# 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：** 类的定义与使用　　**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）掌握类的概念、类的定义格式、类与结构的关系、类的成员属性和类的封装性；

2）掌握类对象的定义；

3）理解类的成员的访问控制的含义，公有、私有和保护成员的区别；

4）掌握构造函数和析构函数的含义与作用、定义方式和实现，能够根据要求正确定义和重载构造函数。能够根据给定的要求定义类并实现类的成员函数；

## 二、实验原理介绍

通过建立类及对象，用类的成员函数和对象访问类的成员；

利用建立类的构造函数，完成类的成员的初始化工作；

## 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

编写一个程序，模拟电梯的功能。功能接口包括电梯上行按钮、下行按钮、楼层选择和电梯在行驶过程中的楼层显示。

要求：

1.由用户选择按上行按钮还是下行按钮，选择操作后再由用户输入要进入的楼层，进而电梯开始运行，显示所到的每一楼层层数。

2.如果是上行，则选择输入的楼层号不能比当前楼层号小，否则应给出不合法提示。

3. 如果是下行，则选择输入的楼层号不能比当前楼层号大，否则应给出不合法提示。

4.电梯一旦开始运作就会始终运行，直到窗口关闭。

5.在程序组织上实现类的接口与类实现的分离。

6.电梯在经过不同楼层时，最好每个楼层的显示之间能有延迟，最终停靠的楼层的输出形式能更加醒目。如果可以，在电梯最初开始运行时，能在电梯由内部显示当前日期（提示：实现这些功能时，需要调用系统 api，实现时间显示功能可以使用 CDate 类）。

## 五 类的设计(类图)

| Elevator |
| --- |
| − floor : int<br>− currentFloor : int |
| + Elevator()<br>+ showFloor() : void<br>+ setUpButton() : void<br>+ setDownButton() : void<br>+ setFloorNumber() : void |

## 六 程序清单

Main . cpp

main.cpp > ...

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

Elevator ele;

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

Elevator.h

elevator.h > { } UI > currentTime

```h
#include "head.h"

class Elevator {
   private:
    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();       //展示当前楼层
    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
};

namespace UI {
string currentTime;
void showTime();                        // 展示当前日期
void Start(Elevator &ele);              // 开始UI
void selectOperation(Elevator &ele);    // 选择按钮
};                                      // namespace UI

```

Elevator.cpp

elevator.cpp > setFloorNumber(int)

```cpp
#include "elevator.h"

#include "date.cpp"

using std::cin;
using std::swap;

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

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

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

void Elevator::setUpButton() {
    int ed;
    cout << "请输入目标楼层: " << endl;
    cin >> ed;
```

```
27      if (ed <= currentFloor or ed > floor) {
28          cout << "非法操作." << endl;
29      } else {
30          setFloorNumber(ed);
31      }
32  }
33
34  void Elevator::setDownButton() {
35      int ed;
36      cout << "请输入目标楼层: " << endl;
37      cin >> ed;
38      if (ed >= currentFloor or ed < 1) {
39          cout << "非法操作." << endl;
40      } else {
41          setFloorNumber(ed);
42      }
43  }
44
45  void Elevator::setFloorNumber(int floorNumber) {
46      int l = currentFloor, r = floorNumber;
47      if (l < r) {
48          for (int cfl = l; cfl <= r; cfl++) {
49              system("clear");
50              cout << "- 电梯运行中 -" << endl;
51              for (int i = floor; i >= 1; i--) {
52                  if (i == cfl)
53                      cout << " |   []   |" << i << endl;
54                  else
55                      cout << " |        |" << i << endl;
56              }
57              cout << endl;
58              sleep(1);
59          }
60      } else {
61          for (int cfl = l; cfl >= r; cfl--) {
62              system("clear");
63              cout << "- 电梯运行中 -" << endl;
64              for (int i = floor; i >= 1; i--) {
65                  if (i == cfl)
66                      cout << " |   []   |" << i << endl;
67                  else
68                      cout << " |        |" << i << endl;
69              }
70              cout << endl;
71              sleep(1);
72          }
73      }
74      currentFloor = floorNumber;
75  }
76
77  void UI::showTime() {
78      CDate currentDate;
```

```
79      currentTime = currentDate.format("ddd");
80      cout << "当前日期: " << currentTime << endl;
81  }
82
83  void UI::selectOperation(Elevator &ele) {
84      ele.showFloor();
85      cout << "请选择操作(1.上行 2.下行 3.退出): " << endl;
86      int op;
87      cin >> op;
88      if (op == 1) {
89          ele.setUpButton();
90          selectOperation(ele);
91      } else if (op == 2) {
92          ele.setDownButton();
93          selectOperation(ele);
94      } else {
95          cout << "已退出系统, 欢迎下次使用." << endl;
96          exit(0);
97      }
98  }
99
100 void UI::Start(Elevator &ele) {
101     system("clear");
102     showTime();
103     selectOperation(ele);
104 }
```

Head.h

C head.h

```
1  #include <unistd.h>
2
3  #include <cstdio>
4  #include <cstdlib>
5  #include <ctime>
6  #include <iostream>
7
8  using std::cout;
9  using std::endl;
10 using std::string;
```

系统 API date.h, date.cpp 不再列出

## 七 运行结果

1. 进入系统

```
问题    输出    调试控制台    终端

当前日期: 2022-10-13
当前楼层: 1
总楼层数: 10
请选择操作(1.上行 2.下行 3.退出):
```

2. 选择上行

```
问题    输出    调试控制台    终端

当前日期: 2022-10-13
当前楼层: 1
总楼层数: 10
请选择操作(1.上行 2.下行 3.退出):
1
请输入目标楼层:
3


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

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

电梯运行时会刷新控制台

3. 选择上行，输入小于当前楼层

```
当前楼层: 3
总楼层数: 10
请选择操作(1.上行 2.下行 3.退出):
1
请输入目标楼层:
2
非法操作.
当前楼层: 3
总楼层数: 10
请选择操作(1.上行 2.下行 3.退出):
```

4. 选择下行

```
当前楼层：3
总楼层数：10
请选择操作(1.上行 2.下行 3.退出):
2
请输入目标楼层：
2
```

问题 输出 调试控制台 终端

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

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

电梯运行时会刷新控制台

5. 选择下行，输入大于当前楼层

```
当前楼层：2
总楼层数：10
请选择操作(1.上行 2.下行 3.退出):
2
请输入目标楼层：
3
非法操作.
当前楼层：2
总楼层数：10
请选择操作(1.上行 2.下行 3.退出):
```

6. 退出

```
当前楼层：2
总楼层数：10
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.
```

## 八 实验心得

1. 应当正确处理好项目中各个文件的包含关系.

2. 本项目的时间函数使用了 linux 中的 unistd.h, 如果在 windows 中应注意更换头文件和时间函数.

3. 项目实现过程中注意类的接口与实现分离.

4. 合理使用命名空间以区分不同功能的函数，便于后期调试与查看代码.

5. 系统API中'sprintf'已被弃用. 这个函数只是为了兼容而提供的. 由于sprintf()设计中固有的安全问题，建议使用 snprintf3)来代替.