浙江理工大学信息学院

实验指导书

实验名称：类的继承机制的实现　　**学时安排**：3

实验类别：设计性实验　　**实验要求**：1人1组

学号：　2015329620004　　姓名：　何宇凤

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：windows 或 linux 下的 c++编译器

硬件需求：对于硬件方面的要求，建议配置是 Pentium III 450 以上的 CPU 处理器，64MB 以上的内存，200MB 的自由硬盘空间、CD-ROM 驱动器、能支持 24 位真彩色的显示卡、彩色显示器、打印机。

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 程序清单

Elevator.h:

```
#include <iostream>
#include<ctime>
#include<cstdlib>
#include<string>
#include<cstdio>
using namespace std;
class CDate{
    int d,m,y;
    static const string df_s;
    static const string df_l;
    public:
    CDate();
    string format(string);
};
class Elevator{
protected:
    int i;
    int number2;
    int number3;
```

```
public:
    Elevator();
    void Showx();
    void Choice();
    void Up_Doc(int n);
    void Down_Doc(int n);
};
class Person{
private:
    int nowfloor;
    int desfloor;
protected:
    int n_floor;
    int d_floor;
public:
    void Cin();
};
class AdvancedElevator:public Elevator,public Person
{
private:
    int xx;
    int num_person;
    int a[20];
    int b[20];
    int per[10];
public:
    void Showy_out();
    void Sort();
```

};**Demo.cpp:**

```
#include"elevator.h"

CDate::CDate()    //初始化

{

    time_t now;


    ::time(&now);


    struct tm *t_now;


    t_now = localtime(&now);
```

```
    y = t_now -> tm_year + 1900;

    m = t_now -> tm_mon + 1;

    d = t_now -> tm_mday;
}
string CDate::format(string df)
{
        char c_df[20];
        if(df == df_s)
        {
            sprintf(c_df, "%d-%d-%d", y, m, d);
            return string(c_df);
        }
        if(df == df_l)
        {
            sprintf(c_df, "%d 年%d 月%d 日", y, m, d);
            return string(c_df);
        }
        return string("");
}
const string CDate::df_s = "ddd";
const string CDate::df_l = "DDD";
Elevator::Elevator()
{
    number3=1;
}
void Elevator::Showx()
{
```

```
    cout<<"该电梯一共十层"<<"当前电梯在"<<number3<<"楼\n"<<endl;
    cout<<"-----请选择操作-----"<<endl;
    cout<<"|***  1.上升    ***|"<<endl;
    cout<<"|***  2.下降    ***|"<<endl;
    cout<<"|***  3.退出    ***|"<<endl;
    cout<<"____________________"<<endl;
}
void Elevator::Choice()
{
    int x, y;
    cin>>x;
    switch(x){
        case 1:
            cout<<"请输入要进入的楼层数\n"<<endl;
            cin>>y;
            Elevator::Up_Doc(y);
            break;
        case 2:
            cout<<"请输入要进入的楼层数\n"<<endl;
            cin>>y;
            Elevator:: Down_Doc(y);
            break;
        case 3:
            break;
        default:
            cout<<"你的选择有误，不能操作\n"<<endl;
            Elevator::Showx();
            break;
    }
}
```

```
void Elevator::Up_Doc(int n)
{
    int n3;
    n3=number3;
    number2=n;
    if(number2>number3){
        for(i=0;i<=(number2-number3);i++){
            cout<<"--"<<number3+i<<"--"<<endl;
            n3++;
        }
        number3=number2;
        cout<<"第"<<number2<<"层到了"<<endl;
    }
}
 void Elevator::Down_Doc(int n)
{
    int n3;
    n3=number3;
    number2=n;
    if(number2<number3){
        for(i=0;i<=(number3-number2);i++){
            cout<<"--"<<number3-i<<"--"<<endl;
           n3++;
        }
    number3=number2;
    cout<<"第"<<number2<<"层到了"<<endl;
    }
 }
 void Person::Cin(){
    cin>>nowfloor>>desfloor;
```

```
    n_floor=nowfloor;
    d_floor=desfloor;
}
void AdvancedElevator::Showy_out()
{
     int i;
     Showx();
     cout<<"请选择操作："<<endl;
     cin>>xx;
     cout<<"        "<<endl;
     cout<<"乘客人数为："<<endl;
     cin>>num_person;
     for(i=1;i<=num_person;i++){
          Cin();
          per[2*i-1]=n_floor;
          per[2*i-2]=d_floor;
     }
     Sort();
     if(xx==1){
         for(i=0;i<num_person;i++){
              Up_Doc(per[i]);
         }
    Showy_out();
     }
    if(xx=2){
         for(i=0;i<num_person;i++){
              Down_Doc(per[num_person-1-i]);
              cout<<"第"<<per[num_person-1-i]<<"到了"<<endl;
         }
         Showy_out();
```

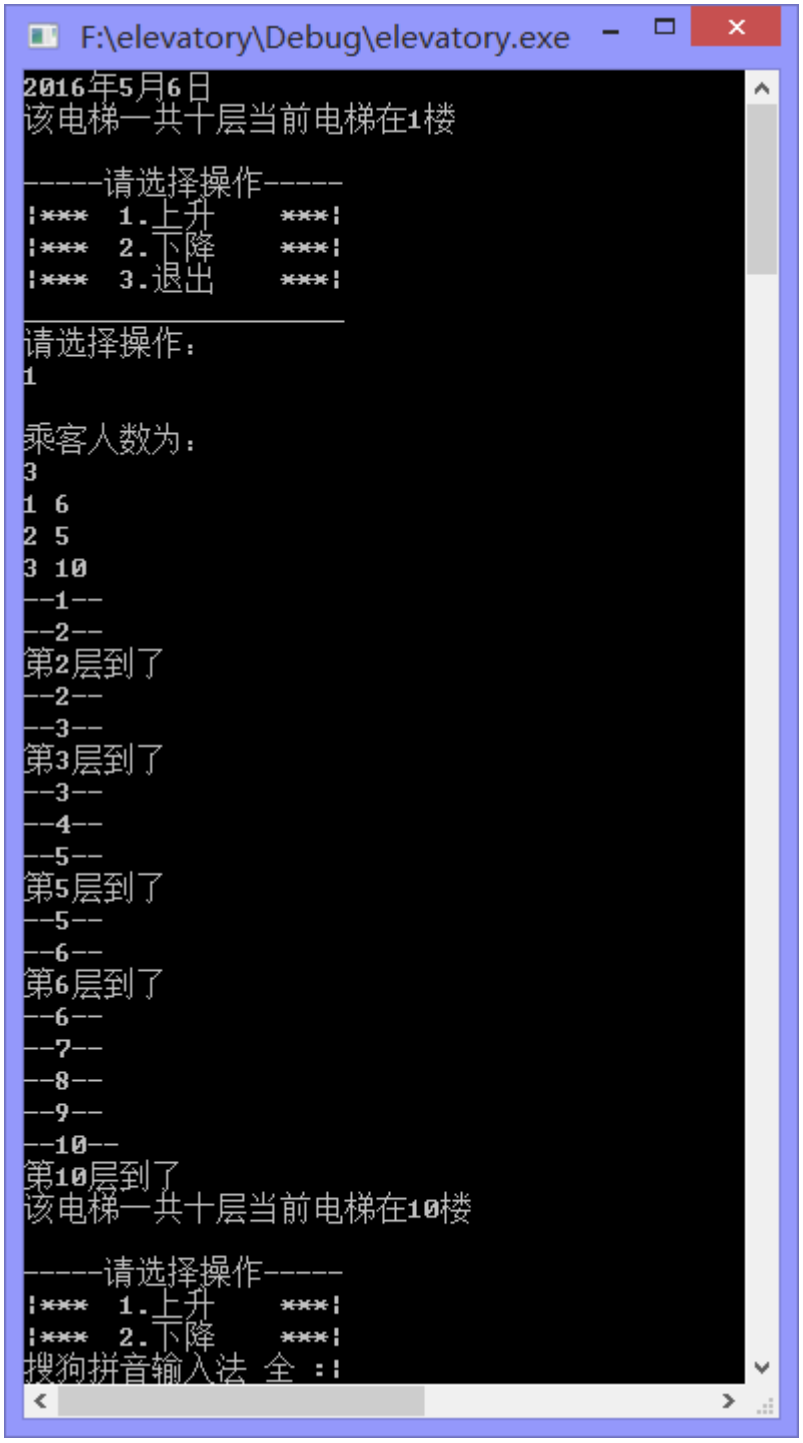
```
    }
}
void AdvancedElevator::Sort()
{
    int i,j,k,temp,min;
    for(i=k=1;i<2*num_person;i++){
        for(j=0;j<k;j++){
            if(per[j]==per[i])
                break;
        }
        if(j==k){
            per[k]=per[j];
            k++;
        }
    }
    num_person=k;
    for(i=0;i<num_person;i++){
        min=i;
        for(j=i+1;j<num_person;j++){
            if(per[min]>per[j])
                min=j;
            temp=per[min];
            per[min]=per[i];
            per[i]=temp;
        }
    }
}
```

Elevator.cpp:

```
#include"elevator.h"
```

```
int main(void)
{
    CDate now;
    AdvancedElevator Elevator1;
    cout << now.format("DDD") <<endl;
    Elevator1.Showy_out();
}
```

六 运行结果



七 实验心得:

本次试验做起来整体还是对继承机制不太熟悉，导致做的过程中还是比较困难，其中对哪些可以直接在基类中定义后，在派生类中直接继承还是拿捏不准，总的来说感觉逻辑还是欠佳，在脑海里罗列整个程序的构造次序一片混乱，写的时候更是混乱，并没有那种程序逻辑先后顺序清楚之感，往往是写了那个之后，才想起写这个，总之十分混乱，跨度大。