# 《C语言程序设计》实验设计报告

题目: 三层电梯状态机编程

学院: 机电工程

专业班级: 自动化31班

学生姓名: 李菁菁

学号: 15040310019

指导老师: 段江涛

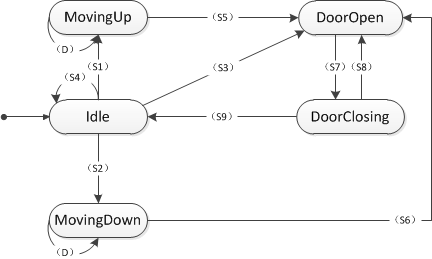
2017年5月18日

# 一、算法思想

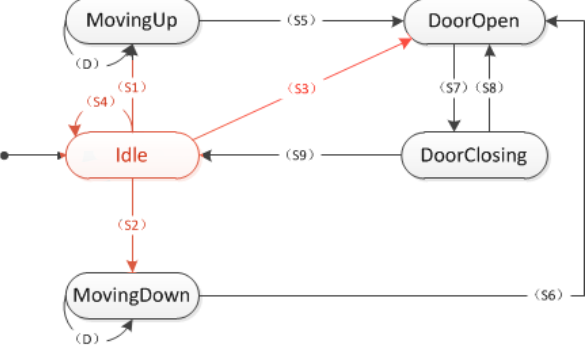
通过有限状态机模型使电梯在个状态及在这些状态之间的转移和动作等完成电梯的上下楼与开关门，状态机通常包括状态的管理、状态的监控、状态的触发、状态触发后引发的动作。

# 二、状态机模型

如图所示，将电梯分为以下五个状态，并按箭头方向进行切换状态：

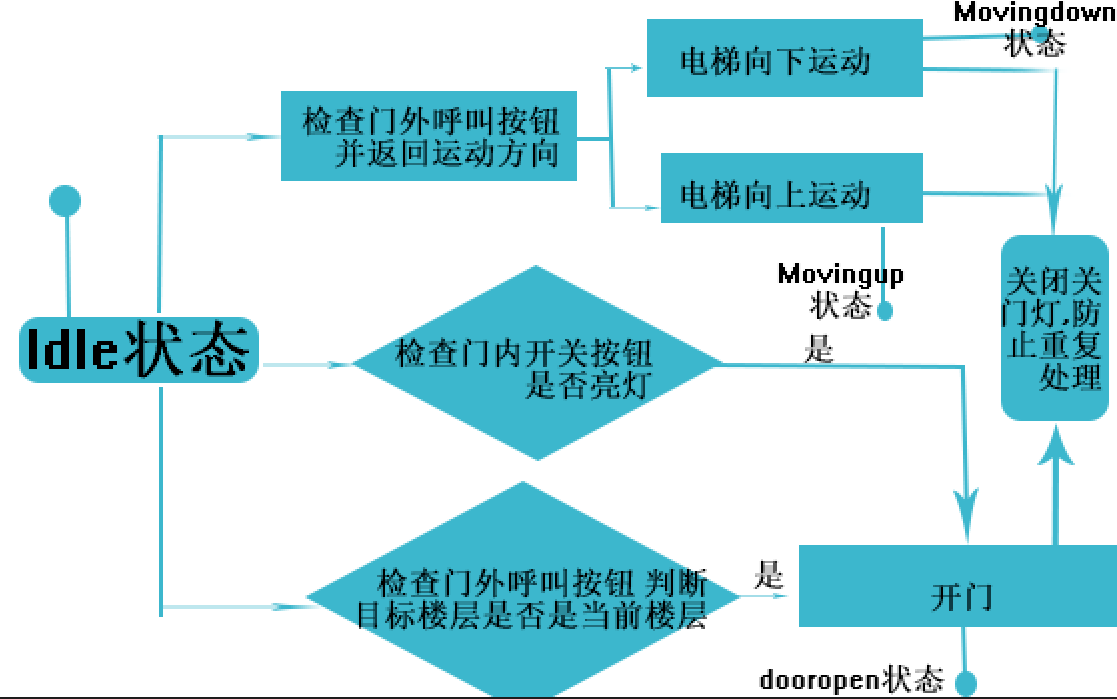


## ldle状态

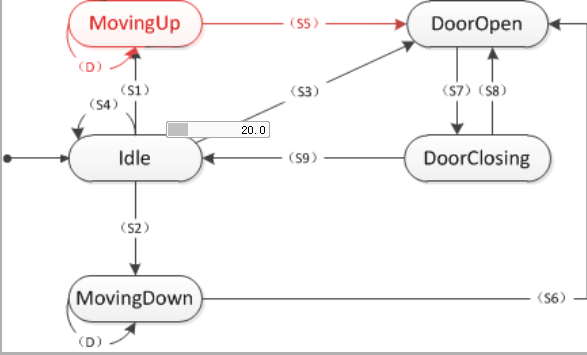


电梯停止在某楼层，门是关闭的，处于静止状态，等待相关事件的发生，从而转换到下一个状态：

## 流程图

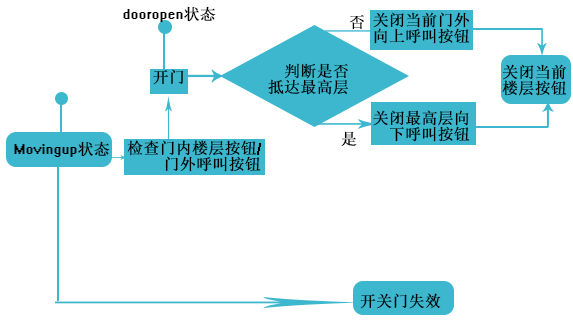


## MovingUp状态

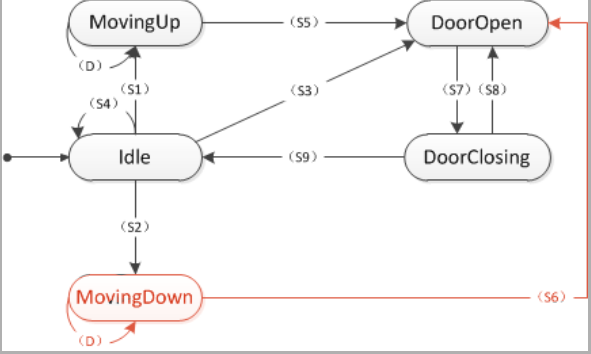


当前处于上升状态，检测事件发生改变状态：

## 流程图

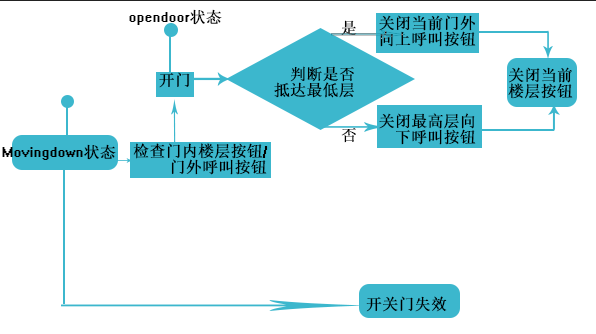


## MovingDown状态

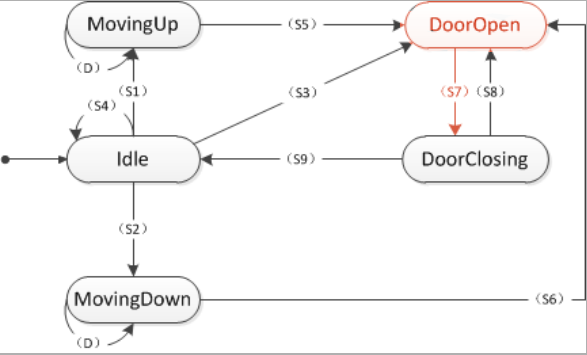


当前处于上升状态，检测事件发生改变状态。

## 流程图

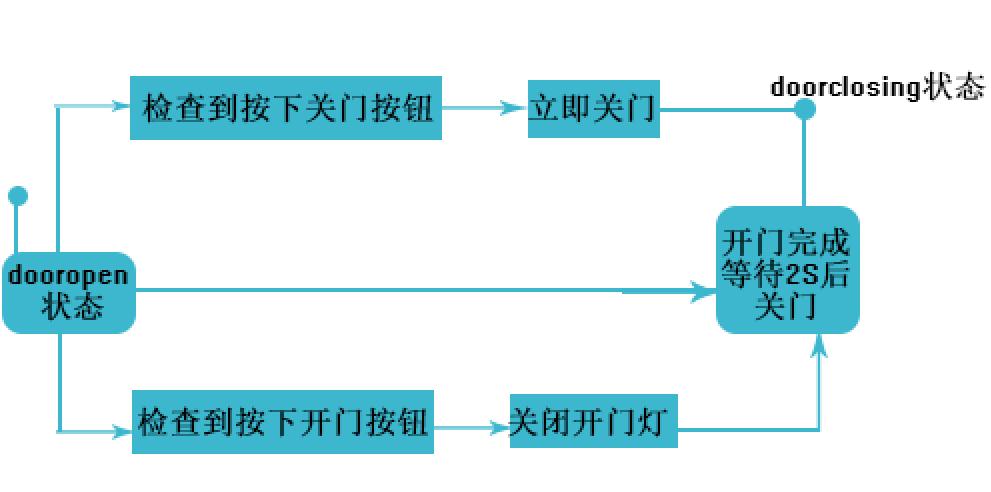


## dooropen状态

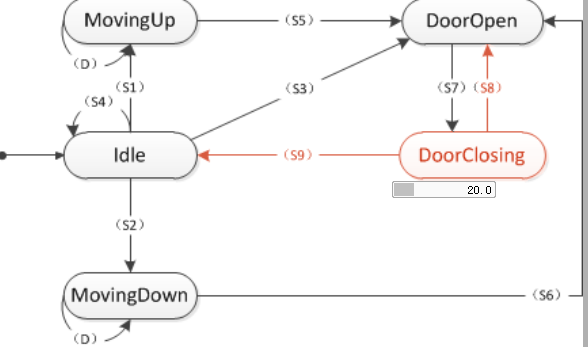


电梯停留在某一层，电梯门处于正在打开和已经打开的状态，等待相关事件发生改变状态：

## 流程图

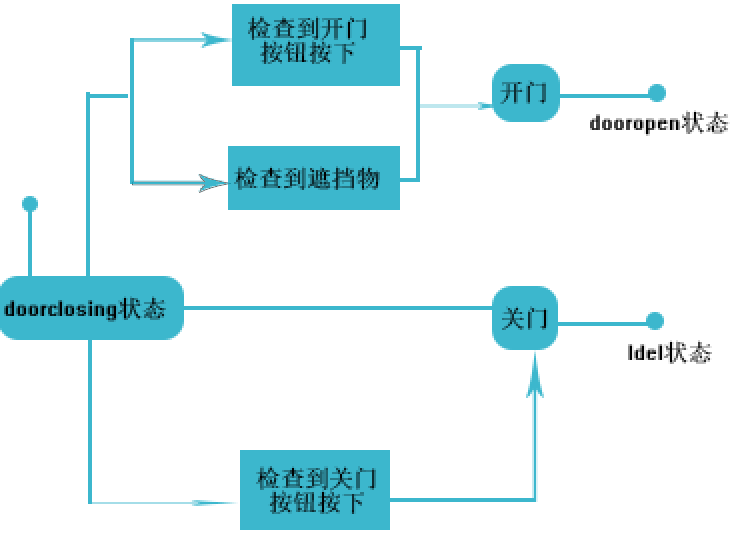


## doorclosing状态



电梯停留在某一层，电梯门处于正在关闭和关闭状态：

## 流程图



# 三、状态机代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Idle状态，电梯停止在某楼层，门是关闭的，处于静止状态，等待相关事件的发生，从而转换到下一个状态。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void StateIdle(int \*state)

{

int floor,nowfloor;

bool up;

nowfloor = GetNearestFloor();

floor = IdleWhatFloorToGoTo(&up);

printf("%d",up);

if (floor > 0 && up) {

SetMotorPower(1);

\*state = MovingUp;

}//上移

if (floor > 0 && !up) {

SetMotorPower(-1);

\*state = MovingDown;

}// 检查E3 E4 E5事件 并返回运动方向

if (GetOpenDoorLight()){

SetDoor(nowfloor, 1);

SetOpenDoorLight(0);

\*state = DoorOpen;//检查门内开关按钮 是否亮灯 亮灯则开门

}

if (GetCallLight(nowfloor, 1) && (up || !GetCallLight(3, 0) && (!up || GetCallLight(1, 1)))){

SetDoor(nowfloor,1);

SetCallLight(nowfloor,1,0);

SetCallLight(nowfloor, 0, 0);

\*state = DoorOpen;

}//检查E4 E5 判断目标 楼层是否是当前楼层 是即开门进入dooropen

GetCloseDoorLight();

if (GetCloseDoorLight()) { SetCloseDoorLight(false); return; }

}

void StateMovingUp(int \*state)//电梯上移

{

int floor;

bool up;

floor = GoingUpToFloor();

printf("目标%d", floor);

printf("当前%f \n", GetFloor());

if ((fabs(GetFloor() - floor) < Lib\_FloorTolerance) &&!( (GetCallLight(3, 0)) && !GetCallLight(2, 1) )){//判断是否到达

SetMotorPower(0);

SetDoor(floor, 1); //开门

if ((GetCallLight(floor, 0) )&& (GetCallLight(floor,1 )))

SetCallLight(floor, 1, false);

else{ SetCallLight(floor, 1, false);

SetCallLight(floor, 0, false); }

SetPanelFloorLight(floor, 0);

SetPanelFloorLight(floor, 0);//关up灯

\*state = DoorOpen;

}

if ((fabs(GetFloor() - 3)) < Lib\_FloorTolerance){

SetMotorPower(0);

SetDoor(floor, 1);

SetCallLight(floor, 0, false);

}

if (GetOpenDoorLight() ||GetCloseDoorLight()){

SetOpenDoorLight(0);

SetCloseDoorLight(0);

}//开关门失效

}

void StateMovingDown(int \*state)

{

int floor;

bool up;

floor = GoingUpToFloor();

printf("目标%d", floor);

printf("当前%f \n", GetFloor());

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance && !(GetCallLight(1, 1) && !GetCallLight(2, 0)) || fabs(GetFloor() - 1) < Lib\_FloorTolerance){//判断是否到达

SetMotorPower(0);

SetDoor(floor, 1); //开门

if (fabs(GetFloor() - 1) < Lib\_FloorTolerance){//判断是否是最低层

SetCallLight(1, 1, false);

SetPanelFloorLight(1, 0);//关up灯

}

else SetCallLight(floor, 0, false);

SetPanelFloorLight(floor, 0);//关down灯

\*state = DoorOpen;

}

if (GetOpenDoorLight() ||

GetCloseDoorLight()){

SetOpenDoorLight(0);

SetCloseDoorLight(0);

}//开关门失效

}

void StateDoorOpen(int \*state)

{

int floor;

floor = GetNearestFloor();

if (GetCloseDoorLight() == 1){

SetDoor(floor,0);

SetCloseDoorLight(0);

\*state = DoorClosing;

}

if (IsDoorOpen(floor)){

SetDoor(floor, 0);

\*state = DoorClosing;

}

if(GetOpenDoorLight())

SetOpenDoorLight(0);

}

void StateDoorClosing(int \*state)

{ int floor;

floor = GetNearestFloor();

if (GetOpenDoorLight()){

SetDoor(floor,1);

SetOpenDoorLight(0);

}

if (GetCloseDoorLight()){

SetCloseDoorLight(0);

\*state = DoorOpen;

}

if (IsBeamBroken()){ SetDoor(floor, 1);

\*state = DoorOpen;

}

if (IsDoorClosed(floor))

\*state = Idle;

}/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 状态机，每隔一定时间(如，100ms)被调用一次，采集系统的运行状态

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main\_control(int \*state)

{

if(IsElevatorRunning()) // 仿真正在运行

{

switch(\*state)

{

case Idle:

// Idle状态，一定时间无动作，自动到一楼

if(GetNearestFloor() !=1 ) {

AutoTo1Floor();

}

StateIdle(state);

break;

case MovingUp:

CancelTo1Floor(); // 其它状态，取消自动到一楼

StateMovingUp(state);

break;

case MovingDown:

CancelTo1Floor();

StateMovingDown(state);

break;

case DoorOpen:

CancelTo1Floor();

StateDoorOpen(state);

break;

case DoorClosing:

CancelTo1Floor();

StateDoorClosing(state);

break;

default:

printf("没有这种状态!!!\n");

}

}

}

# 电梯功能测试

1. 电梯停于1F或2F时，按3F向下呼叫按钮；电梯上升到3F停止，开门，2s关门。

2.电梯停于2F或3F时，按1F向上呼叫按钮；电梯下降到1F停止，开门/关门。

3.电梯停于1F时，按2F向上呼叫按钮；电梯上升到2F停止，开门/关门。

4.电梯停于3F时，按2F向下呼叫按钮；电梯下降到2F停止，开门/关门。

5.电梯停于1F，2F和3F均有按钮呼叫；电梯先上升到2F，开门/关门，然后上升到3F停止，开门/关门。

6.电梯停于3F，2F和1F均有按钮呼叫；电梯先下降到2F，开门/关门，然后下降到1F停止，开门/关门。

7.电梯上升途中或下降途中，任何反方向按钮呼叫均无效。

8.电梯停于1F，按3F向下呼叫按钮，然后立即按2F向下呼叫按钮；电梯上升到3F停止，开门/关门，然后下降到2F停止，开门/关门。

9.电梯停于2F，按门内楼层按钮3，然后门内楼层按钮1；电梯上手到3F停止，开门/关门，然后下降到1F，开门/关门。

10.电梯停于1F，按门内楼层按钮3，当电梯上升在1F到2F中间以下，按2F向上呼叫按钮；.电梯先上升到2F，开门/关门；然后再到3F，开门/关门。

11.电梯停于1F，按门内楼层按钮3，当电梯上升在1F到2F中间以上，按2F向上呼叫按钮；12.电梯先上升到3F，开门/关门；然后再到2F，开门/关门。

13.电梯停于1F，按2F向下呼叫按钮和向上呼叫按钮以及3F的向下呼叫按钮；电梯上升到2F停止，开门/关门，2F的向上呼叫按钮灯关闭，开门/关门， 2F的向下呼叫按钮灯关闭，然后上升到3F停止，开门/关门，3F的向下呼叫按钮关闭。

14.电梯在2F以上，10s无动作，自动降到1楼。

所有停止，开门/关门后，对应楼层的同方向门外呼叫按钮灯（最高楼向下呼叫按钮，最底层向上呼叫按钮）和门内楼层按钮灯关闭。

15.运动状态，开关门按钮失效。正在开门，开门按钮失效；正在关门，关门按钮失效。

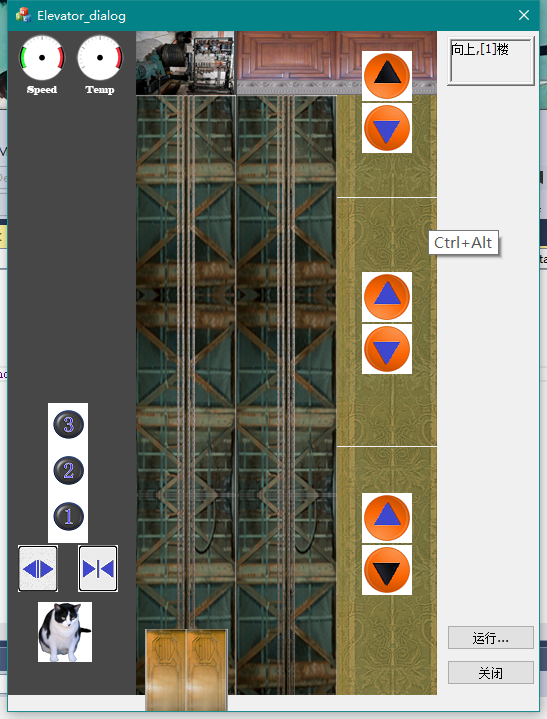
16.空闲状态，门是关闭的，因此按关门按钮失效。

17.开关门结束，延时2秒用于乘客上下电梯，然后进入关门状态。

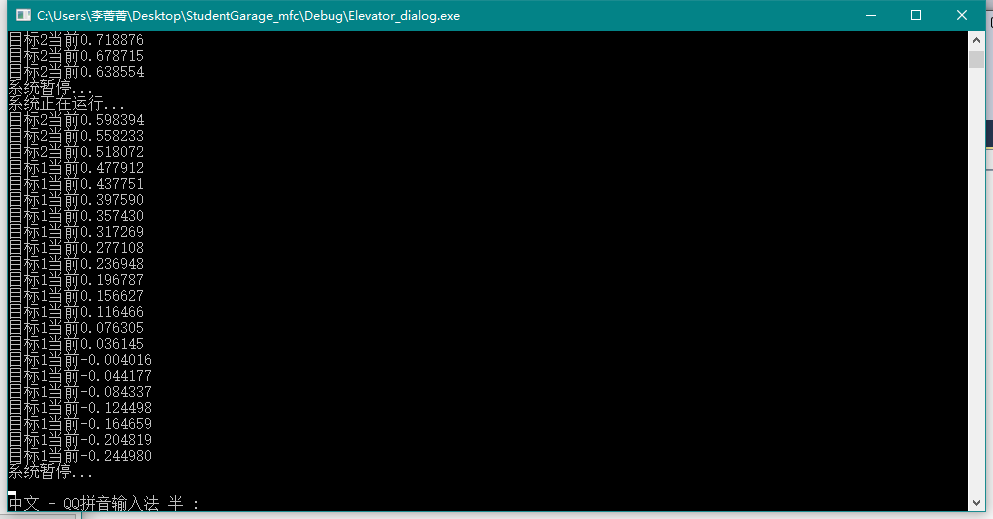
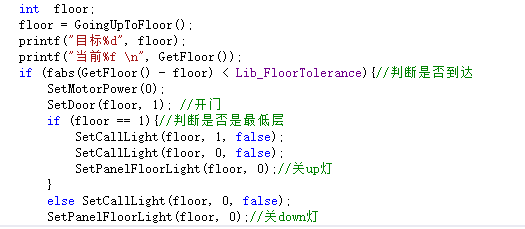
18.开门结束前，按关门按钮，转而进入关门状态；关门结束前，按开门按钮，转而进入开门状态。

## 在测试中遇到的问题和解决办法

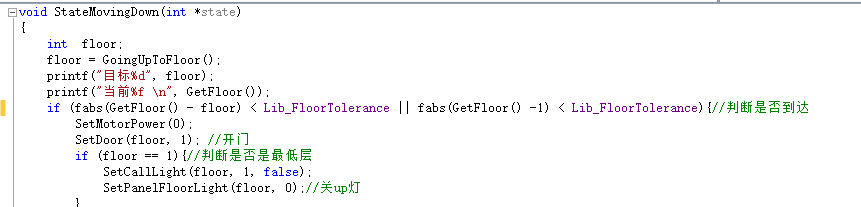
1.在自动回到一楼时无法自动停下



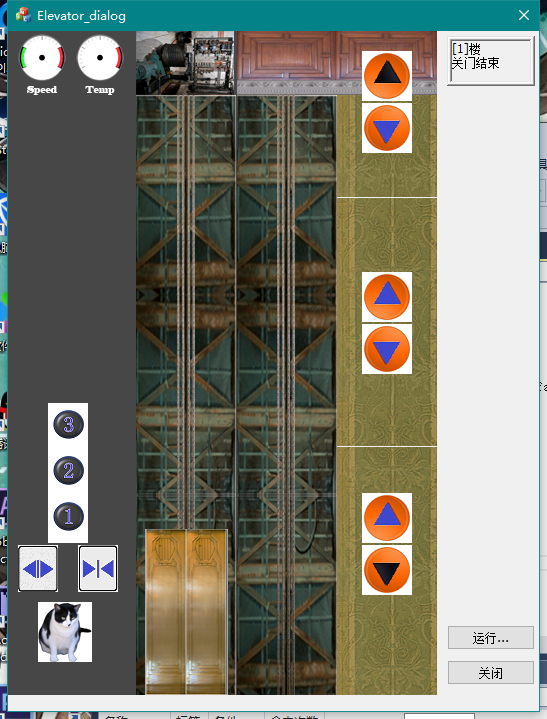
解决方法：加入printf函数观察楼层当前楼层目标楼层的变化

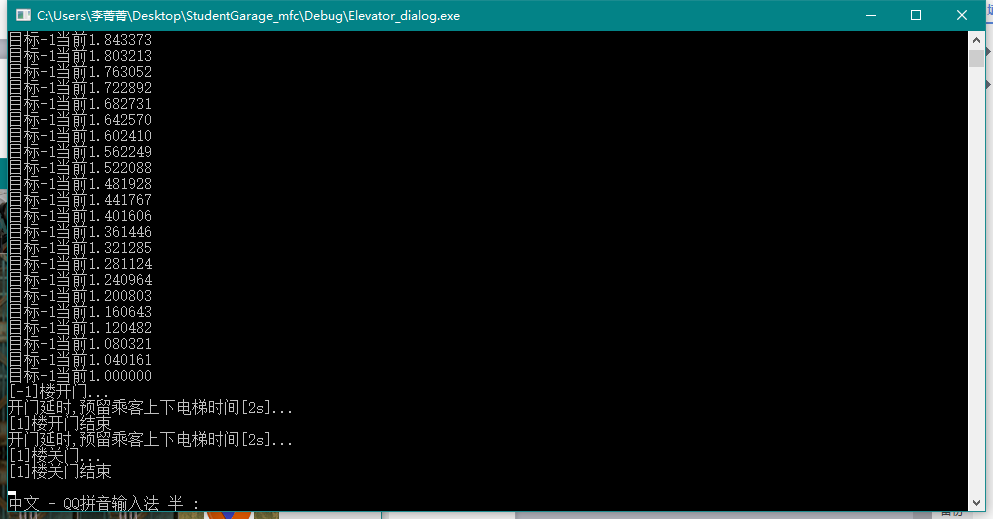


修改movingdown中if条件



再次运行

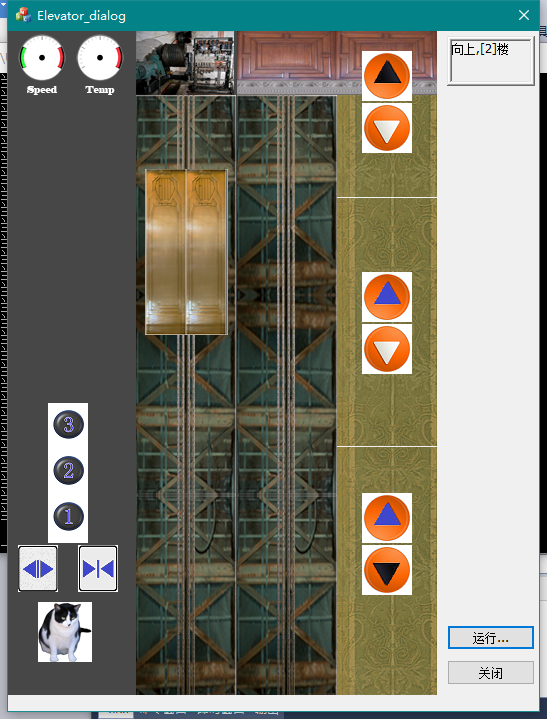




1. 同时按2楼向下呼叫按钮和三楼呼叫按钮后，电梯在2楼停下开关门，然后上至三楼。

Movingup/movingdown的if语句分别添加条件为

if ((fabs(GetFloor() - floor) < Lib\_FloorTolerance) && !(GetCallLight(3, 0) && !GetCallLight(2, 1)))

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance && !(GetCallLight(1, 1) && !GetCallLight(2, 0)) || fabs(GetFloor() - 1) < Lib\_FloorTolerance)

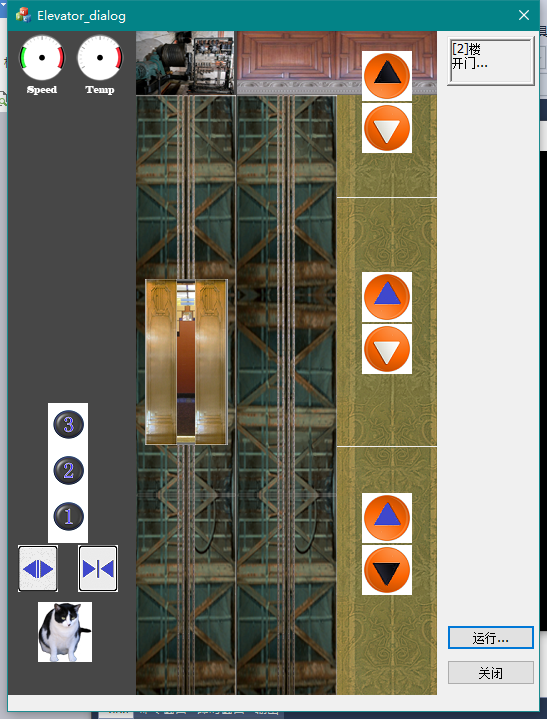
成功。

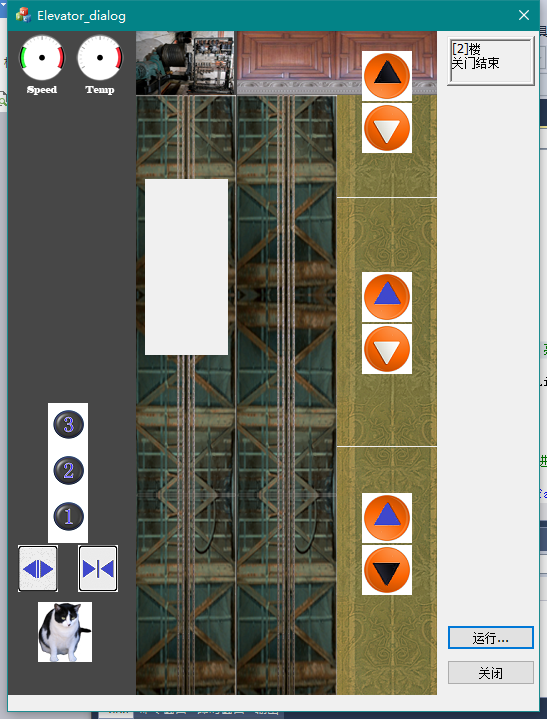
弊端：仅适用于三楼模型。

1. 3.同时按2楼向上下呼叫按钮和三楼呼叫按钮后，电梯在2楼停下开关门两次灯全灭，然后上至三楼。

解决方法：添加printf函数观察发现在二楼之前下楼的情况下up为0，上楼为1

将开门判断条件修改为

if (GetCallLight(nowfloor, 1) && (up || !GetCallLight(3, 0)) || GetCallLight(nowfloor, 0) && (!up || GetCallLight(1, 1)))



成功。

尝试用这种思路修改2问题时发现 用IdleWhatFloorToGoTo(&up)获得up时会对 GoingUpToFloor()产生影响，找不出原因遂对2解决方法不做优化。

弊端：仍旧只适用三楼模型

没有找到获取2与3没有考虑门内按钮，但条件过于复杂是在2做不好。

1. 下到一楼时熄灭2楼向下呼叫的灯，一楼不熄灭开关门两次

解决方法：将if (floor == 1){//判断是否是最低层

SetCallLight(floor, 1, false);

SetPanelFloorLight(1, 0);//关up灯

修改为

if (fabs(GetFloor() - 1) < Lib\_FloorTolerance){//判断是否是最低层

SetCallLight(1, 1, false);

SetPanelFloorLight(1, 0);//关up灯