**Lab2 电梯模拟动态规划**

**唐凯成 PB16001695**

**一.实验要求**

1、通过任意数据结构完成一个电梯模拟系统

·随机生成乘客到达的时间、楼层、目标楼层、最长等待时间

·通过数据结构存放等待队列与电梯里的乘客

·通过时间片模拟电梯运行的过程

1. 提升程序的其他性能

·模块化程序设计（采用类接口）

·考虑程序的鲁棒性

·空间优化：采用动态存储结构

**二.实验平台**

Apple LLVM version 9.1.0 (clang-902.0.39.2)

Visual Studio Code v1.29.0

**三.实验内容**

1、总体设计

对于本次实验，采用了模块化设计，总体分为elevator与queue的类定义文件和main函数程序主模块。

2、elevator

在elevator.h的头文件中申明了每个等待成员的结构及所包含的信息，在elevator类中定义了电梯的结构和成员函数，通过定义一个指针数组来存放不同目标楼层的人，并通过对每个楼层链表的增减来实现乘客的进入与走出，通过进入和输出函数接口来调用。

3、queue

在elevator.h的头文件中同时定义了queue类，同样通过维护一个指针数组来存放每个楼层正在等待的人，通过入队和出队函数来实现队伍中成员的进出。

4、main模块

通过时间片轮转来实现电梯运行的不同流程，电梯运行中共分为以下五个步骤：移动、开门、乘客走出、乘客进入、关门。

五个步骤依次执行并循环，故使用状态机来设计了时间片的轮转，每个时间片为一个时间单位，用于刷新该时间单位内到达的乘客，并使用一个delay变量来实现不同步骤中对时间需要不同的要求。

**四.实验代码**

1、elevator.h：（代码说明已在注释中）

#include<iostream>

#define tp 1 //time piece

#define door\_time 20\*tp //door open and close time

#define get\_in\_time 25\*tp //people enter or exit time

#define test\_time 40\*tp //each time test whether close door autoly

#define reset\_time 300\*tp //time back to basic floor

#define toler\_up\_time 300\*tp //wait upstairs

#define toler\_down\_time 150\*tp //wait downstairs

#define elevater\_speed 15\*tp //each floor time

#define max\_floor 13 //set building floors

#define max\_people 20 //max people number

enum direction{

up, //elevator is up

down //elevator is down

} ;

enum state{

run, //elevator is moving

in, //people get in the elevator

out, //people get out the elevator

opd, //elevator is opening the door

cld //elevator is closing the door

} ;

//declare elevator state

typedef struct p{

int num; //people number

int arr\_time; //the time when people arrive

int arr\_floor; //the floor where people arrive

int des\_floor; //people's destenation

int total\_time; //people total using time

int tol\_time; //people max waiting time

p \*next; //next people pointer

} person;

class elevater

{

private:

person \*floors[max\_floor+1]; //memory structure

public:

direction dir; //elevator direction

int people; //elevator's current people number

int cur\_floor; //elevator current floor

state sta; //elevator current state

elevater(int initfloor); //use init floor to setup elevator

int waittime(int locfloor,int direction); //get floorwaittime

int findnextfloor(int \*que); //find next floor to move to

bool getin(person \*p,int T); //people get in the floor

int getout(int T); //return out people number

};

class quene

{

private:

person\* que[max\_floor+1]; //memory structure

public:

int num[max\_floor+1]; //record each floor's people number

quene();

bool insert(person \*p); //people get in waiting queue

person \*quit(int floor,int T,int \*check); //people quit waiting queue

int waitsum(); //return current total people number in queue

};

2、elevator.cpp

#include"elevator.h"

elevater::elevater(int initfloor)

{

int i;

people=0;

cur\_floor=initfloor;

for(i=1;i<=max\_floor;i++)

{

floors[i]=NULL;

}

dir=up;

sta=run;

}

int elevater::waittime(int locfloor,int direction)

{

if(locfloor==cur\_floor) return 0;

if(cur\_floor>locfloor)

{

if(dir==up)

{

if(direction==dir) return toler\_up\_time;

return (13-cur\_floor+13-locfloor)\*elevater\_speed;

}

else return (cur\_floor-locfloor)\*elevater\_speed;

}

else{

if(direction==dir) return toler\_up\_time;

if(dir==up) return (locfloor-cur\_floor)\*elevater\_speed;

else {

if(direction==dir) return toler\_up\_time;

return (cur\_floor-1+locfloor-1)\*elevater\_speed;

}

}

}

int elevater::findnextfloor(int \*que)

{

if(dir==up)

{

for(int i=cur\_floor+1;i<=max\_floor;i++)

{

if((\*(que+i)>0)||(floors[i]!=NULL))

{

int sub=i-cur\_floor;

cur\_floor=i;

return sub\*elevater\_speed;

}

}

dir=down;

for(int i=cur\_floor;i>=1;i--)

{

if((\*(que+i)>0)||(floors[i]!=NULL))

{

int sub=cur\_floor-i;

cur\_floor=i;

return sub\*elevater\_speed;

}

}

return 0;

}

else{

for(int i=cur\_floor-1;i>=1;i--)

{

if((\*(que+i)>0)||(floors[i]!=NULL))

{

int sub=cur\_floor-i;

cur\_floor=i;

return sub\*elevater\_speed;

}

}

dir=up;

for(int i=cur\_floor;i<=max\_floor;i++)

{

if((\*(que+i)>0)||(floors[i]!=NULL))

{

int sub=i-cur\_floor;

cur\_floor=i;

return sub\*elevater\_speed;

}

}

return 0;

}

}

bool elevater::getin(person \*p,int T)

{

if(people==max\_people) return false;

p->next=NULL;

people++;

person \*fl=floors[p->des\_floor];

if(fl!=NULL)

{

while(fl->next!=NULL)

fl=fl->next;

fl->next=p;

p->next=NULL;

}

else floors[p->des\_floor]=p;

printf("T=%d:No.%d person in,arr\_floor:%d,des\_floor:%d,use time:%d\n",T,p->num,p->arr\_floor,p->des\_floor,p->total\_time);

return true;

}

int elevater::getout(int T)

{

person \*fl=floors[cur\_floor],\*p;

int k=0;

while(fl!=NULL)

{

fl->total\_time=T-fl->arr\_time;

printf("T=%d:No.%d person out,arr\_floor:%d,des\_floor:%d,use time:%d\n",T,fl->num,fl->arr\_floor,fl->des\_floor,fl->total\_time);

//p=fl;

fl=fl->next;

//free(p);

k++;

}

floors[cur\_floor]=NULL;

people-=k;

//printf("out %d\n",k);

return k;

}

quene::quene()

{

for(int i=1;i<=max\_floor;i++)

{

num[i]=0;

que[i]=NULL;

}

}

bool quene::insert(person \*p)

{

if(num[p->arr\_floor]>=max\_floor) return false;

num[p->arr\_floor]++;

person \*fl=que[p->arr\_floor];

if(fl!=NULL)

{

while(fl->next!=NULL)

fl=fl->next;

fl->next=p;

p->next=NULL;

}

else que[p->arr\_floor]=p;

return true;

}

person\* quene::quit(int floor,int T,int \*check)

{

if(num[floor]==0) return NULL;

person \*p=que[floor];

//printf("floor=%d\n",p->arr\_floor);

//printf("kkk\n");

if(p!=NULL)

{

que[floor]=p->next;

//printf("kkk2\n");

if((T-p->arr\_time)>p->tol\_time)

{

printf("T=%d:No.%d person give up waiting\n",p->arr\_time+p->tol\_time,p->num);

\*check=\*check+1;

p=quit(floor,T,check);

}

num[floor]--;

}

//printf("kkk3\n");

return p;

}

int quene::waitsum()

{

int sum=0;

for(int i=1;i<=max\_floor;i++)

{

sum+=num[i];

}

return sum;

}

3、main.cpp

其中onetp()函数内的五个switch即为状态机，构成执行的主要部件：

#include"elevator.h"

#include <math.h>

#include <stdlib.h>

#define max\_text\_people 1000

int T=0;

int Tmax;

int delay=0;

int n;

int rep=1;

int check\_in=0,check\_out=0,check\_giveup=0;

person waitque[max\_text\_people];

elevater \*ele=new elevater(1);

quene \*qu=new quene();

void addperson()

{

int i;

for(i=0;i<n;i++)

if(waitque[i].arr\_time==T)

qu->insert(&waitque[i]);

}

void onetp()

{

addperson();

//printf("delay=%d\n",delay);

if((ele->sta==run)&&(delay>0)&&(\*(qu->num+ele->cur\_floor)>0)&&(ele->people<max\_people))

{

delay=door\_time-delay;

ele->sta=in;

//printf("here\n");

}// interfere the situation: a person arrive while the door is closing.

if(delay==0)

{

switch(ele->sta)

{

case run:

{

delay=ele->findnextfloor(qu->num);

ele->sta=opd;

if((delay==0)&&(\*(qu->num+ele->cur\_floor)==0))

{

ele->sta=run;

if(rep)

printf("T=%d:The elevator is waiting for people at floor %d.\n",T,ele->cur\_floor);

rep=0;

}

else {

rep=1;

printf("T=%d:The elevator is running to floor %d.\n",T,ele->cur\_floor);

}

break;

}

case opd:

{

//printf("opd\n");

printf("T=%d:The elevator is opening door at floor %d.\n",T,ele->cur\_floor);

delay=door\_time;

ele->sta=out;

break;

}

case out:

{

//printf("out\n");

printf("T=%d:People is getting out at floor %d.\n",T,ele->cur\_floor);

int k=ele->getout(T);

delay=k\*get\_in\_time;

check\_out+=k;

ele->sta=in;

break;

}

case in:

{

//printf("in\n");

printf("T=%d:People is getting in at floor %d.\n",T,ele->cur\_floor);

int k=0;

bool m=true;

while((\*(qu->num+ele->cur\_floor)>0)&&(m))

{

if(ele->people<max\_people)

{

person \*p;

p=qu->quit(ele->cur\_floor,T,&check\_giveup);

if(p!=NULL)

{

ele->getin(p,T);

k++;

}

else m=false;

}

else m=false;

}

delay=k\*get\_in\_time;

check\_in+=k;

ele->sta=cld;

break;

}

case cld:

{

//printf("cld\n");

printf("T=%d:The elevator is closing door at floor %d.\n",T,ele->cur\_floor);

delay=door\_time;

ele->sta=run;

break;

}

}

}

else delay--;

}

void randpeople()

{

srand((unsigned)time(NULL));

for(int i=0;i<n;i++)

{

waitque[i].next=NULL;

waitque[i].num=i+1;

waitque[i].arr\_time=(rand()%Tmax)\*tp;

waitque[i].arr\_floor=1+(rand()%(max\_floor));

do waitque[i].des\_floor=1+(rand()%(max\_floor));

while(waitque[i].des\_floor==waitque[i].arr\_floor);

waitque[i].tol\_time=((rand()%(4\*max\_floor\*elevater\_speed))+toler\_up\_time)\*tp;

waitque[i].total\_time=10000\*tp;

printf("No.%d arrive person:T=%d,arr\_floor:%d,des\_floor:%d.\n",waitque[i].num,waitque[i].arr\_time,waitque[i].arr\_floor,waitque[i].des\_floor);

}

}

int main()

{

printf("Please input elevator running max time:\n");

scanf("%d",&Tmax);

Tmax=Tmax\*tp;

printf("Please input random people number:\n");

scanf("%d",&n);

randpeople();

while(T<=Tmax)

{

onetp();

T+=tp;

}

printf("===============================\n");

printf("Report:Total people:%d\nGet in people:%d\nGive up people:%d\nWaiting people:%d\nGet out people:%d\nElevator people:%d\n",n,check\_in,check\_giveup,qu->waitsum(),check\_out,ele->people);

}

**五.实验结果**

1、实验结果

Please input elevator running max time:

1000

Please input random people number:

20

No.1 arrive person:T=402,arr\_floor:8,des\_floor:1.

No.2 arrive person:T=436,arr\_floor:3,des\_floor:5.

No.3 arrive person:T=428,arr\_floor:7,des\_floor:1.

No.4 arrive person:T=487,arr\_floor:2,des\_floor:8.

No.5 arrive person:T=194,arr\_floor:8,des\_floor:13.

No.6 arrive person:T=571,arr\_floor:6,des\_floor:9.

No.7 arrive person:T=150,arr\_floor:5,des\_floor:10.

No.8 arrive person:T=462,arr\_floor:9,des\_floor:13.

No.9 arrive person:T=858,arr\_floor:6,des\_floor:3.

No.10 arrive person:T=684,arr\_floor:5,des\_floor:7.

No.11 arrive person:T=895,arr\_floor:7,des\_floor:10.

No.12 arrive person:T=568,arr\_floor:1,des\_floor:13.

No.13 arrive person:T=896,arr\_floor:3,des\_floor:12.

No.14 arrive person:T=76,arr\_floor:4,des\_floor:6.

No.15 arrive person:T=891,arr\_floor:5,des\_floor:13.

No.16 arrive person:T=471,arr\_floor:4,des\_floor:11.

No.17 arrive person:T=253,arr\_floor:6,des\_floor:10.

No.18 arrive person:T=12,arr\_floor:6,des\_floor:2.

No.19 arrive person:T=326,arr\_floor:3,des\_floor:4.

No.20 arrive person:T=474,arr\_floor:3,des\_floor:11.

T=0:The elevator is waiting for people at floor 1.

T=12:The elevator is running to floor 6.

T=88:The elevator is opening door at floor 6.

T=109:People is getting out at floor 6.

T=110:People is getting in at floor 6.

T=110:No.18 person in,arr\_floor:6,des\_floor:2,use time:10000

T=136:The elevator is closing door at floor 6.

T=157:The elevator is running to floor 5.

T=173:The elevator is opening door at floor 5.

T=194:People is getting out at floor 5.

T=195:People is getting in at floor 5.

T=195:No.7 person in,arr\_floor:5,des\_floor:10,use time:10000

T=221:The elevator is closing door at floor 5.

T=242:The elevator is running to floor 4.

T=258:The elevator is opening door at floor 4.

T=279:People is getting out at floor 4.

T=280:People is getting in at floor 4.

T=280:No.14 person in,arr\_floor:4,des\_floor:6,use time:10000

T=306:The elevator is closing door at floor 4.

T=327:The elevator is running to floor 3.

T=343:The elevator is opening door at floor 3.

T=364:People is getting out at floor 3.

T=365:People is getting in at floor 3.

T=365:No.19 person in,arr\_floor:3,des\_floor:4,use time:10000

T=391:The elevator is closing door at floor 3.

T=412:The elevator is running to floor 2.

T=428:The elevator is opening door at floor 2.

T=449:People is getting out at floor 2.

T=449:No.18 person out,arr\_floor:6,des\_floor:2,use time:437

T=475:People is getting in at floor 2.

T=476:The elevator is closing door at floor 2.

T=497:People is getting in at floor 2.

T=497:No.4 person in,arr\_floor:2,des\_floor:8,use time:10000

T=523:The elevator is closing door at floor 2.

T=544:The elevator is running to floor 3.

T=560:The elevator is opening door at floor 3.

T=581:People is getting out at floor 3.

T=582:People is getting in at floor 3.

T=582:No.2 person in,arr\_floor:3,des\_floor:5,use time:10000

T=582:No.20 person in,arr\_floor:3,des\_floor:11,use time:10000

T=633:The elevator is closing door at floor 3.

T=654:The elevator is running to floor 4.

T=670:The elevator is opening door at floor 4.

T=691:People is getting out at floor 4.

T=691:No.19 person out,arr\_floor:3,des\_floor:4,use time:365

T=717:People is getting in at floor 4.

T=717:No.16 person in,arr\_floor:4,des\_floor:11,use time:10000

T=743:The elevator is closing door at floor 4.

T=764:The elevator is running to floor 5.

T=780:The elevator is opening door at floor 5.

T=801:People is getting out at floor 5.

T=801:No.2 person out,arr\_floor:3,des\_floor:5,use time:365

T=827:People is getting in at floor 5.

T=827:No.10 person in,arr\_floor:5,des\_floor:7,use time:10000

T=853:The elevator is closing door at floor 5.

T=874:The elevator is running to floor 6.

T=890:The elevator is opening door at floor 6.

T=911:People is getting out at floor 6.

T=911:No.14 person out,arr\_floor:4,des\_floor:6,use time:835

T=937:People is getting in at floor 6.

T=774:No.17 person give up waiting

T=937:No.6 person in,arr\_floor:6,des\_floor:9,use time:10000

T=937:No.9 person in,arr\_floor:6,des\_floor:3,use time:10000

T=988:The elevator is closing door at floor 6.

===============================

Report:Total people:20

Get in people:11

Give up people:1

Waiting people:8

Get out people:4

Elevator people:7

2、实验结果分析

对于实验结果，最开始设置了本次模拟电梯的时间和随机生成的人数，然后进入输出部分。输出部分由三部分组成：

1、随机生成的乘客信息

2、日志（时间对应的电梯状态、电梯位置和人员进出）

3、结果统计电梯运行过程中成功乘坐和放弃、仍在等待的人数

第3条中依次输出了总人数、进入电梯人数、放弃乘坐人数、结束时共完成运送人数、结束里的电梯人数。可见对于20人在1000个时间单位间的模拟结果，共有11个人在时间内走入电梯，并有4个人成功通过电梯离开。

**五.实验总结**

本次实验中通过动态规划模拟了一个电梯的运行问题，核心通过指针数组作为数据结构来存放对应数据，并通过状态机模拟了电梯的交替运行。

在寻找数据结构的过程中，起初出现了一系列函数内指针出错的问题，后来找到了具体变量并重新使用。

感谢本次实验，希望后续实验有更多收货！