the main list and input the name and password, the figure 1 will be shown.

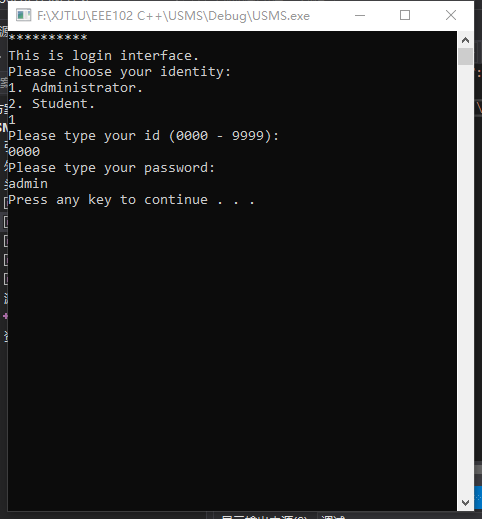


Figure1: Administrator login

Figure 2 shows the result, when the administrator wants to Browse a student basic information.

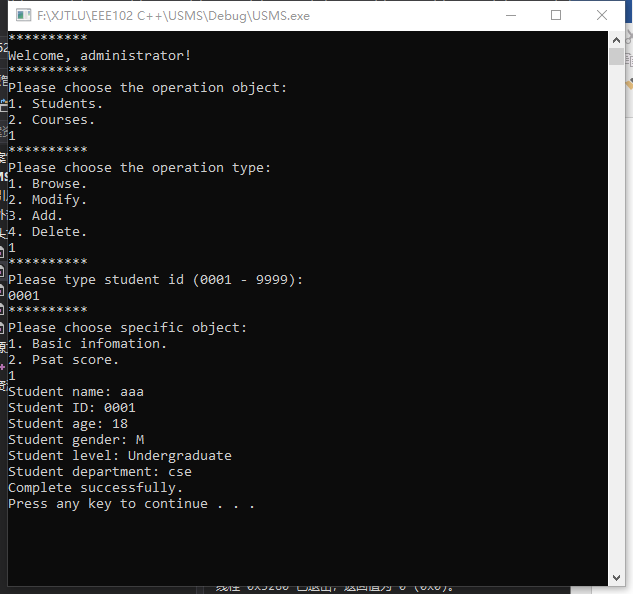


Figure 2: Browse a student Basic information

After completing an operation, the system asks the user whether to continue, log in under a different identity, or log out. We chose to continue. The result is shown in figure 3.

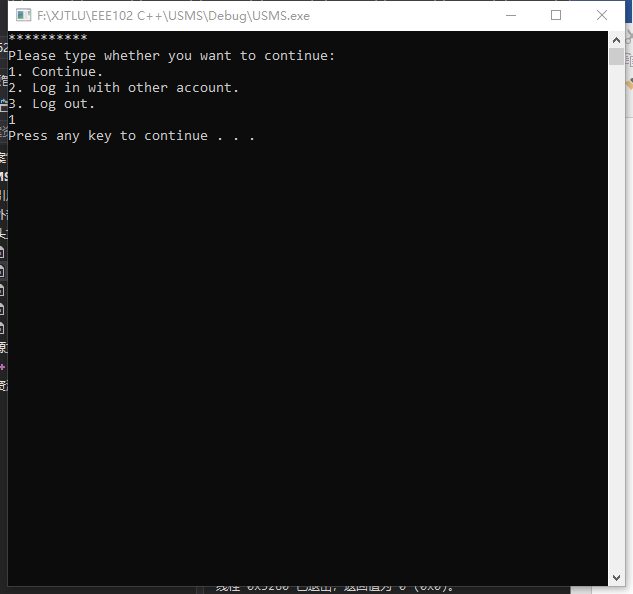


Figure 3: Continue to operate

We continue to modify student information. Enter the student number 9999 but the student does not exist. The result is shown in figure 4.

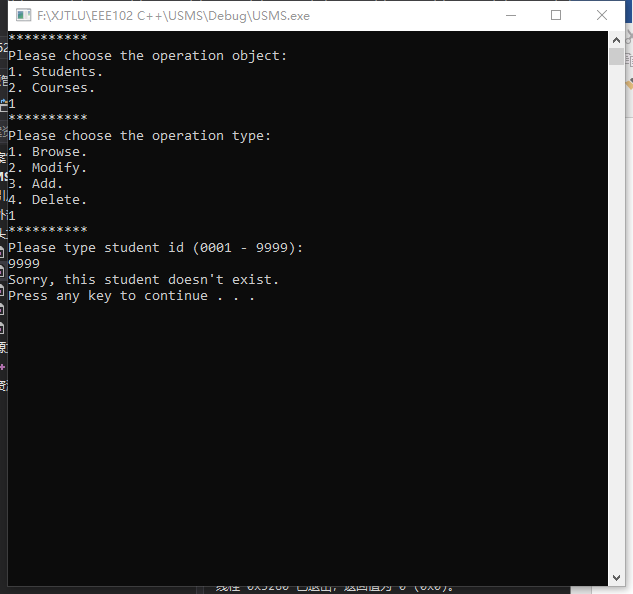
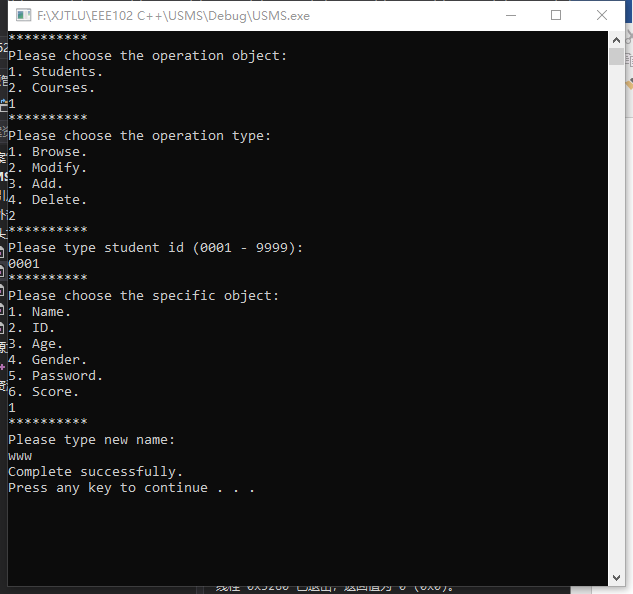
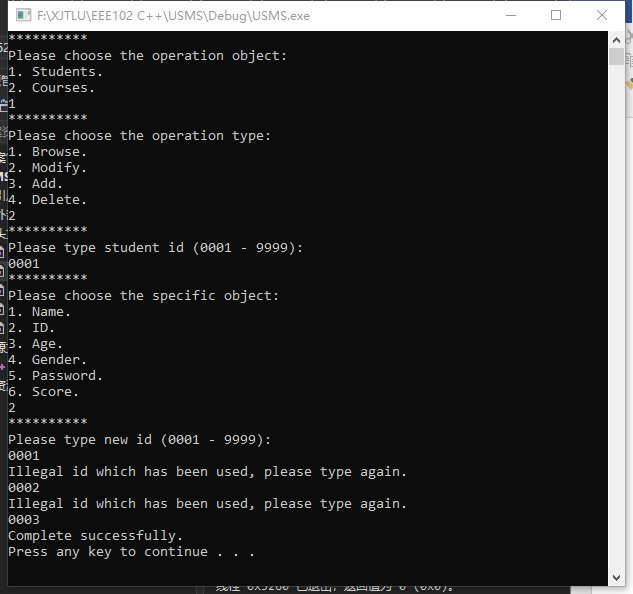
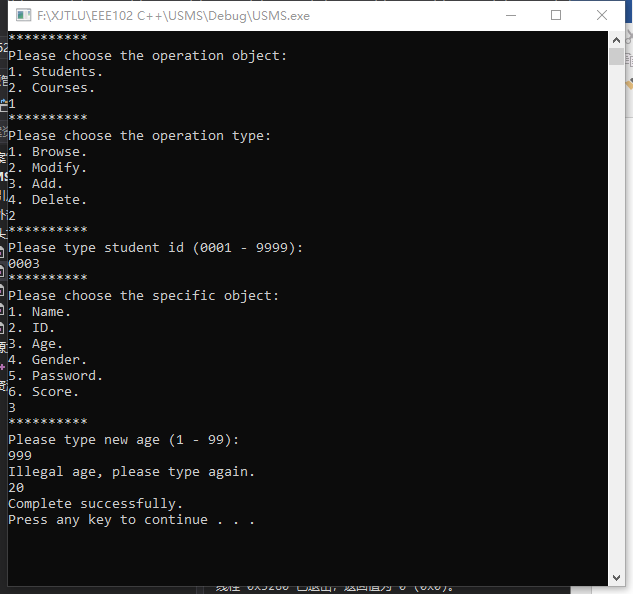
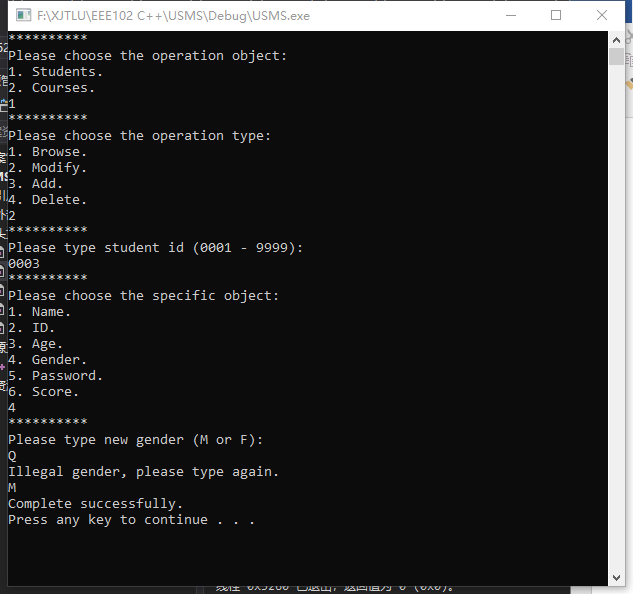
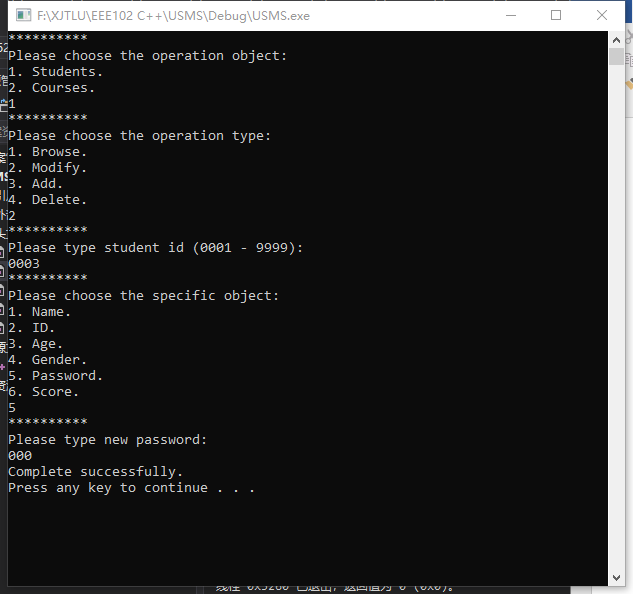


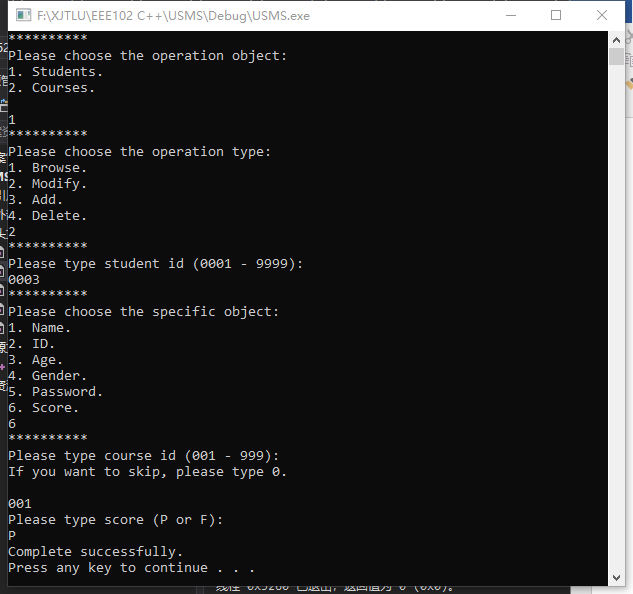
Figure 4: Enter a student ID 9999, but this student does not exist.

Enter the correct student number and we can modify the student's information.  Figure 5: modify the student's name

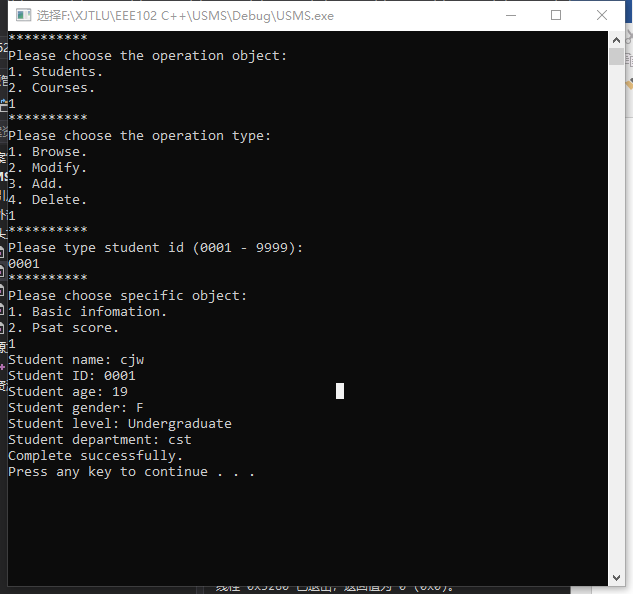
 Figure 6: modify the student's ID Figure 7: modify the student's age

 Figure 8: modify the student's gender

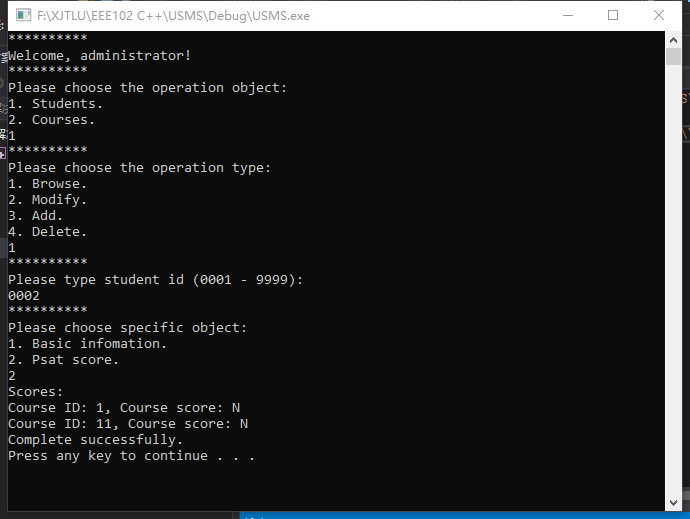
 Figure 9: modify the student's password

 Figure 10: modify the student's score

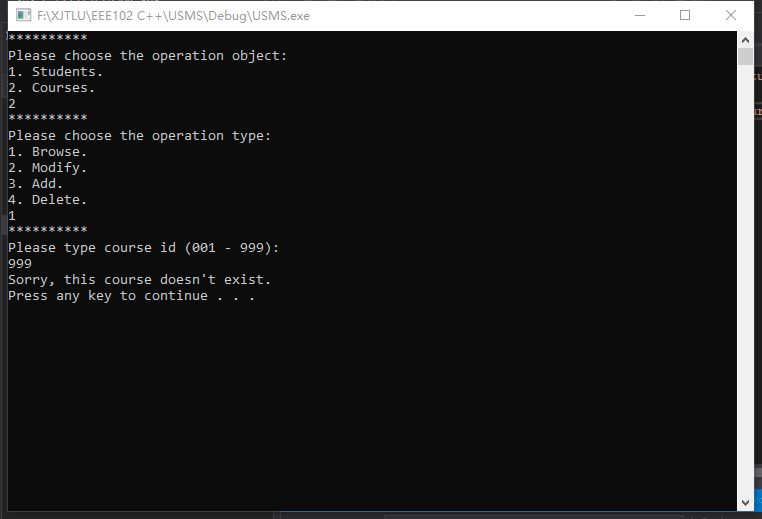
We can view the modified student information, as shown in figure 11.

 Figure 11: modified student information

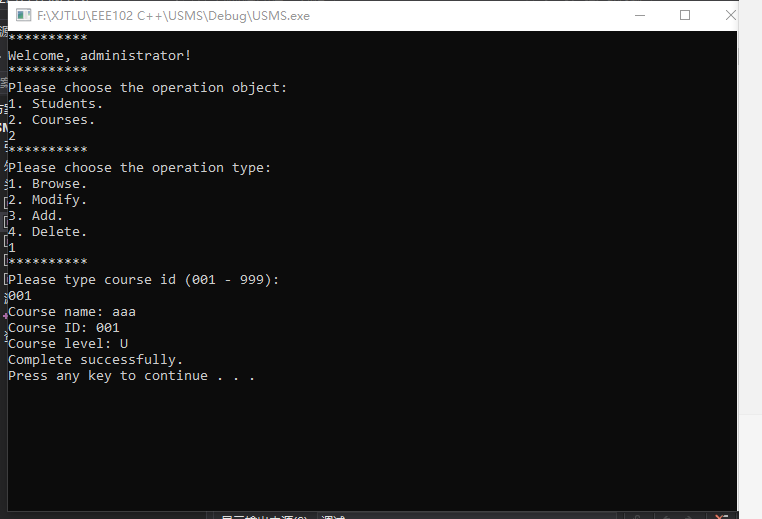
We can view the students' scores, as shown in figure 12.

 Figure 12: View the students' scores

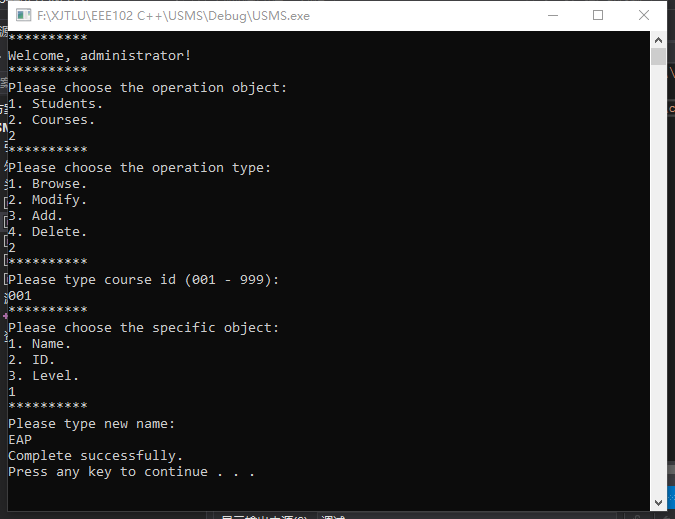
Now let's manipulate the course information. If you enter the wrong course ID, the system will remind the user that the course does not exist. as shown in figure 13.

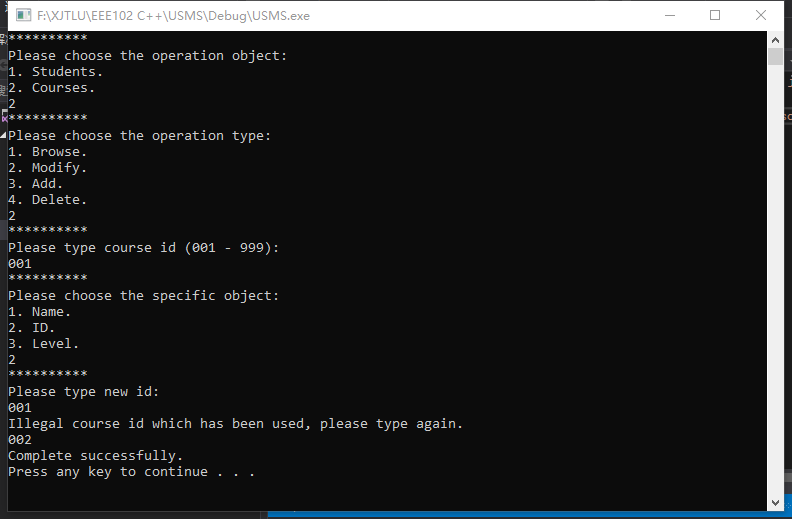
 Figure 13: Course does not exist.

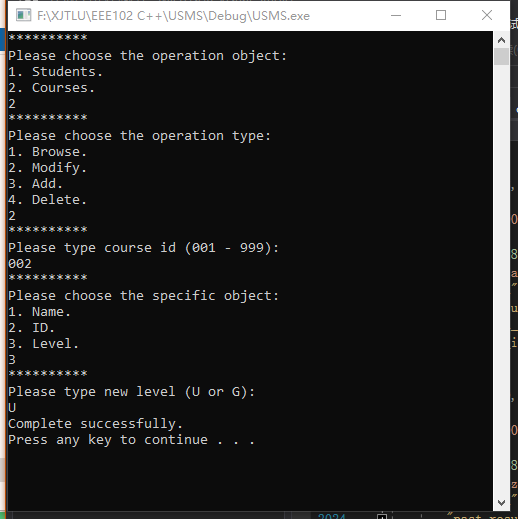
After entering the correct ID, you can view the course information. See figure 14.

 Figure 13: Browse course information.

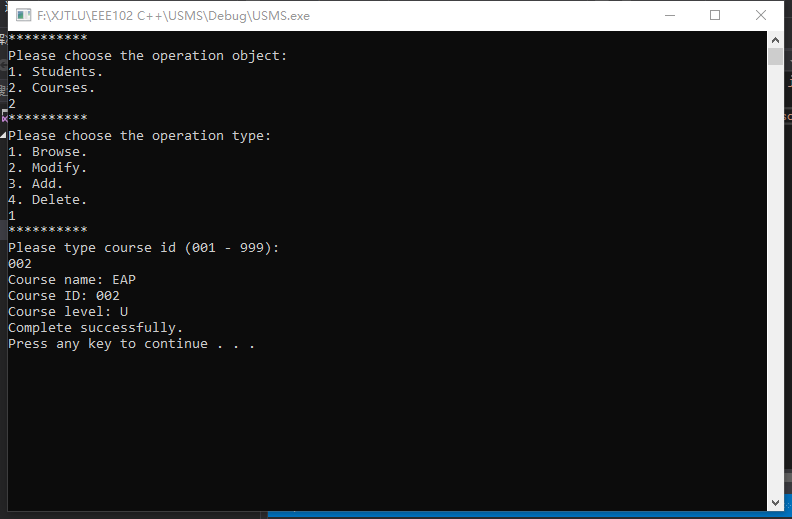
Next, we will modify the course information.

 Figure 14: Modify course name.

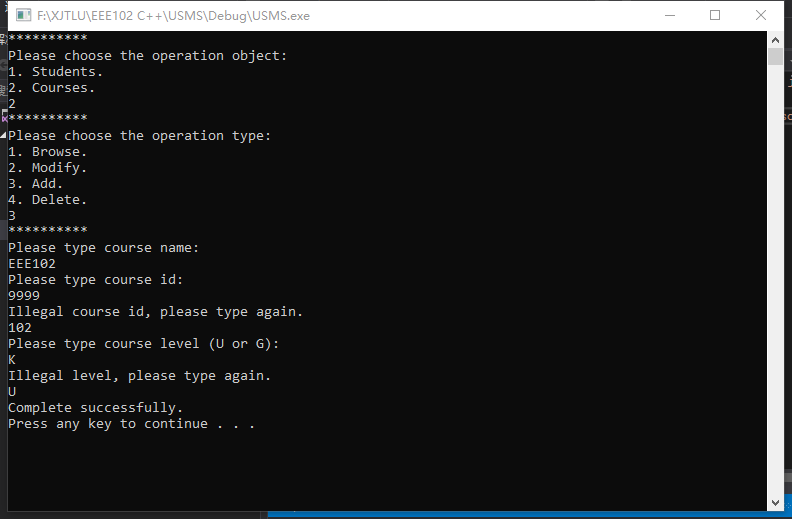
 Figure 15: Modify course ID.

 Figure 16: Modify course level.

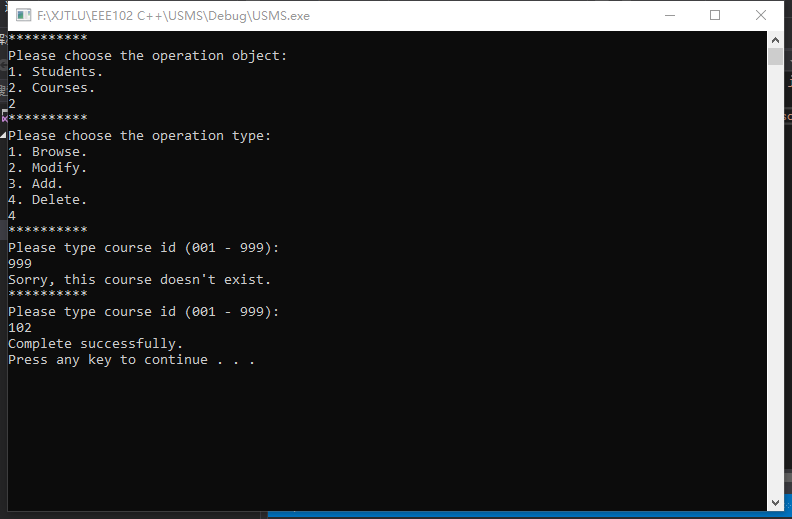
We can review the revised course information again. As shown in figure 17.

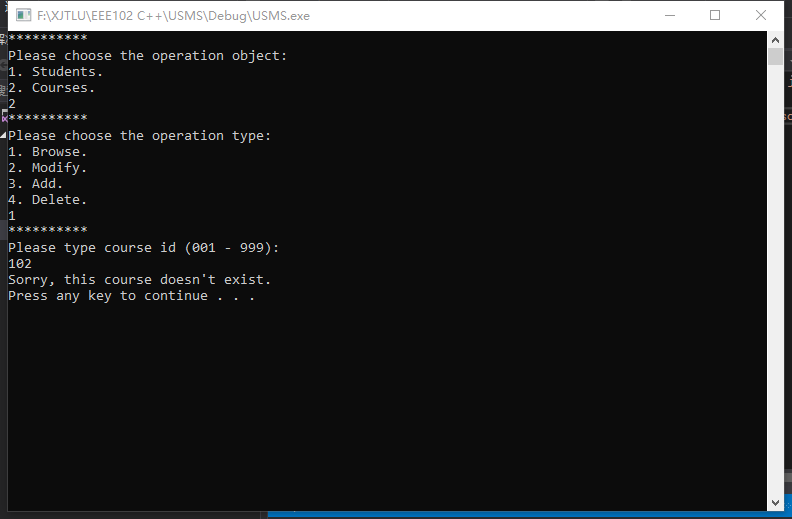
Figure 17: Browse revised course information

Now let's add a course, as shown in figure 18.

 Figure 18: Add a course.

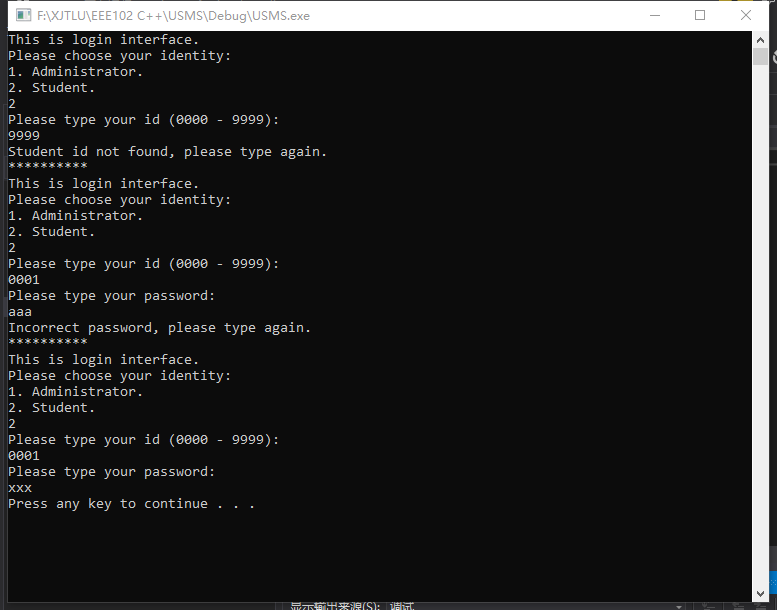
We can try to delete the EEE102 course we just added, as shown in figure 19. Moreover, we searched the course again and found that it did not exist, indicating that the deletion was successful, as shown in figure 20.

 Figure 19: Delete course EEE102.

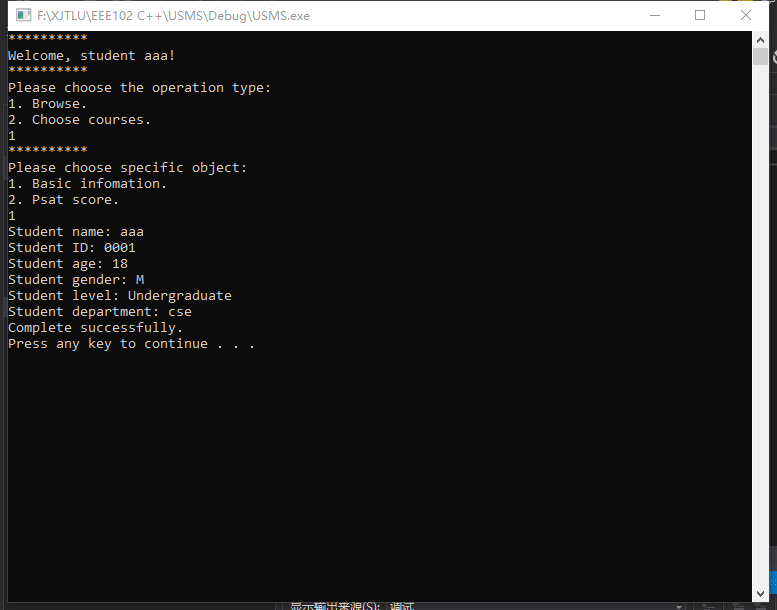
 Figure 20: Delete successfully.

1. Student:

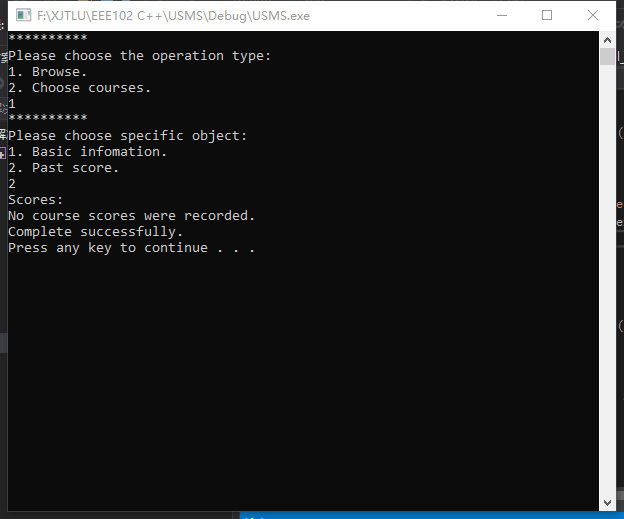
Log in as a student. You cannot log in with the wrong ID or password. As shown in figure 21.

 Figure 21: Login as a student.

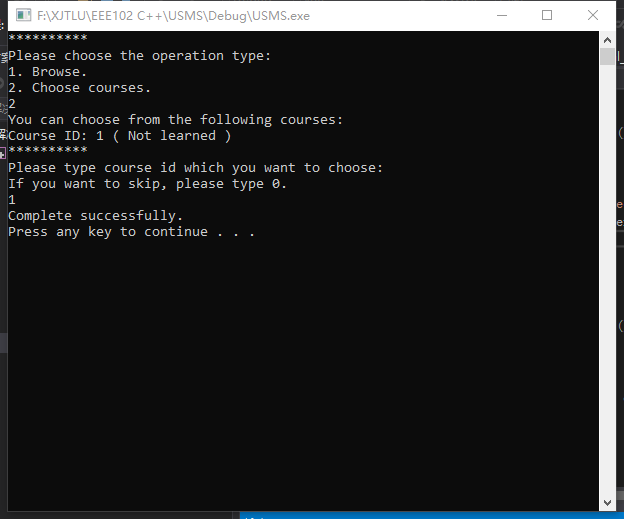
Students can select 1 to browse their personal information, as shown in figure 22.

 Figure 22: Browse basic information.

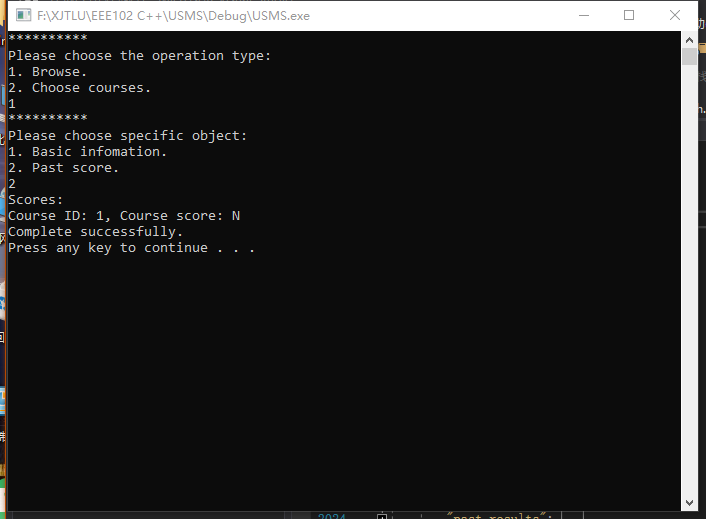
Students can also view their own course scores, as shown in figure 23.

 Figure 23: Browse courses score.

Students can select 2 to enter the course selection interface, and choose courses he hasn't learned, as shown in figure 24.

 Figure 24: Choose courses.

After selecting the course, check the course score again, showing that he has a course and "N" means he has not studied it. As shown in figure 25.

 Figure 24: Choose course successfully.

Finally, the user can log out of the system, as shown in figures 25 and 26.

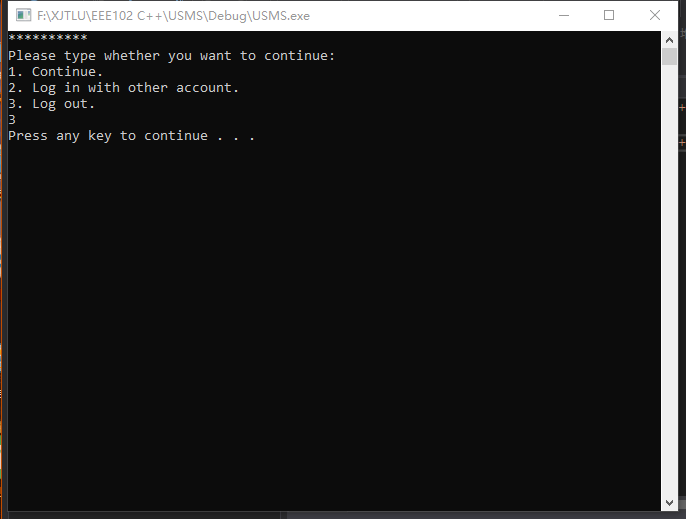
 Figure 25: Logout.

 Figure 24: Logout successfully.