浙江理工大學
ZHEJIANG SCI-TECH UNIVERSITY

School of Computer Science and Technology

Lab Report of Object-Oriented Programming A

Lab 1: Classes and Objects Credit hour: 3

Student Name: 代翔 Student ID: 2023337621159

1 Objective

1.1 To master the concept of class, relationship between class and structure, member property and encapsulation of class;

1.2 To master the method to define objects of class;

1.3 To understand the access control of members of the class, the difference among private, public and protected;

1.4 To master the definition, function and realization of constructor and destructor, can define and overload constructor according to the requirement, can define and realize the member function of class.

2 Introduction to lab principle

2.1 By creating class and object, to access the member of class by using the member function and its object;

2.2 By creating the constructor of class, to realize the initialization of the members of class.

3 Lab requirement

3.1 Software: C++ compiler under Windows or linux

3.2 Hardware: main memory(>2GB), free secondary memory(>40G), monitor and printer.

4 Lab content

Write a program to simulate the running of elevator. Function interface include the set-up and set-down button, set_floor_number and get_current_floor (to show the current floor when the elevator is running.)

Requirement:

4.1 It is determined by the user to set up or set down and after the user's selection (up or down). Then the users should set the floor they want to and then the elevator works and shows every floor.

4.2 If the setting is up, then the setting number of floor cannot smaller than the current floor, or the invalid message will pop up.

4.3 If the setting is down, then the setting number of floor cannot larger than the current floor, or the invalid message will pop up.

4.4 As soon as the elevator works, it will work until the window is closed.

4.5 When the elevator goes to the different floor, there should be delay of the display of each floor. And the output of the final floor that the elevator stops should be striking.

4.6 At the beginning of the elevator works, the current date should be displayed in the elevator. (Tip: To realize the function, system api should be called. And the class CDate can be used to realize the function)

5 Code list

1. CDate Header file

```
#pragma once
#include <iostream>
#include<windows.h>
using namespace std;

class CDate {
public:
	void show();
private:
	std::string timeString;
};
```

2. Elevator Header file

```
#pragma once
class elevator
{
public:
	int curfloor;
	int choose;
	int targetfloor;
public:
	elevator();

	void up_display();

	void down_display();
```

```
    void slection();

    void running();

};
```

3. CDate.cpp

```
#include<iostream>
#include <windows.h>
#include"Cdate.h"
using namespace std;

void CDate::show()
{
    SYSTEMTIME time;
    GetLocalTime(&time);
    printf("%04d/%02d/%02d %02d:%02d:%02d\n", time.wYear, time.wMonth, time.wDay, time.wHour, time.wMinute, time.wSecond);
}
```

4. Elevator.cpp

```
#include<iostream>
#include"elevator.h"
#include <windows.h>
#include<vector>
#include<algorithm>
using namespace std;

vector<int> upline;
vector<int> downline;

elevator::elevator(): curfloor(1), targetfloor(1), choose(-1) {};

void elevator::up_display()
{
    Sleep(1000);
    curfloor++;
    cout << endl;
    cout << "Current floor: " << curfloor << endl;
}

void elevator::down_display()
```

```
{
    Sleep(1000);
    curfloor--;
    cout << "Current floor: " << curfloor << endl;
}

void elevator::running()
{
    sort(upline.begin(), upline.end());
    sort(downline.rbegin(), downline.rend());
    auto highest = upline.end()-1;
    auto lowest = downline.end() - 1;
    int up_index = 0;
    int down_index = 0;

    while (curfloor < *highest)
    {
        up_display();
        if (curfloor == upline[up_index])
        {
            cout << "Arrived." << endl;
            up_index++;
        }
    }
    cout <<endl;

    while (curfloor > *lowest)
    {
        down_display();
        if (curfloor == downline[down_index])
        {
            cout << "Arrived." << endl;
            up_index++;
        }
    }
    cout << "Arrived." <<endl;
}

void elevator::slection()
{
    upline.clear();
```

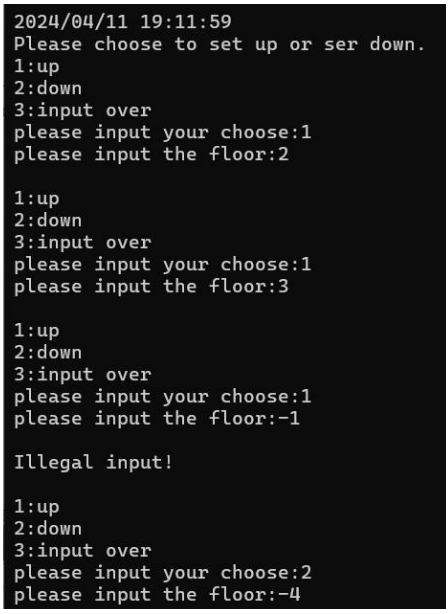
```
downline.clear();
cout << "Please choose to set up or ser down." << endl;
while (choose != 3)
{
    cout << "1:up" << endl << "2:down" << endl << "3:input over" << endl;
    cout << "please input your choose:";
    cin >> choose;
    if (choose == 1)
    {
        cout << "please input the floor:";
        cin >> targetfloor;
        cout << endl;
        if (targetfloor < curfloor)
        {
            cout << "Illegal input!" << endl<<endl;
        }
        else
        {
            upline.push_back(targetfloor);
        }
    }
    if (choose == 2)
    {
        cout << "please input the floor:";
        cin >> targetfloor;
        cout << endl;
        if (targetfloor > curfloor)
        {
            cout << "Illegal input!" << endl<<endl;
        }
        else
        {
            downline.push_back(targetfloor);
        }
    }
    if(choose>3)
    {
        cout << "Illegal input!" << endl<<endl;
    }
}
choose = -1;
```

```
	running();
}
```

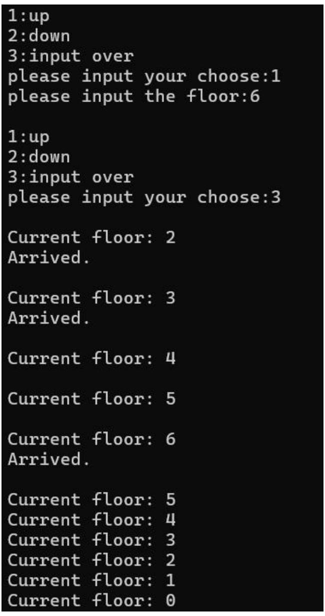
5. Main.cpp

```
#include<iostream>
#include<Windows.h>
using namespace std;
#include"elevator.h"
#include"Cdate.h"

int main()
{
	CDate a;
	a.show();

	elevator test;
	test.slection();
}
```

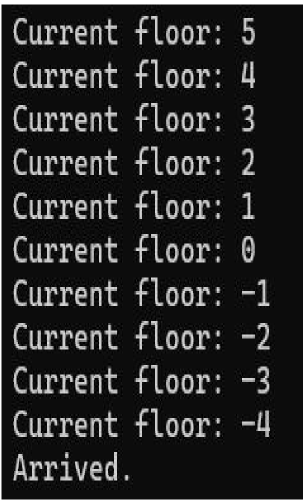
6 Output



```
2024/04/11 19:11:59
Please choose to set up or ser down.
1:up
2:down
3:input over
please input your choose:1
please input the floor:2

1:up
2:down
3:input over
please input your choose:1
please input the floor:3

1:up
2:down
3:input over
please input your choose:1
please input the floor:-1

Illegal input!

1:up
2:down
3:input over
please input your choose:2
please input the floor:-4
```



```
1:up
2:down
3:input over
please input your choose:1
please input the floor:6

1:up
2:down
3:input over
please input your choose:3

Current floor: 2
Arrived.

Current floor: 3
Arrived.

Current floor: 4

Current floor: 5

Current floor: 6
Arrived.

Current floor: 5
Current floor: 4
Current floor: 3
Current floor: 2
Current floor: 1
Current floor: 0
```



```
Current floor: 5
Current floor: 4
Current floor: 3
Current floor: 2
Current floor: 1
Current floor: 0
Current floor: -1
Current floor: -2
Current floor: -3
Current floor: -4
Arrived.
```

7 Analysis and conclusions

My code using CDate header file initialize the function in CDate.cpp,using <windows.h> SYSTEMTIME to initialize the time form,and using GetLocalTime and inference time to evaluate the local time to time,and print it.Elevator header file initialize the function and objects of class elevator,in the Elevator.cpp,I create selection to get the input of the client,and using two dependent vector to store the setting_up floors and setting_down floors,and using <algorithm>header filer to sort them,and using running()function to judge whether the input is legal or not and to show the target floor had arrived or not,and two depart display function() show the current floor.In the main.cpp,I create two class to test the CDate.cpp and elevator.cpp.

Through this experiment,I have mastered the definitions of class and objects,and I have a deeper understanding of the encapsulation of the c++.what's more,I am more familiar with the way to create a header file by myself,and how to call it legally.If we want to call something in header file in the first time,we must add a definition domain before the name of the function or objects.Through this experiment,I also learned we can call a function after defining a class,and use this class add a dot,then input the function name.But there is something we should take a notice of that is if the function or the objects is private or protected,we will fail to call them out of the class.On the contrary,if the function or the objects are public,we can call them out of the class.

Besides,I have also realized that constructor can initialize the objects,and if we don't initialize the constructor of the destructor,the system will help us to create them.Through this experiment,I also get to know to use header file <windows.h> SYSTEMTIME time and GetLocalTime(&time) to get the local time.

This experiment applied the theoretical knowledge learned in the classroom to practice, consolidating and strengthening the learning effect on classes and objects.