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: 余洋 Student ID: 2023337621052

---

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

、、、 elevator.h

```
#pragma once
#include <iostream>
#include<vector>
#include<string>
#include <iomanip>
#include<windows.h>
#include<vector>
#include<algorithm>
#include<string>
using namespace std;

class Cdate
{
private:
    int year;
    int month;
    int day;
    int hour;
    int minute;
    int second;
public:

    Cdate() {};
    void run();
```

```
        void desplay();
        void is_timeset();
};
class elevator {
private:
        int total_floor;
        int current_floor;
        vector<int> target_floor;
        int up_down = 0;
        Cdate time;
public:
        elevator();
        void start();
        void run_elevator(Cdate&time);
        void set();
};
```

、、、

、、、elevator.cpp

```
#include"elevator.h"

elevator::elevator()
{
        cout << "welcome to this elevator,Starting initialization now。\nPlease enter the total floor and current floor" << endl;
        cout << "total_floor: ";
        cin >> total_floor;
        cout << "current_floor:";
        cin >> current_floor;
        cout << "you can press the 0 on the keyboard to input a new one in anytime" << endl;
};

void elevator::set()
{
        cout << "please set up or doun(1 represent up,2 represent down)" << endl;
        int temp_up_down;
        cin >> temp_up_down;
        while (temp_up_down != 1 && temp_up_down != 2)
        {
                cout << "invalid operation,please input again" << endl;
```

```
            cin >> temp_up_down;
        }
        int temp_target_floor;
        cout << "please set your target floor" << endl;
        cin >> temp_target_floor;
        while (temp_target_floor > total_floor || temp_target_floor < 0)
        {
            cout << "invalid floor, please input again" << endl;
            cin >> temp_target_floor;
        }
        target_floor.push_back(temp_target_floor);
        sort(target_floor.begin(), target_floor.end());
        if(up_down ==0) up_down = temp_up_down;
};
void elevator::run_elevator(Cdate& time) {
    while (!target_floor.empty())
    {
        cout << "current_floor: " << current_floor << endl;
        if (total_floor == current_floor || current_floor == 0) up_down = -up_down;
        else if (current_floor < target_floor.front() && up_down == -1) up_down = -up_down;
        else if (current_floor > target_floor.back() && up_down == 1) up_down = -up_down;
        current_floor += up_down;
        for (int i = 0; i < 5; i++)
        {
            time.run();
            if(GetAsyncKeyState('0') & 0x8000) set();
        }
        auto it = find(target_floor.begin(), target_floor.end(), current_floor);
        if (it != target_floor.end())
        {
            cout << "------------------------------------------------------" << endl;
            cout << "          The target floor:   "<< *it<<"   has arrived " << endl;
            cout << "------------------------------------------------------" << endl;
            auto it = target_floor.begin();
            while (it != target_floor.end()) {
                if (*it == current_floor) {
```

```
                        it = target_floor.erase(it);
                    }
                    else {
                        ++it;
                    }
                }
            }
        }
        up_down == 0;

};
void Cdate::is_timeset()
{
    {
        SYSTEMTIME current_time;
        GetLocalTime(&current_time);
        hour = current_time.wHour; minute = current_time.wMinute; second = current_time.wSecond; year = current_time.wYear; month = current_time.wMonth; day = current_time.wDay;
    }
}

void Cdate::run()

{
    Sleep(1000);
    if (second > 59)
    {
        minute += 1;
        second = 0;
    }
    if (minute > 59)
    {
        hour += 1;
        minute = 0;
    }
    if (hour >= 24)
    {
        hour = 0;
    }
```

```
        desplay();
        second++;

};

void Cdate::desplay()
{
        if (second < 10 && minute < 10) cout << year<<" / "<<month <<" / " <<
day <<endl << hour << ":0" << minute << ":0" << second << endl;
        else if (minute < 10) cout << year << " / " << month << " / " << day << endl
<< hour << ":0" << minute << ":" << second << endl;
        else if (second < 10) cout << year << " / " << month << " / " << day << endl
<< hour << ":" << minute << ":0" << second << endl;
        else cout << year << " / " << month << " / " << day << endl << hour << ":"
<< minute << ":" << second << endl;

};
void elevator::start()
{
        static Cdate time;
        time.is_timeset();
        while (true)
        {
                set();
                system("CLS");
                run_elevator(time);
        }
}
```

```
main.cpp
#include "elevator.h"
int main()
{
      elevator A;
      A.start();
}
```

6 Output

```
welcome to this elevator,Starting initialization now。
Please enter the total floor and current floor
total_floor: 10
current_floor:5
you can press the 0 on the keyboard to input a new one in anytime
please set up or down(1 represent up, 2 represent down)
1
please set your target floor
8
current_floor: 5
2023 / 9 / 20
15:04:01
2023 / 9 / 20
15:04:02
2023 / 9 / 20
15:04:03
2023 / 9 / 20
15:04:04
2023 / 9 / 20
15:04:05
please set up or down(1 represent up, 2 represent down)
1
please set your target floor
2
current_floor: 6
2023 / 9 / 20
15:04:06
2023 / 9 / 20
15:04:07
2023 / 9 / 20
15:04:08
2023 / 9 / 20
15:04:09
2023 / 9 / 20
15:04:10
please set up or down(1 represent up, 2 represent down)
2
please set your target floor
3
current_floor: 7
2023 / 9 / 20
```

7 Analysis and conclusions

My program has designed two classes, namely elevators and Cdate. The elevator class is initialized with a constructor, set() is used to read the user's target floor and record it, is_timeset() is used to read the Windows time, desplay() is used to display the time and correct the leading 0, and run() is used to implement the movement of Cdate time. Run_elevator (Cdate&time) is the core of the program, used to schedule all functions and maintain the up and down operation of the elevator. Start is used to initialize the elevator class and start the program.

The main function only includes class creation and calling the start

interface, which is very convenient for users.