

ADA Report

Class No: ENG2002_20211_E

Team No: 10

Group Member:

Fan Zhendi(20101448d), Hao Jiadong(20084595d), Huang Yufeng(20074902d)

Task No: 0

Abstract

In this assignment, our group is asked to build a class that could represent the complex number and do the addition operation between two complex numbers. A program that lets the user login and another function that could show the main menu are also needed in the whole program. Additionally, we have realized the function which could compute the square root of the complex number in the class.

Introduction

In detail, our group's tasks could be divided below:

For the class construction:

1. Build a class named "complexNum", which could represent a complex number ($x + yi$), where x and y are parameters in double type.
2. Within the defined class, we are told to use a private member variable "errNumber" to check whether the operand is non-zero or not. In the constructor, "errNumber" is set to be false. When the function "addition" is executed, it first checks whether the operand is no-zero, and if so, the "errNumber" will be set to true instead of doing the operation.
3. For other member functions, we should realize the addition operation between two complex numbers, the function that could get the square root of a complex number (for those two functions we decide to assign a new double value as the final output) and so on.

For the user login interface:

4. We should write a function which will let the user login. In this program, there are totally 2 situations:
 1. The username is new. In this case, the program will ask for a password, then store the username-password pair into a file. After that, the user will get into the main program.
 2. The username is in the file. Then the user will be given three chances to enter the correct password. If the user enters the correct password within 3 times, it will get into the main program. If the user can't enter the correct password within 3 times, the whole program will end.

For the menu and main program:

5. A function that could display a menu which includes all the choices. Here we have 5 choices:
 - a. Input two complex numbers.
 - b. Display the two complex numbers you input.
 - c. Do the addition.
 - d. Compute the square root.
 - q. End the program.
6. A main function will call the login and menu functions.

Methodology

1. Work division

Fan Zhendi	Hao Jiadong	Huang Yufeng
Build the class "complexNum" and realize all of its member functions. Give the header file and library for final use.	Create the user login interface including username-password verification and realize different possibilities.	Make the menu for user to choose and write the main function. Testing and debugging.

2. Schedule and steps of developing the project

Date	Progress
Nov.15	Discuss together and divide the task
Nov.16--18	Complete individual parts
Nov.19--20	Integrate all parts together and debugging.
Nov.21-27	Writing report.

3. The details of the developed application

Our application is divided into two parts. The static lib contains the class named `complexNum`. The main program contains function `menu()`, `login()`, and `main()`. This part is to show the specifications of the class defined, and the public/private member functions/variables inside class or the main program. Then explain as far as possible why our group makes such choices for coding.

(1). Detail of class `complexNum`:

A. Features:

- i. Initialize the members when the object is created
- ii. Store new complex number and operand
- iii. Show the values of complex number and operand
- iv. Add complex number and operand together
- v. Show the square roots of the complex number
- vi. Show a complex number in an appropriate format

B. Specifications:

- i. Code:

```

class complexNum
{
public:
    complexNum();
    complexNum addition(complexNum);
    bool read_errNumber();
    double get_x();
    double get_y();
    void set_x(double);
    void set_y(double);
    complexNum square_root();
    void show();
private:
    double x;
    double y;
    bool errNumber;
};

```

C. Detailed explanation:

Private member variables:

```

private:
    double x;
    double y;
    bool errNumber;

```

The variable x of type double is used to store the real part of the complex number.

The variable y of type double is used to store the imaginary part of the complex number.

The variable errNumber of type bool is used to check whether the operand for the addition is non-zero. If the operand is 0+0i, errNumber will be set to true indicating that this complex number pair can't do the addition. Notice that only the first operand's errNumber will be set to true to indicate this complex number pair can't do the addition. For example, if the input complex number pair is 2+3i and 5+7i, only the errNumber of 2+3i will be set to true. Then the program will prompt the user to input another pair of complex numbers to do the addition.

Public member functions:

i.complexNum():

code:

```

complexNum::complexNum()
{
    errNumber = false;
    x = 0;
    y = 0;
}

```

Explanation:

Called when the object is created, which is for initializing the values of the object.

ii. complexNum addition(complexNum):

code:

```

complexNum complexNum::addition(complexNum num)
{
    complexNum outcome;
    if (num.get_x() == 0 && num.get_y() == 0)
        errNumber = true;
    else
    {
        outcome.set_x(x + num.get_x());
        outcome.set_y(y + num.get_y());
    }
    return outcome;
}

```

Explanation:

This function is used to pass in the operand and add the complexNum and operand together. If the complexNum passed in is (0+0i), set errNumber to true instead of doing the addition. Then return a blank complexNum. If the complexNum is not (0+0i), which is a reasonable operand, do the addition and store the answer in complexNum outcome. Then return the outcome.

This determination for passed in complexNum is for helping users know if their input is wrong and remind them to enter the right value.

iii. bool read_errNumber():

Code:

```

bool complexNum::read_errNumber()
{
    return errNumber;
}

```

Explanation:

If this function is called, return the value of errNumber. errNumber is variable to determine whether the operand for this complex number is non-zero.

iv. double get_x() and double get_y():

Code:

```
double complexNum::get_x()
{
    return x;
}
double complexNum::get_y()
{
    return y;
}
```

Explanation:

When one of both functions is called, return the real part number or imaginary part number in double type respectively.

v. void set_x(double a) and void set_y(double a):

Code:

```
void complexNum::set_x(double a)
{
    errNumber = false;
    x = a;
}
void complexNum::set_y(double a)
{
    errNumber = false;
    y = a;
}
```

Explanation:

When one of both function is called and passed in a double number. Set real part number or imaginary part number to the passed in value respectively. Reset errNumber to false(Accurate judgment will be implemented in addition function).

vi. bool read_errNumber():

Code:

```
complexNum complexNum::square_root()
{
    double xx = sqrt((x + sqrt(x * x + y * y)) / 2);
    int sgn;
    if (y < 0)
        sgn = -1;
    else if (y == 0)
        sgn = 0;
    else if (y > 0)
        sgn = 1;
    double yy = sqrt((-x + sqrt(x * x + y * y)) / 2) * sgn;
    complexNum sq_result;
    sq_result.set_x(xx);
    sq_result.set_y(yy);
    return sq_result;
}
```

Explanation:

When this function is called, return the square root of corresponding object.

According to the equation to calculate γ :

$$\gamma = \sqrt{\frac{x + \sqrt{x^2 + y^2}}{2}}$$

We can set `double xx = sqrt((x + sqrt(x * x + y * y)) / 2)`. Where `xx` is stand for γ

According to the equation to calculate δ :

$$\delta = \text{sgn}(y) \sqrt{\frac{-x + \sqrt{x^2 + y^2}}{2}}$$

`double yy = sqrt((-x + sqrt(x * x + y * y)) / 2) * sgn`. Where `yy` is stand for δ .

Here `sgn` is stand for `sgn(y)`, which is the signum function defined as follows,

$$\text{sgn}(y) = \begin{cases} -1, & \text{if } y < 0 \\ 0, & \text{if } y = 0 \\ 1, & \text{if } y > 0 \end{cases}$$

We use following codes to determine the value of `sgn`,

```
int sgn;
if (y < 0)
    sgn = -1;
else if (y == 0)
    sgn = 0;
else if (y > 0)
    sgn = 1;
```

After calculation, define a new `complexNum` object named `sq_result`, set its value to our calculation result. Then return one of the square roots in `complexNum` type.

vii. `void complexNum::show()`:

Code:

```
void complexNum::show()
{
    if (y < 0)
        cout << x << y << 'i';
    else
        cout << x << "+" << y << 'i';
}
```

Explanation:

When this function is called, show the value of `complexNum` in format of `a+bi` or `a-bi` (depend on whether `b` is less than 0).

When `y < 0`, we don't need to add a minus sign because it already has one.

When `y >= 0`, we add a plus sign to show the imaginary part is non-negative.

(2). Detail of function `bool menu()`:

A. Features:

i. Show the interface to ask user to enter a character to determine which function should be used.

ii. If user's input is a, ask user to input new complex number and operand.

iii. If user's input is b, show the two complex numbers.

iv. If user's input is c, do addition and show the result.

v. If user's input is d, output two square roots of the first complex number.

vi. If user's input is q, return true to end the program

vii. If user's input is invalid, show messages to remind user to enter valid character.

B. Details of the function:

i. For feature i:

Code:

```
bool menu()
{
    char choice;//store the user input choice

    cout << "a.Enter one complex numbers and one operand." << endl;
    cout << "b.Display two complex numbers." << endl;
    cout << "c.Addition" << endl;
    cout << "d.Square root of the complex number." << endl;
    cout << "q.Quit" << endl;
    cout << "Please enter your choice: ";
    cin >> choice;
    //input a character for choice

    complexNum result;//store the operation result of addition or square root
    double m, n;//store the real part and the imaginary part of a complex number
    /*pro1*/
}
```

Explanation:

Define character choice to store the input character. Then show the messages for user to make choices. After user determine which function they will use and enter the corresponding character, store it to choose.

ii. For feature ii:

Code:


```

switch (choice)
{
case 'a': //input the two complex numbers
    cout << "Please enter a complex number: " << endl;
    cout << "The real part: ";
    cin >> m;
    cout << "The imaginary part: ";
    cin >> n;
    num1.set_x(m);
    num1.set_y(n);
    cout << "Please enter an operand: " << endl;
    cout << "The real part: ";
    cin >> m;
    cout << "The imaginary part: ";
    cin >> n;
    num2.set_x(m);
    num2.set_y(n);
    break;
}

```

Explanation:

Apply switch statement to tell which choice have been made by user.

In case a, set values for num1 (stand for the first complex number) and num2 (stand for the operand). Then break to show the menu loop again.

iii. For feature iii:

Code:

```

case 'b': //show the two complex numbers
    cout << "The first complex numbers is ";
    /*pro2 show()*/

    num1.show();
    cout << endl;
    cout << "The operand is ";
    num2.show();
    cout << endl;
    break;
}

```

Explanation:

In case b, show values of num1 (stand for the first complex number) and num2 (stand for the operand). Then break to start over the menu loop.

iv. For feature iv:

Code:

```

case 'c': //do addition and show the result
    result = num1.addition(num2);
    if (num1.read_errNumber() == true)
        //if the operand is 0+0i, private member variable errNumber of num1 will be set to true, in that case, show the below error message.
        cout << "The operand is 0+0i. Please enter two valid complex numbers again!" << endl;
    else
    {
        cout << "The addition result is ";
        result.show();
        cout << endl;
    }
    break;
}

```

Explanation:

In case c, call the addition function of num1 with passing num2 in, return the result to complexNum result. If the operand = $0+0i$, we consider this addition is invalid and show the error message and break to start over the menu loop. Otherwise, the result will show correctly and break to start over the menu loop.

v. For feature v:

Code:

```
case 'd': //output two square roots of the first complex number
    result = num1.square_root(); //firstly, use result to store one of the square root
    cout << "The square roots of the first complex number are: ";
    result.show();
    cout << " and ";
    //let result be the other square root and show it
    result.set_x(-result.get_x());
    result.set_y(-result.get_y());
    result.show();
    cout << endl;
    break;
```

Explanation:

In case d, call the square_root function of num1, return the result to complexNum result. Then show the result, which is one of the square roots of first complex number. After that invert the real part number and imaginary part number of result using set_x and set_y, show the result again. This is the other square root of first complex number. Finally break to start over the loop.

vi. For feature vi:

Code:

```
case 'q': //return true to end the program
    cout << "Goodbye!" << endl;
    return true;
```

Explanation:

In case q, show the message to say "Goodbye!" to user. Return true to end the loop.

vii. For feature vii:

Code:

```
default: //if other inputs are input, show the error message
    cout << "Invalid input! Please Enter again!" << endl;
}
return false; //if the user does not press 'q', return false to continuously show the menu
}
```

Explanation:

If there is no case match with the 'choice', show an error message and return false to start over the loop.

(3). Detail of function bool login():

A. Features:

- i. Ask user to enter a username and determine whether this username has been already registered.
- ii. If username has been registered, ask user enter password to login. If the password is entered wrong for three times, end the program.
- iii. If username is new one, ask user enter password to register.

B. Details of the function:

- i. For feature i:

Code:

```
while (fin.getline(user_inf, 100))//while it is not the end of the file
{
    exist = true;//reset exist
    char stored_user[100] = {};//used to get the username stored in the file
    char stored_password[100] = {};//used to get the password stored in the file
    int count = 0;//record the length of the username stored in the file
    for (count = 0; user_inf[count] != ' '; count++)
        //if the current character in the line is not ' ',
        //which indicates that the username stored in the file has not been fully got
        stored_user[count] = user_inf[count];//get the character for username
    count--;
    //when ending the loop above, the count is the position of the character ' ',
    //move count to the position of the last character of the username
    if (len == count + 1)//if the username input by the user and the username stored in the file has the same length
    {
        for (int i = 0; i <= len - 1; i++)
            if (stored_user[i] != input_name[i])
                //compare the content character by character, in this process,
                //if one difference is detected, which means the input username is not the username stored in the file,
                //set the exist to false
                {
                    exist = false;
                    break;
                }
    }
    else
        exist = false;//if the length of the two strings are different, no need to check character by character
```

Explanation: This part is to compare the input username with saved username.

If they are not same, set exist to false. (More information can be found in comments)

- ii. For feature ii:

Code:

```

if (exist == true)
    //Through the process above, if exist is still true,
    //the username input by the user is the username stored in the file
{
    int times = 1; //store how many times the user has attempted to input the password.
    count += 2;
    //The count before this line shows the last character's position of the username,
    //count+1 is the position of the ' ' between the username and the password,
    //and count+2 is the position of the first character of the password.
    for (int i = count; i <= strlen(user_inf) - 1; i++) //get all characters of the password in the file
        stored_password[i - count] = user_inf[i];
    int lenp = strlen(user_inf) - count; //store the length of the password in the file
    while (times <= 3) // The user can only try three times when inputting a password
    {
        bool flag1 = true; //used to check whether the password is correct.
        cout << "Please enter your password: ";
        char password[100];
        cin.getline(password, 100); //let the user input the password
        int lenpw = strlen(password); //store the length of the password input by the user
        if (lenpw == lenp) //if the password input by the user is of the same length of the password stored in the file
        {
            for (int i = 0; i <= lenpw - 1; i++)
                if (password[i] != stored_password[i]) //check character by character, if one difference is detected
                {
                    times++; //add one time
                    flag1 = false; //set the flag
                    cout << "Wrong password!" << endl; //show message
                    break;
                }
        }

        if (flag1 == true)
            //if through the check process above, the flag is still true,
            //close the file and show message. return true.
        {
            cout << "You have successfully logged in." << endl;
            fin.close();
            return true;
        }
        else //if the two strings are of different length, no need to check character by character
        {
            flag1 = false;
            times++;
            cout << "Wrong password!" << endl;
        }
    }
    //if the program is currently here,
    //which indicates that the user inputs an existing username but fails to input the correct password in three times,
    //show the message and close the file, return false
    cout << "You have entered the password wrongly three times!" << endl;
    fin.close();
    return false;
}
}

```

Explanation: This part is to ask the user to enter the password that matches the username. If the password is right, return true and show the success message to user. If the password is wrong, ask for another input. If the password is wrong for 3 times, end the program. (More information can be found in comments)

iii. For feature iii:

Code:

```

{
    //if the username does not exist in the file, let user set the password
    fin.close();
    cout << "Please set your password: ";
    char set[100]; //store the password the user wants to set
    cin.getline(set, 100);
    ofstream fout("user", ios::app);
    fout << input_name << " " << set << endl;
    cout << "You have successfully registered!" << endl;
    return true;
}

```

Explanation: If the username is not registered, ask for a new password. Then store the password to user file. Add ' ' between username and password to separate them for further use.

(4). Detail of function int main():

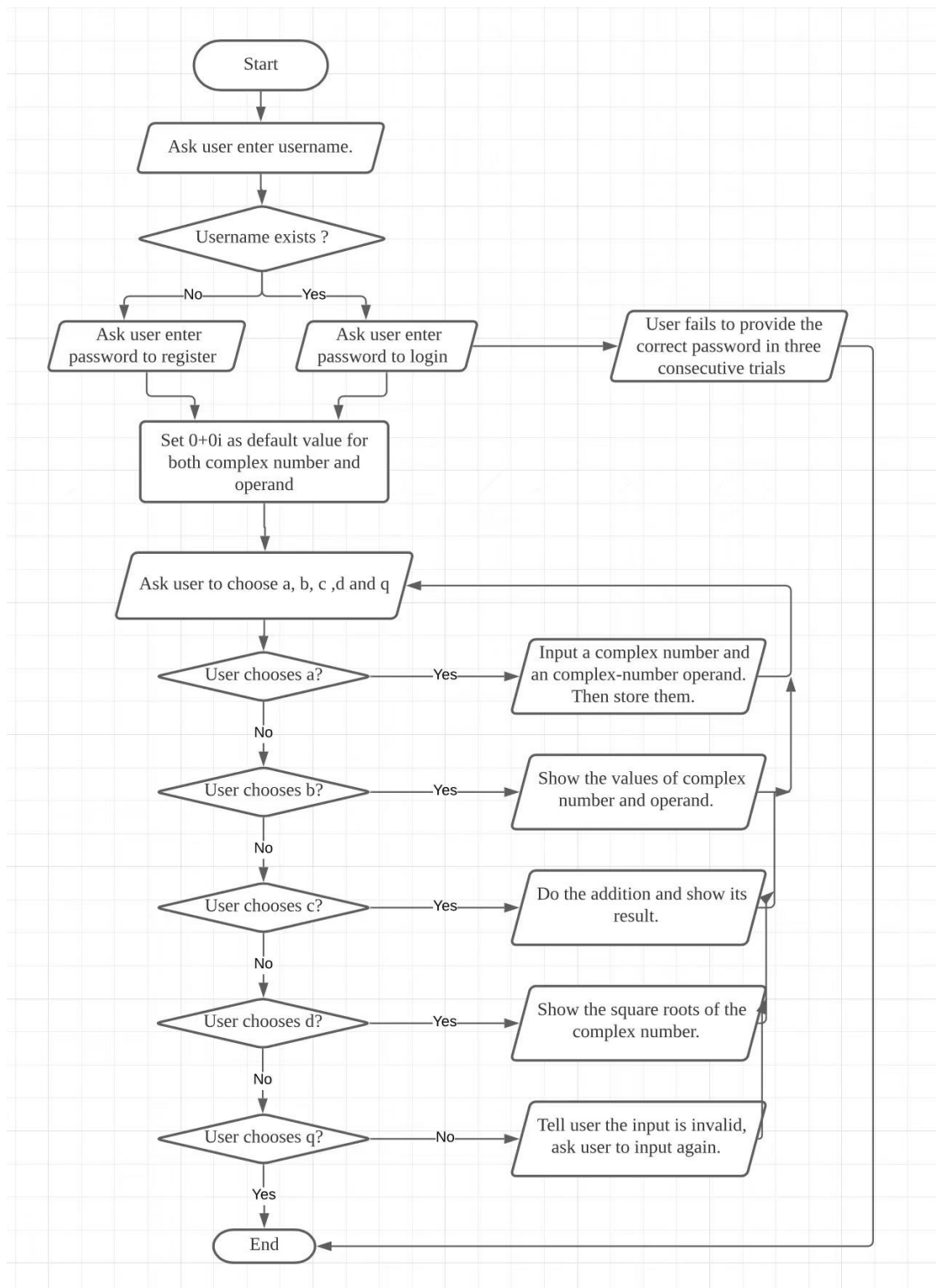
A. Features: Start the whole program and implement loop.

B. Code:

```
int main()
{
    bool log = login();//login
    if (log == false)//if failed to login,end the program
        return 0;
    bool exitflag = false;//used to define when the menu loop will end
    while (exitflag == false)
    {
        exitflag = menu();
    }
    return 0;
}
```

C. Explanation: Start login then start menu loop (More information can be found in comments).

(5). The flow of execution:



4. The problems encountered and corresponding solutions

The first problem we encountered is about the menu function when we did the switch case statements. At the beginning, we mistakenly initialized some values in the “case” statements, the result is that we can’t run the program successfully:

```

37     switch (choice)
38     {
39     case 'a': //input the two complex numbers
40         complexNum result; //store the operation result of addition or square root
41         double m, n; //store the real part and the imaginary part of a complex number
42         cout << "Please enter a complex number: " << endl;
43         cout << "The real part: ";
44         cin >> m;
45         cout << "The imaginary part: ";
46         cin >> n;
47         num1.set_x(m);
48         num1.set_y(n);
49         cout << "Please enter an operand: " << endl;
50         cout << "The real part: ";
51         cin >> m;
52         cout << "The imaginary part: ";
53         cin >> n;

```

By debugging, we found that the error was about the position where we initialized the value. After some attempts, we recalled the truth that we can't do initialization within any "case". Therefore, we just put the two statements outside and then solved the problem.

The second problem we met is when we wanted to show the complex numbers that we input. At first, we didn't build a ".show()" function. We just write "num1" to show the first complex number we entered. However, when the imaginary part is a negative number, the output is something like "7+-9i". To solve this, we create a ".show()" function. In this function, we use an if statement to distinguish two different cases. Here is the detail:

```

58     void complexNum::show()
59     {
60         if (y < 0)
61             cout << x << y << 'i' ;
62         else
63             cout << x << "+" << y << 'i' ;
64     }

```

The third problem we have encountered is about the login function. When we wanted to compare the username that user entered with the username that is already in the text file which stores the username-password, we didn't care about the length of the input name at first. However, a problem happened. When we enter "abcd" and there was only username "abc" in the text file, the program will think there is an account in the file. That is because we read the name character by character and the first 3 characters "abc" are

the same. Then the program just thought the username had already existed. To solve this, we divide the comparison into two parts: First, we check if there is a username having the same length as the username that user inputs. Then we check the username character by character and get the comparison result. By comparing the length first, we ensured that the wrong situation before would not happen and solved the problem.

```
104 cout << "Please input your username: ";
105 cin.getline(input_name, 100);
106 int len = strlen(input_name); //len is the length of the input username
107 ifstream fin("user", ios::in); //read the file from the start
108 char user_inf[100]; //store a line in the file
109 bool exist = true; //check whether the username has been stored in the file already
110
111 while (fin.getline(user_inf, 100)) //while it is not the end of the file
112 {
```

5. Verification of our application:

A. Verification of login function:

register:

```
Please input your username: newtest
Please set your password: newtest
You have successfully registered!
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
```

login:

```
Please input your username: newtest
Please enter your password: newtest
You have successfully logged in.
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
```

fails to provide the correct password in three consecutive trials:

```
Please input your username: newtest
Please enter your password: 2
Wrong password!
Please enter your password: 1
Wrong password!
Please enter your password: 5
Wrong password!
You have entered the password wrongly three times!
```

B. Verification of function of entering and displaying two complex numbers.


```

Please input your username: 1
Please enter your password: 1
You have successfully logged in.
a.Enter one complex numbers and one operand.
b.Display two complex numbers.
c.Addition
d.Square root of the complex number.
q.Quit
Please enter your choice: b
The first complex numbers is 0+0i
The operand is 0+0i
a.Enter one complex numbers and one operand.
b.Display two complex numbers.
c.Addition
d.Square root of the complex number.
q.Quit
Please enter your choice: a
Please enter a complex number:
The real part: 2
The imaginary part: 3
Please enter an operand:
The real part: 4
The imaginary part: 5
a.Enter one complex numbers and one operand.
b.Display two complex numbers.
c.Addition
d.Square root of the complex number.
q.Quit
Please enter your choice: b
The first complex numbers is 2+3i
The operand is 4+5i
a.Enter one complex numbers and one operand.
b.Display two complex numbers.
c.Addition
d.Square root of the complex number.
q.Quit
Please enter your choice:

```

C. Verification of adding two complex number:

```

Please enter your choice: b
The first complex numbers is 2+3i
The operand is 4+5i
a.Enter one complex numbers and one operand.
b.Display two complex numbers.
c.Addition
d.Square root of the complex number.
q.Quit
Please enter your choice: c
The addition result is 6+8i

```

D. Verification of calculating square root:

```

Please enter your choice: b
The first complex numbers is 2+3i
The operand is 4+5i
a.Enter one complex numbers and one operand.
b.Display two complex numbers.
c.Addition
d.Square root of the complex number.
q.Quit
Please enter your choice: d
The square roots of the first complex number are: 1.67415+0.895977i and -1.67415-0.895977i

```

E. Verification of quit function:

```

The square roots of the first complex number are: 1.67415+0.895977i and -1.67415-0.895977i
a.Enter one complex numbers and one operand.
b.Display two complex numbers.
c.Addition
d.Square root of the complex number.
q.Quit
Please enter your choice: q
Goodbye!

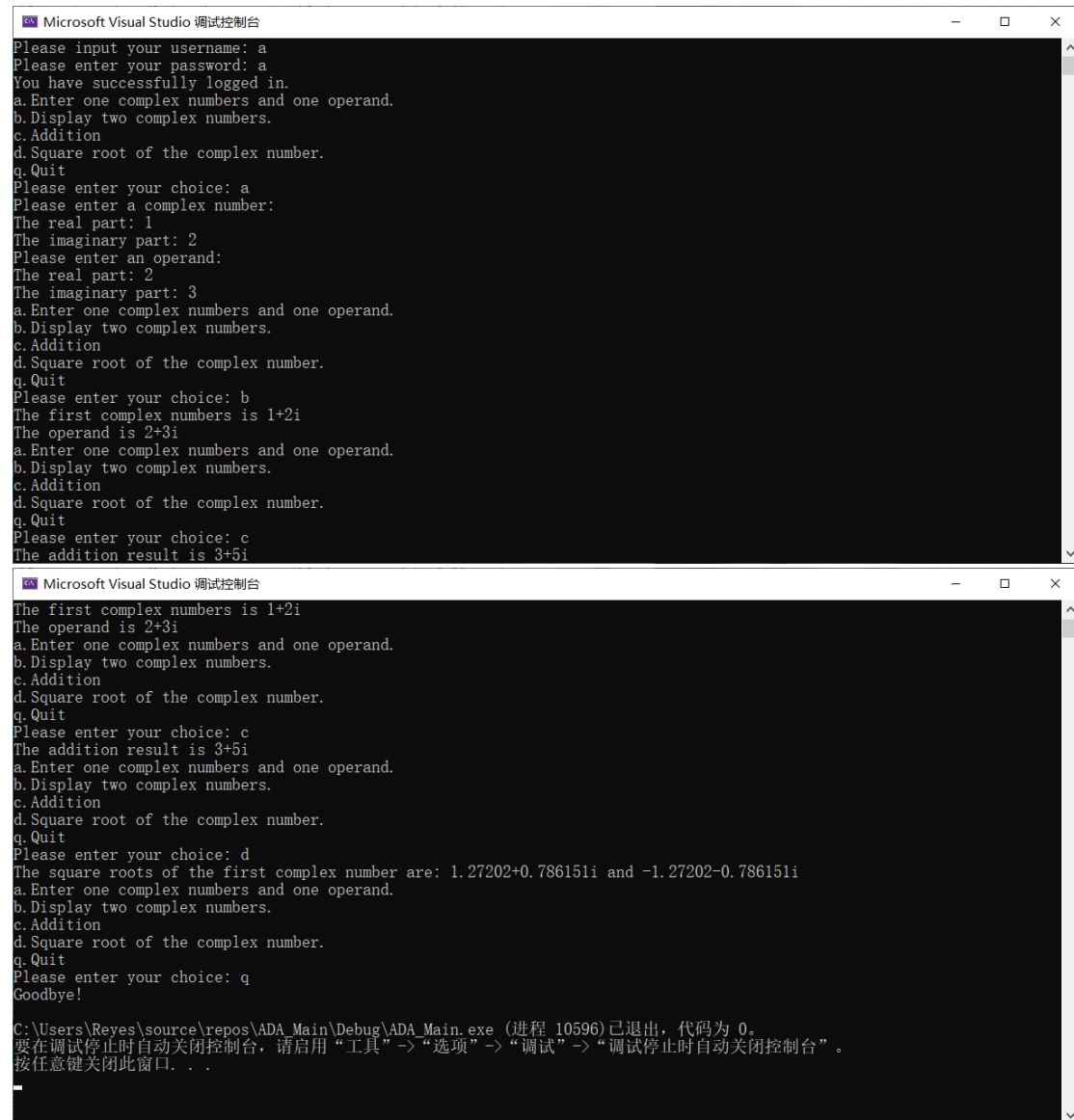
```

After completing all the verifications as above, we can confirm that our program is working properly.

Results

The situation that the user login and runs the program successfully (username: a/password:

a):



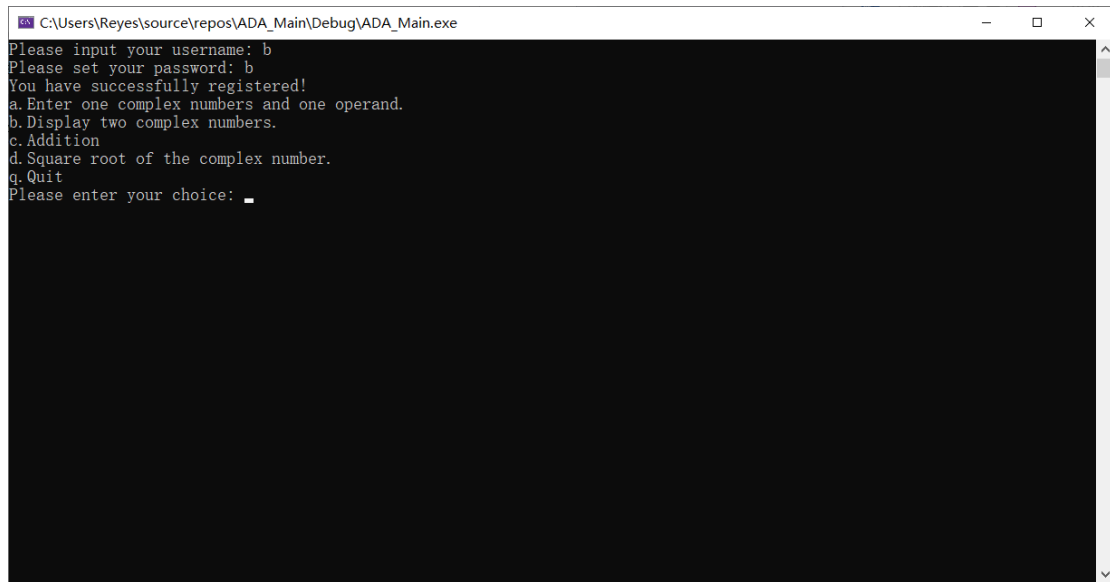
```
Microsoft Visual Studio 调试控制台
Please input your username: a
Please enter your password: a
You have successfully logged in.
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: a
Please enter a complex number:
The real part: 1
The imaginary part: 2
Please enter an operand:
The real part: 2
The imaginary part: 3
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: b
The first complex numbers is 1+2i
The operand is 2+3i
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: c
The addition result is 3+5i

Microsoft Visual Studio 调试控制台
The first complex numbers is 1+2i
The operand is 2+3i
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: c
The addition result is 3+5i
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: d
The square roots of the first complex number are: 1.27202+0.786151i and -1.27202-0.786151i
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: q
Goodbye!

C:\Users\Reyes\source\repos\ADA_Main\Debug\ADA_Main.exe (进程 10596) 已退出。 代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```

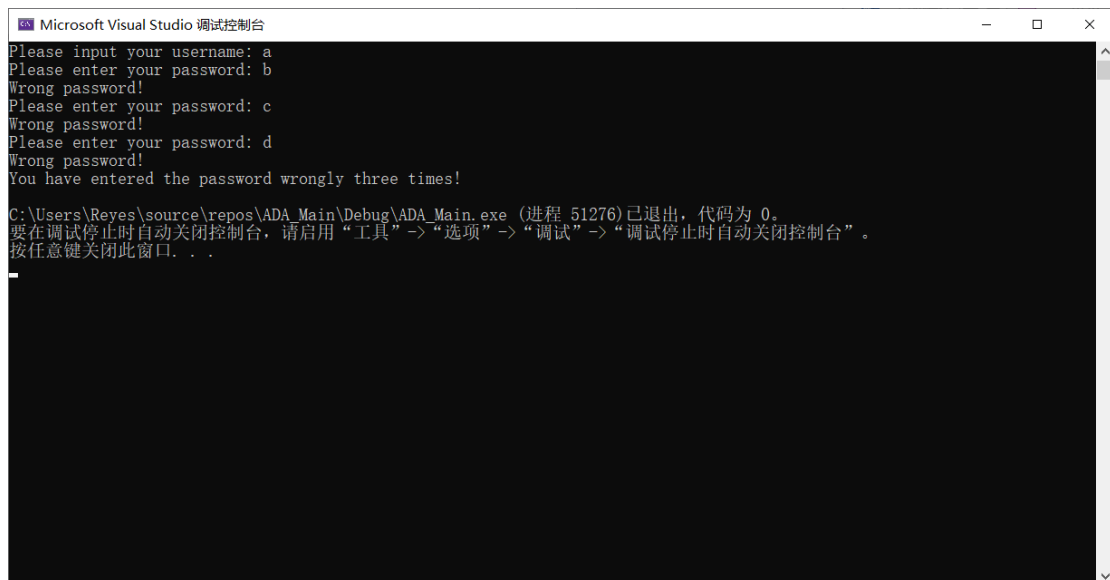
The situation that the username doesn't exist and the program just create one for the user

(username: b/password: b):



```
C:\Users\Reyes\source\repos\ADA_Main\Debug\ADA_Main.exe
Please input your username: b
Please set your password: b
You have successfully registered!
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: _
```

The situation that the user tries the password three times and failed:



```
Microsoft Visual Studio 调试控制台
Please input your username: a
Please enter your password: b
Wrong password!
Please enter your password: c
Wrong password!
Please enter your password: d
Wrong password!
You have entered the password wrongly three times!

C:\Users\Reyes\source\repos\ADA_Main\Debug\ADA_Main.exe (进程 51276) 已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。...
```

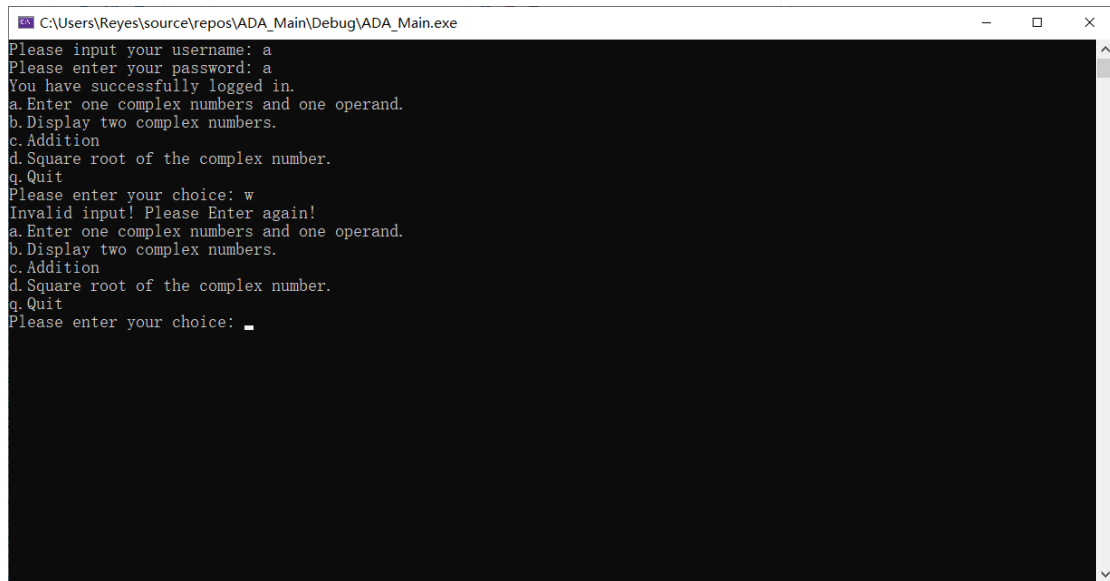
The situation that the operand is zero when doing the addition:

```
C:\Users\Reyes\source\repos\ADA_Main\Debug\ADA_Main.exe
Please input your username: a
Please enter your password: a
You have successfully logged in.
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: a
Please enter a complex number:
The real part: 1
The imaginary part: 2
Please enter an operand:
The real part: 0
The imaginary part: 0
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: b
The first complex numbers is 1+2i
The operand is 0+0i
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: c
The operand is 0+0i. Please enter two valid complex numbers again!
```

The situation that the first input complex number is zero but it doesn't influence the result:

```
C:\Users\Reyes\source\repos\ADA_Main\Debug\ADA_Main.exe
Please input your username: a
Please enter your password: a
You have successfully logged in.
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: a
Please enter a complex number:
The real part: 0
The imaginary part: 0
Please enter an operand:
The real part: 1
The imaginary part: 2
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: b
The first complex numbers is 0+0i
The operand is 1+2i
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: c
The addition result is 1+2i
```

The situation that the user enters invalid choice:



```
C:\Users\Reyes\source\repos\ADA_Main\Debug\ADA_Main.exe
Please input your username: a
Please enter your password: a
You have successfully logged in.
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: w
Invalid input! Please Enter again!
a. Enter one complex numbers and one operand.
b. Display two complex numbers.
c. Addition
d. Square root of the complex number.
q. Quit
Please enter your choice: _
```

Conclusion and further development

In this project, we have realized the operation of addition and getting square root of complex number. Besides, we have completed the function that could show the main menu and the function that could let the user login.

The first experience we get, which is also the most meaningful experience, is that code programming can really realize by a team. Although in the class, we learned that a huge project programming could be completed by a team. However, before this project, what we have experienced is working and solving all kinds of problems met in the programming just by one person. In this project, we have someone to write the class and its member function, a person who is responsible for finishing the user login interface and another one who writes the menu and the main function that will call all the functions above. Our team cooperate and work hard with each other in the whole process. We face all the troubles together and really work as a team. This experience is what we think the most meaningful and important.

Another experience we get from the assignment is "Be patient and keep trying in the programming". As the professor told us when teaching run-time error and compile-time error, at most of time, we can't write the whole program once and run it successfully, there must be

some errors. What we should do is being patient and solve the problem one by one. When necessary, asking for other's help is also important.

As for the further development, the first thing we think is about the user login interface. In our program, basically, the username can only be a consecutive string because we use a space to set boundary between a username and a password, which means there can't be any space. However, in the real world, it is possible that user wants to add some spaces in his/her name, like "Chan Daming". If so, we should improve our program to make it could read username including any number of spaces.

Another improvement we think is for the addition operation. In our program, we could only do addition between two complex numbers. However, the user may want to do addition among 3 or more complex numbers. To achieve this, we think there could be some improvement in the addition member function in the class.

Also, for getting the square root, we could only get the square root of the first complex number input. We think we could let the user choose which square root he/she wants to get, the first complex number or the second.

The last improvement is about when the program prompts the user to input a character for his choice, if the user input a string, the program may get into a dead loop. Although the input is not allowed, we are worried about the user wrongly input a string and cause the trouble. If we have time, we think we can solve this problem by other input methods other than cin.

In conclusion, we have learned many things from this assignment and gained lots of useful experience. And we also become more interesting in the field of computer programming.