一、状态机图及其分解描述。

Events：

E1: 门内开门按钮(OpenDoorLight)

E2: 门内关门按钮(CloseDoorLight)

E3: 门内楼层按钮(PanelFloorLight)

E4: 门外up呼叫按钮(Call Light)

E5: 门外down呼叫按钮(Call Light)

二、根据状态机图描述各个状态函数的设计过程。

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。

(S1) 检查E3、E4、E5事件。

静态检测，bool up; 目标楼层=IdleWhatFloorToGoTo(&up);

关闭本层门外up按钮，SetCallLight(); 即消费门外up按钮，防止下一周期重复处理此按钮行为。

(S2) 同(S1)，消费门外down按钮。

1层以上，一定时间无动作，自动下降到1楼。AutoTo1Floor();[其它状态，取消此功能，CancelTo1Floor()]

(S3) 检查E1事件, 开门, 消费开门按钮;

上升 (up && E4事件), 开门，消费门外up按钮

下降 (!up && E5事件), 开门，消费门外down按钮

(S4) 检查E2事件,此时门应该是关闭的,因此仅读取关门灯，并关闭关门灯，即消费按键行为，防止下一周期重复处理该按钮的行为。

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

Moving状态：MovingUp/MovingDown  DoorOpen

(S5) 检查E3、E4、E5事件。动态检测，目标楼层floor=GoingUpToFloor();

if(fabs(GetFloor() - floor) < Lib\_FloorTolerance) 到达目标楼层，停止，开门

消费门外up按钮; 到了最高层Lib\_FloorNum, 消费门外down按钮。消费门内楼层按钮。

(D) 检查E1、E2事件，无动作，消费开/关门按钮。

(S6) 检查E3、E4、E5事件。动态检测，目标楼层=GoingDownToFloor();其它与(S5)类似。

DoorOpen状态: 电梯门打开  DoorClosing

(S7) 检查E2事件，转而关门，GetCloseDoorLight(),SetDoor(); 消费关门按钮。

开门结束后，自动进入关门状态。IsDoorOpen();SetDoor();

检查E1事件, 无动作，消费开门按钮。

DoorClosing状态: 正在关门  DoorOpen

(S8) 检查E1事件，转而开门。GetOpenDoorLight();SetDoor(); 消费关门按钮。

检查E2事件，无动作，消费关门按钮。

DoorClosing状态: 正在关门  Idle

(S9) 关门结束后，进入Idle状态。IsDoorClosed();

三、状态机代码，注释清楚，结构简洁。(在elevator.cpp中完成)

四、运行测试，描述实现的功能及测试结果。

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/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；电梯先上升到3F，开门/关门；然后再到2F，开门/关门。

12、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，然后上升到3F停止，开门/关门，3F的向下呼叫按钮关闭，然后下降到2F，开门/关门，2F的向下呼叫按钮关闭。

13、电梯在2F以上，10s无动作，自动降到1楼。

14、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【延时功能在库函数中实现，不用在状态函数中实现】，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。

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

三层电梯仿真程序状态函数

学号：16040310091

姓名：张又文

日期：2017.5.18

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

/\*空闲状态函数\*/

void StateIdle(int \*state)

{

int floor;

bool up;

floor = IdleWhatFloorToGoTo(&up);/\*静态检测，bool up;\*/

if (GetNearestFloor() != 1)

{

AutoTo1Floor();

}/\*1层以上，一定时间无动作，自动下降到1楼。\*/

if (GetPanelFloorLight(floor) && floor > GetNearestFloor())

{

SetMotorPower(1);

\*state = MovingUp;

}

else if (GetCallLight(floor, true) && floor > GetNearestFloor())

{

SetMotorPower(1);

\*state = MovingUp;

}

else if (GetCallLight(floor, false) && floor > GetNearestFloor())

{

SetMotorPower(1);

\*state = MovingUp;

}

else if (GetOpenDoorLight() || (GetCallLight(GetNearestFloor(), true) || GetCallLight(GetNearestFloor(), false)))

{

SetDoor(GetNearestFloor(), true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

else if (GetCloseDoorLight())

{

SetCloseDoorLight(false);/\*消费\*/

}

else if (GetCallLight(floor, true) && floor < GetNearestFloor())

{

SetMotorPower(-1);

\*state = MovingDown;

}

else if (GetCallLight(floor, false) && floor < GetNearestFloor())

{

SetMotorPower(-1);

\*state = MovingDown;

}

else if (GetPanelFloorLight(floor)&&floor < GetNearestFloor())

{

SetMotorPower(-1);

\*state = MovingDown;

}

}/\*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 StateMovingUp(int \*state)

{

int floor;

floor = GoingUpToFloor();/\*动态检测\*/

if (GetOpenDoorLight())

{

SetOpenDoorLight(false);/\*消费\*/

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);/\*消费\*/

}

if (floor - GetFloor() < Lib\_FloorTolerance)

{

SetMotorPower(0);

SetDoor(floor, true);

SetCallLight(floor,true,false);

SetCallLight(floor, false, false);

SetPanelFloorLight(floor,false);

\*state = DoorOpen;

}/\*到达目标楼层，停止，开门 消费门外up按钮; 到了最高层Lib\_FloorNum, 消费门外down按钮。消费门内楼层按钮。\*/

}/\*Moving up状态\*/

void StateMovingDown(int \*state)

{

int floor;

floor = GoingDownToFloor();/\*动态检测\*/

if (GetOpenDoorLight())

{

SetOpenDoorLight(false);/\*消费\*/

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);/\*消费\*/

}

if (GetFloor() - floor < Lib\_FloorTolerance)

{

SetMotorPower(0);

SetDoor(floor,true);

SetPanelFloorLight(floor,false);

SetCallLight(floor, true, false);

SetCallLight(floor, false, false);

\*state = DoorOpen;

}/\*与moving up状态类似\*/

}/\*Moving down状态\*/

void StateDoorOpen(int \*state)

{

int floor;

floor = GetNearestFloor();

if (GetCloseDoorLight())/\*转而关门\*/

{

SetDoor(floor,false);

SetCloseDoorLight(false);

\*state = DoorClosing;

}

if (IsDoorOpen(floor))/\*开门结束后，自动进入关门状态\*/

{

SetDoor(floor, false);

\*state = DoorClosing;

}

}/\*电梯门打开\*/

void StateDoorClosing(int \*state)

{

int floor;

floor = GetNearestFloor();

if (GetOpenDoorLight())/\*转而开门\*/

{

SetDoor(floor,true);

\*state = DoorOpen;

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

}

if (IsDoorClosed(floor))/\*关门结束后，进入Idle状态。\*/

{

\*state = Idle;

}

}/\*正在关门\*/

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

\* 主控循环： 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())

{

switch (\*state)

{

case Idle:

StateIdle(state);

break;

case MovingUp:

StateMovingUp(state);

break;

case MovingDown:

StateMovingDown(state);

break;

case DoorOpen:

StateDoorOpen(state);

break;

case DoorClosing:

StateDoorClosing(state);

break;

default:

printf("没有这种状态!!!\n");

}

}

}