浙江理工大学计算机科学与技术学院

《C++程序设计》实验报告

实验名称： 类继承机制的实现　　**学时安排：** 3

实验类别： 设计性实验　　**实验要求：** 1 人 1 组

姓名：陈昊天　　学号：2021329600006

一、实验目的

1.掌握单继承和多重继承的概念。

2.理解不同的继承类型：public、protected 和 private，掌握何时使用何种继承类型。

3.掌握类层次中构造函数的定义方式和建立对象时构造和析构次序

二、实验原理介绍

通过继承机制实现对类功能的扩展,合理设计派生类的构造函数、成员函数。

三、实验设备介绍

软件需求：Visual Studio C++或 Codeblocks 或 Dev C++或其他 C++ IDE

硬件需求：能够流畅运行 C++ IDE 的计算机一台。

四、实验内容

实现对第一次实验结果 Elevator 类的功能扩展。在 Elevator 类已有功能的基础上派生 AdvancedElevator 类。AdvancedElevator 类可以实现当多人在不同楼层等待乘坐上行或下行的同一部电梯时,能够合理的根据乘坐人的需求对电梯经停的楼层进行排序。

要求:

1.为了实现上的方便性，我们假设同一组要求乘坐电梯的乘客或者都是上行，或者都是下行。

2.在主函数中对该类的功能进行测试,测试方法是首先选择在某一时间段一组要乘坐电梯的乘客是上行还是下行,然后输入组中乘客的人数及每一个乘客所

在楼层和目的楼层，由 AdvancedElevator 类实例化后的电梯对象在运作的过程中，如果电梯是上行，则能根据乘客所在的楼层和目的楼层从下向上依次停靠；如果电梯是下行，则能根据乘客所在的楼层和目的楼层从上向下依次停靠。

3.在测试的过程中，还需要注意测试当多个用户在同一楼层或多个用户的目的楼层为同一楼层时情况的处理。

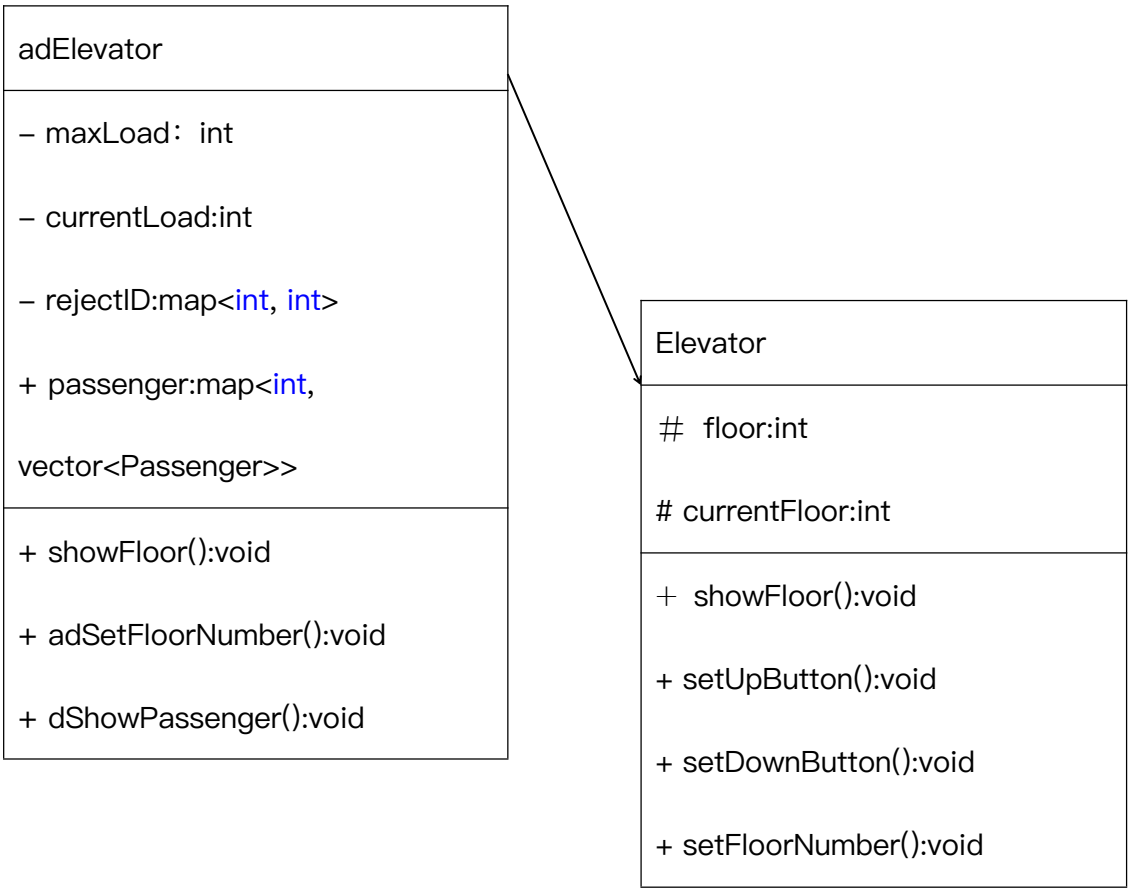
提示：

为了方便描述乘客，我们可以定义一个 Person 类，主要描述每一个乘客所在楼层和目的楼层。AdvancedElevator 类从 Elevator 类继承而来，它从某一个时间段要乘坐电梯的每个乘客的信息当中提取其所在楼层和目的楼层信息，然后对它们进行排序，再由继承自基类 Elevator 的成员 setFloorNumber 对要停靠的楼层序列依次输出。

思考（可选）

如果加入乘客的体重信息，如何实现在停靠楼层对超载信息的提示。

五、类的设计（给出类图）



Passenger
+ id:int + action:string + weight:int

六、程序清单

Main.cpp

expe_c_inherit > main.cpp > ...

```cpp
#pragma optimize(2)
#include "elevator.cpp"

adElevator ele(10, 1, 800);

int main() {
    UI::Start(ele);
    return 0;
}
```

Head.h

expe_c_inherit > C head.h > ...

```
#define windows

#ifdef windows
#include <windows.h>
#endif

#ifndef windows
#include <unistd.h>
#endif

#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <ctime>
#include <iostream>
#include <map>
#include <vector>

using std::cin;
using std::cout;
using std::endl;
using std::map;
using std::max;
using std::min;
using std::string;
using std::swap;
using std::vector;
```

Elevator.h

```cpp
#include "head.h"

class Elevator {
   protected:
    int floor;          //电梯总的楼层数
    int currentFloor;  //当前所在楼层
   public:
    Elevator();
    Elevator(int floor, int currentFloor);
    void showFloor();       //展示楼层信息
    void setUpButton();     //按下上行按钮
    void setDownButton();  //按下下行按钮
    void setFloorNumber(int floorNumber);
    //根据要进入的楼层电梯开始运行，并逐层显示经过的楼层
};

class Passenger {
   public:
    int id;          //乘客编号
    string action;  //乘客动作(in/out)
    int weight;      //乘客体重
};

class adElevator : public Elevator {
   private:
    int maxLoad;              //载荷
    int currentLoad;          //当前载重
    map<int, int> rejectID;  //因超载而无法进入的乘客
   public:
    adElevator();
    adElevator(int floor, int currentFloor, int maxLoad);
    void showFloor();  //展示楼层信息
    void adSetFloorNumber(int minFloor, int maxFloor, string action);
    //根据要进入的楼层电梯开始运行，并逐层显示经过的楼层
    map<int, vector<Passenger>> passenger;   //乘客信息
    void adShowPassenger(int currentFloor);  //显示乘客进出信息
};

namespace UI {
    string currentTime;
    void showTime();                          // 展示当前日期
    void Start(adElevator &ele);              // 开始UI
    void selectOperation(adElevator &ele);  // 选择按钮
    void cls();                               // 清屏
    void slp();                               // sleep
    void fslp();                              // fast sleep
};                                            // namespace UI

```

Elevator.cpp

expe_c_inherit > elevator.cpp > ...

```cpp
#include "elevator.h"

#include "date.cpp"

Elevator::Elevator() {
    floor = 10;
    currentFloor = 1;
}

Elevator::Elevator(int nfloor, int cfloor) {
    floor = nfloor;
    currentFloor = cfloor;
}

adElevator::adElevator() {
    Elevator();
    maxLoad = 800;
}

adElevator::adElevator(int floor, int currentFloor, int nmaxLoad)
    : Elevator(floor, currentFloor) {
    maxLoad = nmaxLoad;
}

void Elevator::showFloor() {
    cout << "当前楼层: " << currentFloor << endl;
    cout << "总楼层数: " << floor << endl;
}

void adElevator::showFloor() {
    cout << "当前楼层: " << currentFloor << endl;
    cout << "总楼层数: " << floor << endl;
    cout << "最大载荷(kg): " << maxLoad << endl;
}

```

```cpp
void Elevator::setUpButton() {
    int ed;
    cout << "请输入目标楼层: " << endl;
    cin >> ed;
    if (ed <= currentFloor or ed > floor) {
        cout << "非法操作." << endl;
    } else {
        setFloorNumber(ed);
    }
}

void Elevator::setDownButton() {
    int ed;
    cout << "请输入目标楼层: " << endl;
    cin >> ed;
    if (ed >= currentFloor or ed < 1) {
        cout << "非法操作." << endl;
    } else {
        setFloorNumber(ed);
    }
}

void Elevator::setFloorNumber(int floorNumber) {
    int l = currentFloor, r = floorNumber;
    if (l < r) {
        for (int cfl = l; cfl < r; cfl++) {
            UI::cls();
            cout << "- 电梯运行中 -" << endl;
            for (int i = floor; i >= 1; i--) {
                if (i == cfl)
                    cout << " |   []   |" << i << endl;
                else
                    cout << " |        |" << i << endl;
            }
            cout << endl;
```

```
71              UI::slp();
72          }
73      } else {
74          for (int cfl = l; cfl > r; cfl--) {
75              UI::cls();
76              cout << "- 电梯运行中 -" << endl;
77              for (int i = floor; i >= 1; i--) {
78                  if (i == cfl)
79                      cout << " |   []   |" << i << endl;
80                  else
81                      cout << " |        |" << i << endl;
82              }
83              cout << endl;
84              UI::slp();
85          }
86      }
87      currentFloor = floorNumber;
88  }
89
90  void adElevator::adShowPassenger(int currentFloor) {
91      for (auto i : passenger[currentFloor]) {
92          if (i.action == "in") {
93              if (currentLoad + i.weight > maxLoad) {
94                  cout << "已超载，拒绝编号为" << i.id << "的乘客进入!"
                    << endl;
95                  rejectID[i.id] = 1;
96              } else {
97                  cout << "在第" << currentFloor << "层接编号为" << i.
                    id
98                       << "的乘客" << endl;
99                  currentLoad += i.weight;
100             }
101         } else if (i.action == "out") {
102             if (rejectID[i.id] == 1) {
```

```
103                 rejectID[i.id] = 0;
104             } else {
105                 cout << "在第" << currentFloor << "层送编号为" << i.
                    id
106                     << "的乘客" << endl;
107                 currentLoad -= i.weight;
108             }
109         }
110         cout << endl;
111         UI::fslp();
112     }
113     passenger[currentFloor].clear();
114 }
115
116 void adElevator::adSetFloorNumber(int minFloor, int maxFloor,
    string action) {
117     int l = minFloor, r = maxFloor;
118     if (action == "down") swap(l, r);
119     setFloorNumber(l);
120     if (l < r) {
121         for (int cfl = l; cfl <= r; cfl++) {
122             UI::cls();
123             cout << "- 电梯运行中 -" << endl;
124             for (int i = floor; i >= 1; i--) {
125                 if (i == cfl)
126                     cout << " |   []   |" << i << endl;
127                 else
128                     cout << " |        |" << i << endl;
129             }
130             cout << endl;
131             adShowPassenger(cfl);
132             UI::slp();
133         }
134         currentFloor = maxFloor;
```

```
    } else {
        for (int cfl = l; cfl >= r; cfl--) {
            UI::cls();
            cout << "- 电梯运行中 -" << endl;
            for (int i = floor; i >= 1; i--) {
                if (i == cfl)
                    cout << " |   []   |" << i << endl;
                else
                    cout << " |        |" << i << endl;
            }
            cout << endl;
            adShowPassenger(cfl);
            UI::slp();
        }
        currentFloor = minFloor;
    }
}
#ifdef windows
void UI::cls() { system("cls"); }
void UI::slp() { Sleep(1000); }
void UI::fslp() { Sleep(500); }
#endif

#ifndef windows
void UI::cls() { system("clear"); }
void UI::slp() { sleep(1); }
void UI::fslp() { Sleep(0.5); }
#endif

void UI::showTime() {
    CDate currentDate;
    currentTime = currentDate.format("ddd");
    cout << "当前日期: " << currentTime << endl;
}
```

```
169
170 void UI::selectOperation(adElevator &ele) {
171     ele.showFloor();
172     cout << "请选择操作(1.上行 2.下行 3.退出): " << endl;
173     int op;
174     cin >> op;
175     if (op == 3) {
176         cout << "已退出系统, 欢迎下次使用." << endl;
177         exit(0);
178     }
179     cout << "请输入乘坐电梯的人数" << endl;
180     int sum, minFloor = INT32_MAX, maxFloor = 0, legal = 1;
181     cin >> sum;
182     for (int i = 1; i <= sum; i++) {
183         int st, ed, weight;
184         cout << "请输入第" << i << "位乘客的楼层起点" << endl;
185         cin >> st;
186         cout << "请输入第" << i << "位乘客的楼层终点" << endl;
187         cin >> ed;
188         cout << "请输入第" << i << "位乘客的体重" << endl;
189         cin >> weight;
190         if (op == 1) {
191             if (st > ed or st < 1 or ed > 10) {
192                 cout << "非法操作." << endl;
193                 legal = 0;
194                 break;
195             }
196         } else {
197             if (st < ed or st > 10 or st < 1) {
198                 cout << "非法操作." << endl;
199                 legal = 0;
200                 break;
201             }
202         }
```

```
203             ele.passenger[st].push_back({i, "in", weight});
204             ele.passenger[ed].push_back({i, "out", weight});
205             minFloor = min({minFloor, st, ed});
206             maxFloor = max({maxFloor, st, ed});
207         }
208
209         if (!legal) {
210             selectOperation(ele);
211         } else if (op == 1) {
212             ele.adSetFloorNumber(minFloor, maxFloor, "up");
213             selectOperation(ele);
214         } else if (op == 2) {
215             ele.adSetFloorNumber(minFloor, maxFloor, "down");
216             selectOperation(ele);
217         }
218     }
219
220     void UI::Start(adElevator &ele) {
221         cls();
222         showTime();
223         selectOperation(ele);
224     }
225
```

七、运行结果

1. 进入系统

```
当前日期: 2022-11-16
当前楼层: 1
总楼层数: 10
最大载荷(kg): 800
请选择操作(1.上行 2.下行 3.退出):
```

2. 选择上行

输入乘客信息

```
请输入乘坐电梯的人数
3
请输入第1位乘客的楼层起点
3
请输入第1位乘客的楼层终点
10
请输入第1位乘客的体重
100
请输入第2位乘客的楼层起点
3
请输入第2位乘客的楼层终点
7
请输入第2位乘客的体重
100
请输入第3位乘客的楼层起点
5
请输入第3位乘客的楼层终点
9
请输入第3位乘客的体重
100
```

电梯运行

```
- 电梯运行中 -
 |       |10
 |       |9
 |       |8
 |       |7
 |       |6
 |       |5
 |       |4
 |   []  |3
 |       |2
 |       |1

在第3层接编号为1的乘客

在第3层接编号为2的乘客
```

电梯运行结束

```
- 电梯运行中 -
|   []   |10
|        |9
|        |8
|        |7
|        |6
|        |5
|        |4
|        |3
|        |2
|        |1

在第10层送编号为1的乘客

当前楼层: 10
总楼层数: 10
最大载荷(kg): 800
请选择操作(1.上行 2.下行 3.退出):
```

3. 选择下行，输入乘客信息

```
当前楼层: 10
总楼层数: 10
最大载荷(kg): 800
请选择操作(1.上行 2.下行 3.退出):
2
请输入乘坐电梯的人数
2
请输入第1位乘客的楼层起点
7
请输入第1位乘客的楼层终点
3
请输入第1位乘客的体重
90
请输入第2位乘客的楼层起点
6
请输入第2位乘客的楼层终点
3
请输入第2位乘客的体重
60
```

电梯运行

```
- 电梯运行中 -
|        |10
|        |9
|        |8
|   []   |7
|        |6
|        |5
|        |4
|        |3
|        |2
|        |1

在第7层接编号为1的乘客
```

电梯运行结束

```
- 电梯运行中 -
 |        |10
 |        |9
 |        |8
 |        |7
 |        |6
 |        |5
 |        |4
 |   []   |3
 |        |2
 |        |1

在第3层送编号为1的乘客

在第3层送编号为2的乘客

当前楼层: 3
总楼层数: 10
最大载荷(kg): 800
请选择操作(1.上行 2.下行 3.退出):
```

4. 发生超载情况

```
- 电梯运行中 -
 |        |10
 |        |9
 |        |8
 |        |7
 |        |6
 |        |5
 |        |4
 |   []   |3
 |        |2
 |        |1

已超载，拒绝编号为2的乘客进入！
```

5. 退出系统

```
当前楼层: 8
总楼层数: 10
最大载荷(kg): 800
请选择操作(1.上行 2.下行 3.退出):
3
已退出系统，欢迎下次使用.
PS E:\Projects\Github\zstu-study\C++程序设计\expe_c_inherit>
```

八、实验心得

（1）学会使用不同的继承类型，掌握何时使用何种继承类型。

（2）派生类不仅能够继承基类的功能，还能够进行扩充、修改和重定义。继承能够减少代

码和数据的重复冗余度，增强程序的重用性。

（3）派生类可以对基类继承得到的成员函数进行覆盖或重载，同时也会影响到它们在派生类中的可见性。基类的同名函数会被派生类重载的同名函数所隐藏。

（4）合理使用条件编译语句，便于控制程序在不同平台时的兼容性问题，根据不同的操作系统使用不同的函数库。

(5) 在项目中慎用 namespace std 与 bits/stdc++.h ，有利于减少可能出现的冲突问题。