中 北 大 学

数据结构课程设计说明书

|  |  |  |  |
| --- | --- | --- | --- |
| **学 院、系：** | 软件学院 | | |
| **专 业：** | 软件工程 | | |
| **班 级：** | 20130415 | | |
| **学 生 姓 名：** | 杨毓栋 | **学 号：** | 2013041529 |
| **设 计 题 目：** | 校园安防逃生系统 | | |
|  |  | | |
| **起 迄 日 期:** | 2021年6月22日**~** 2021年7月2日 | | |
| **指 导 教 师:** | 孙乔 | | |

日期: 2021年7月2日

**1 设计目的**

在校园火灾、地震、暴力危险发生时，使用系统可以快速帮助学生规划最佳逃生路线、巧妙避开风险危险楼宇、查询消防系统信息、进行安全普及教育等。

**2 任务概述**

设计一个在校园火灾、暴力事件背景下规划逃生与教育使用。

(1) 校园安防地图可视化；

(2) 安防教育（文件读取）；

(3）查找、修改某处防火器材损耗度，展示全部消防器材损耗度、并包括文件输出；

(4) 校园逃生窗口全部路径可视化以及最短路径查询；

(5) 躲避火源及暴恐路线（随机生成灾害点），路径点去除操作；

(6)使用冒泡排序、快速排序等选择最佳灭火器地点。

**3 本设计采用的数据结构**

整个过程中主要采用图结构、邻接矩阵、迪杰斯特拉算法、冒泡排序、快速排序、线性表的查询、修改。

typedef struct {

int b[20] = {1,0,1,1,0,0,1,1,1,0,1,0,1,1,0}; //是否有消防器材

float c[20] = {0.5,0,0.6,0.9,0,0,0.1,0.1,0.2,0,0.3,0,0.1,0.1,0};

}fire\_data;

typedef struct {

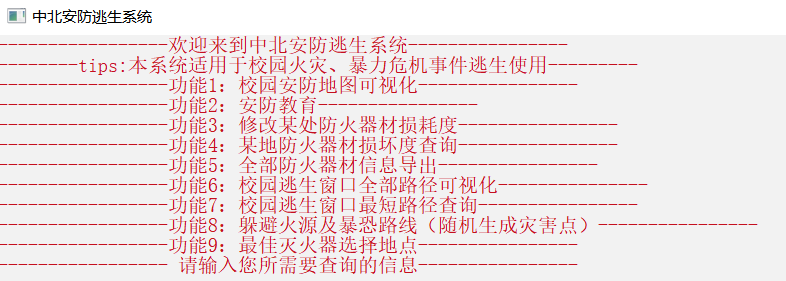
int P[VNUM]; // 辅助数组

int Dist[VNUM]; // 存放最短路径，权值

int Mark[VNUM]; // 标记数组

}route;

**4 系统功能模块结构图及主要函数说明**

图4.1

功能函数：

void print\_all\_dist(route &r,int start) 打印路径

void Quckily\_sort(fire\_data &f, int low, int high) 快速排序

int Bubble(route &r, int q[], int is) 冒泡排序

void disaster(route &r, int start) 随机灾害生成、路径规划

void find\_current\_dist(route &r, int start) 最短路径

int find\_home(fire\_data f,int x) 找到最佳灭火器

float best\_hit\_fire(fire\_data f) 排序后返回最小值

void Dijkstra(route &r,int start) 迪杰斯特拉算法、最短路径

void data\_map() 地图信息打印

void fire\_find(fire\_data &f,int n) 查询灭火器

void fire\_modify(fire\_data &f,int n,float k) 灭火器信息修改

void fire\_all(fire\_data &f) 全部灭火器信息打印及输出文件

void menu() 菜单

void read\_file() 文件读取

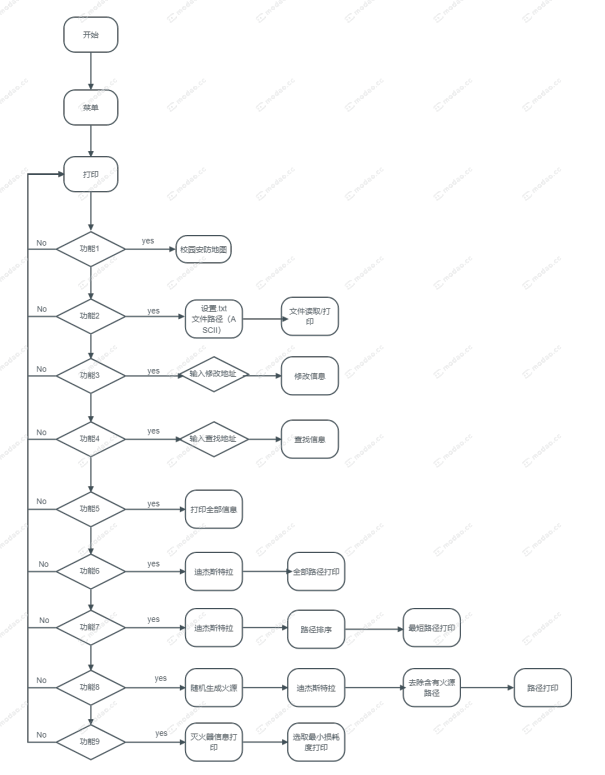
功能流程图

图4.2

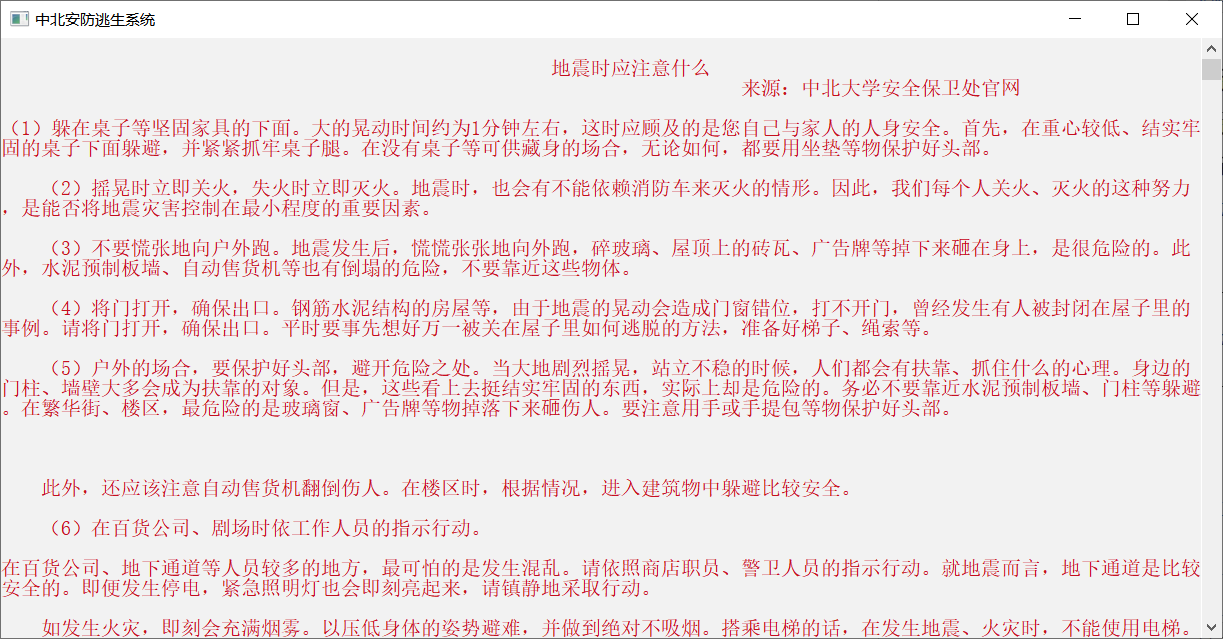
**5 程序运行数据及其结果**

**(1)校园安防地图可视化**

图5.1

键盘输入数据来源，根据地图模拟打印地图信息、结合百度地图规划路径长度并存储于邻接表中

**(2)安防教育**

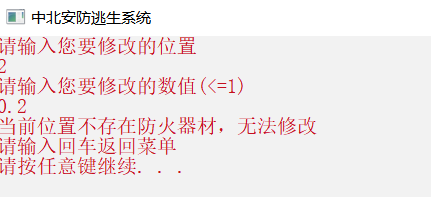
图5.2

采用文件读取流、将txt文件修改格式为ASCI避免中文乱码影响。修改文件存储路径为绝对路径。

1. **查询某处防火器材损耗**

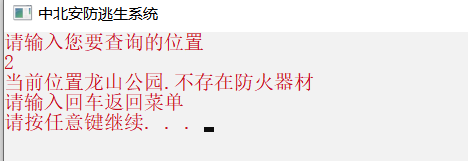
图5.3

当数据真实存在时，修改并打印损耗度，当数据非真实存在时,修改信息驳回。

图5.4

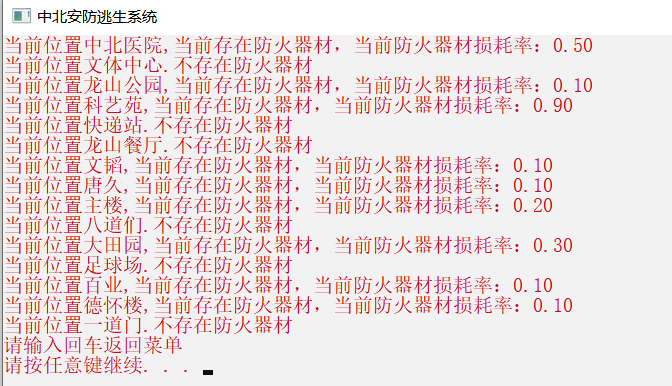
**(4)查询某处防火器材损耗**

当学校更换消防安全设备时，我们对损耗度进行修改(同上两种情况)

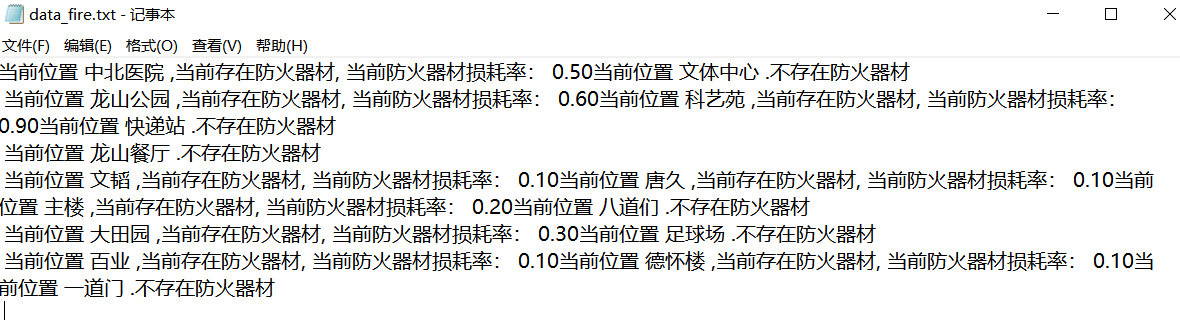
图5.5

**(5)全部防火器材信息导出**

同时，我们可以对全部的消防器材信息进行导出。

图5.6

同时，我们可以考虑将其写入到txt文件中，我们采用文件输出流的方式进行，便于后续导入到数据库中。(甚至我们可以让他变为json数据)

图5.7

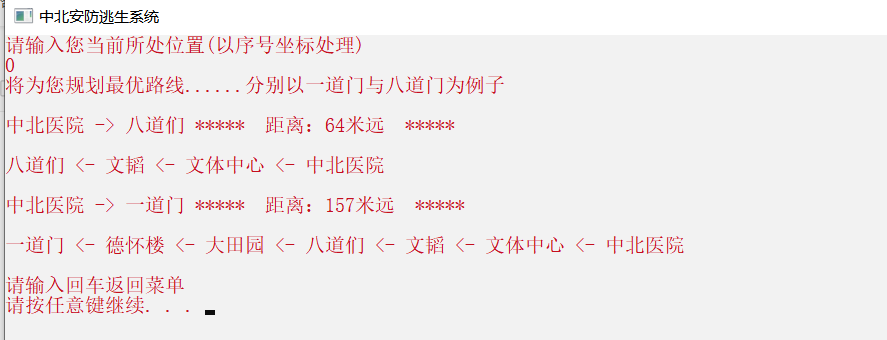
**(6)逃生窗口全部路径可视化**

然后我们进行全部路径可视化，从某地到某地的路劲信息以及距离，全部可视。

图5.8

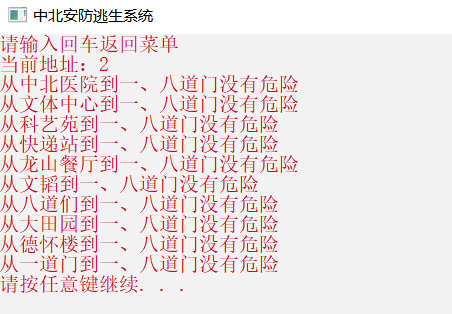
之后，我们输入当前所处位置，可以查询出到逃生窗口的最短路径(分别以一道门和八道们为例子。

**(7)逃生窗口最短路径可视化**

图5.9

**(8) 躲避火源及暴恐路线（随机生成灾害点）**

当暴恐事件发生时，总有一个楼宇充满危险，所以我们随机生成一个危险地址，重新规划逃生窗口，避开危险位置。

图5.10

**(9) 最佳灭火器选择地点**

之后我们利用冒泡或快排进行排序，可以查询到最低损耗度的消防器材存放位置。



图5.11

**6 课程设计心得**

通过这次数据结构的编程题目，使我对C语言有了更深的了解，明白了在C中数据的流动和转移方法。从最初的不知道从何入手到最后编写程序的完成，虽然耗费了我们一定的时间跟精力，但同时我们也收获硕果累累，一方面使我对函数的运用有了更加深刻的了解，对面向对象语言更深层次的掌握和应用。在初始编程的过程中，出现了许多的错误，但通过改正，借阅书本，对程序的运行方式有了进一步体会

通过课程设计让我学习到一个程序正确运行需要各个子程序正确结合,以及严密的语法结构,还要有清晰的流程图。同时，在后期，我还将继续完善这个作品，将这个作品引入c语言easyx库进行整体的地图信息路径动画化，使其更加规范的拥有可以投入生活实用的可行性，并讲整体编译转移到vscode/clion中去。

**附录：**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <windows.h>

#define MAX\_LINE 100010

#define M 99999

#define VNUM 15

const char \*a[] = {"中北医院","文体中心","龙山公园","科艺苑","快递站","龙山餐厅","文韬","唐久","主楼","八道们","大田园"

,"足球场","百业","德怀楼","一道门"};

int Matrix[VNUM][VNUM] = {

{0, 15, M, 20, M,M, M, M, M, M,M, M, M, M, M},

{M, 0, M, M, M,M, 30, M, M, M,M, M, M, M, M},

{M, M, 0, 22, M,M, M, M, M, M,M, M, M, M, M},

{M, M, M, 0, 22,M, M, M, M, M,M, M, M, M, M},

{M, M, M, M, 0,22, M, M, M, M,M, M, M, M, M},

{M, M, M, M, M,0, 23, M, M, M,M, M, M, M, M},

{M, M, M, M, M,M, 0, M, M, 19,M, M, M, M, M},

{M, M, M, M, M,M, M, 0, 22, M,M, M, M, M, M},

{M, M, M, M, M,M, M, M, 0, M,M, M, 45, M, M},

{M, M, M, M, M,M, M, M, M, 0,8, M, M, M, M},

{M, M, M, M, M,M, M, M, M, M,0, M, M, 55, M},

{M, M, M, M, M,M, M, M, M, M,M, 0, 20, M, 40},

{M, M, M, M, M,M, M, M, M, M,M, M, 0, M, M},

{M, M, M, M, M,M, M, M, M, M,M, M, M, 0, 30},

{M, M, M, M, M,M, M, M, M, M,M, M, M, M, 0},

}; //邻接矩阵，查找路径

typedef struct {

int b[20] = {1,0,1,1,0,0,1,1,1,0,1,0,1,1,0}; //是否有消防器材

float c[20] = {0.5,0,0.6,0.9,0,0,0.1,0.1,0.2,0,0.3,0,0.1,0.1,0};

}fire\_data;

typedef struct {

int P[VNUM]; // 辅助数组

int Dist[VNUM]; // 存放最短路径，权值

int Mark[VNUM]; // 标记数组

}route;

void print\_all\_dist(route &r,int start) {

if( (0 <= start) && (start < VNUM) ) {

for(int i = 0; i < VNUM; i++) // 打印最短路径及顶点

{

int p = i;

if(r.Dist[p] == 99999){

printf("当前没有路可寻\n\n");

}else printf("%s -> %s ———距离：%d米远 ———\n\n", a[start], a[p], r.Dist[p]); // 顶点sv到其它顶点的路径

do

{

printf("%s <- ", a[p]);

p = r.P[p];

} while( p != start );

printf("%s\n\n", a[p]);

}

}

}

//快速排序

void Quckily\_sort(fire\_data &f, int low, int high) {

int i, j;

int pivotkey;

f.c[16] = f.c[low]; //第一个位置放满了，后期加入的，所以放到最后

pivotkey = f.c[low];

while (low < high) {

while (f.c[high] >= pivotkey && low < high) high--; //找到第一个小于的放进来

f.c[low] = f.c[high];

while (f.c[high] <= pivotkey && low > high) low--; //找到第一个大于的放进来

f.c[high] = f.c[low];

}

f.c[low] = f.c[16]; //哨兵即中间那个

return;

}

int Bubble(route &r, int q[], int is) {

int str = sizeof(q)/sizeof(q[0]);

printf("%d",str);

for(int i = 0; i < str - 1; i++)

for(int j = i; j < str - 1; j++)

if(q[j] >= q[j + 1]) {

int temp = q[j];

q[j] = q[j + 1];

q[j + 1] = temp;

}

printf("%d",str);

if(is == 0) {

return q[str-1];

}else return q[0];

}

void disaster(route &r, int start) {

int new\_Dist[VNUM];

memset(new\_Dist,0,15);

int t = 0;

int address = rand() % 14 + 1; //生成一个随即损坏地址，但是八道们和一道门我们不能损坏，所以判空一下

printf("当前地址：%d\n",address);

if(address == 9 || address == 14) {

printf("当前校园非常安全，躲在校园内才是最佳策略\n");

}else {

//在我的想法中，这是一个损坏地址，也就是说，如果逃生窗口中包含损坏地址，我就得抛弃掉，选择去离他最远的小房间

for(int i = 0; i < VNUM; i++) {

int p = i;

int flag = 0;

if(r.Dist[p] != 99999) //如果存在这个路线我们再说

{

do

{

if(p == address){

flag = 1;

break;

}else p = r.P[p];

} while( p != start );

if(flag == 0) { //说明我这条路行得通

printf("从%s到一、八道门没有危险\n",a[i]);

new\_Dist[t++] = r.Dist[i];

}

}

}

}

}

void find\_current\_dist(route &r, int start) {

printf("将为您规划最优路线......分别以一道门与八道门为例子\n\n");

if( (0 <= start) && (start < VNUM) ) {

int temp[2] = {9,14}; //一道门和八道们

for(int i = 0; i < 2; i++) {

int p = temp[i];

if(r.Dist[p] == 99999){

printf("当前没有路可寻\n\n");

}else printf("%s -> %s \*\*\*\*\* 距离：%d米远 \*\*\*\*\*\n\n", a[start], a[p], r.Dist[p]); // 顶点sv到其它顶点的路径

do

{

printf("%s <- ", a[p]);

p = r.P[p];

} while( p != start );

printf("%s\n\n", a[p]);

}

}

}

int find\_home(fire\_data f,int x){

for(int i = VNUM - 1; i >= 0; i--)

{

if(x == f.c[i]){

if(i != 9 && i != 14) //排除八道们和一道门

printf("最佳灭火器位于%s,损耗率为%.2f\n",a[i],x);

}

}

}

float best\_hit\_fire(fire\_data f){

//排序后返回最小值

for(int i = 0; i < VNUM - 1; i++)

for(int j = i; j < VNUM - 1; j++)

if(f.c[j] >= f.c[j + 1]) {

int temp = f.c[j];

f.c[j] = f.c[j + 1];

f.c[j + 1] = temp;

}

return f.c[0];

}

void Dijkstra(route &r,int start) {

int i = 0;

int j = 0;

if( (0 <= start) && (start < VNUM) ) //判空

{

for(i = 0; i < VNUM; i++) // 初始化辅助数组

{

r.Dist[i] = Matrix[start][i]; //初始化第一层

r.P[i] = start; //最短路径的顶点的上一个顶点

r.Mark[i] = 0; //标记点

}

r.Mark[start] = 1; //初始点已标记

for(i = 0; i < VNUM; i++) //循环求得sv到某个顶点的最短路径

{

int min = M; //当前离start顶点最短路径

int index = -1; //下个最短路径的顶点

for(j = 0; j < VNUM; j++)

{

if( !r.Mark[j] && (r.Dist[j] < min) ) //如果没有标记并且小于最小值

{

min = r.Dist[j]; //该点位最小值

index = j; //索引过去

}

}

if( index > -1 ) //标记选择

{

r.Mark[index] = 1; //index可以演化为列，也就是说我第一次选择的地方，第二次从这个点走

}

// 更新当前最短路径及顶点

for(j = 0; j < VNUM; j++)

{

if( !r.Mark[j] && (min + Matrix[index][j] < r.Dist[j]) )

{

r.Dist[j] = min + Matrix[index][j];

r.P[j] = index; //我上一次的路径

}

}

}

}

}

void data\_map(){

printf(" ———— ————\n");

printf(" 1中北医院 2文体中心\n");

printf(" ———— ————\n");

printf(" ————— —————— —————— ————— ————\n");

printf(" 3龙山公园 4科艺苑 5快递站 6龙山餐厅 7文韬\n");

printf(" ————— —————— —————— ————— ————\n\n");

printf(" ———— ———— —————\n");

printf(" 8唐久 9主楼 10八道门\n");

printf(" ———— ———— —————\n");

printf(" ————\n");

printf(" 11大田园\n");

printf(" ————\n");

printf(" ———— ————\n");

printf(" 12足球场 13百业\n");

printf(" ———— ————\n");

printf(" —————\n");

printf(" 14德怀楼\n");

printf(" —————\n");

printf(" —————— \n");

printf(" 15一道门 \n");

printf(" —————— \n\n");

}

void fire\_find(fire\_data &f,int n){

int find = n - 1; //找到该处

if(f.b[find] == 1)

printf("当前位置%s,当前存在防火器材，当前防火器材损耗率：%.2f\n",a[n],f.c[find]);

else printf("当前位置%s.不存在防火器材\n",a[n]);

}

void fire\_modify(fire\_data &f,int n,float k) {

int find = n - 1; //我去修改某一处的防火损耗率数据

if(f.b[find] == 1){

f.c[find] = k;

printf("修改成功,当前位置%s,当前存在防火器材，当前防火器材损耗率：%.2f\n",a[n],f.c[find]);

} else {

printf("当前位置不存在防火器材，无法修改\n");

}

}

void fire\_all(fire\_data &f) {

FILE \*fp;

fp = fopen("data\_fire.txt","w");

for(int i = 0; i < 15; i++) {

if(f.b[i] == 0){

fprintf(fp,"%s %s %s ","当前位置",a[i],".不存在防火器材\n");

printf("当前位置%s.不存在防火器材\n",a[i]);

}else {

fprintf(fp,"%s %s %s %s %.2f","当前位置",a[i],",当前存在防火器材,","当前防火器材损耗率：",f.c[i]);

printf("当前位置%s,当前存在防火器材，当前防火器材损耗率：%.2f\n",a[i],f.c[i]);

}

}

}

void menu(){

printf("-----------------欢迎来到中北安防逃生系统----------------\n");

printf("--------tips:本系统适用于校园火灾、暴力危机事件逃生使用---------\n");

printf("-----------------功能1：校园安防地图可视化----------------\n");

printf("-----------------功能2：安防教育----------------\n");

printf("-----------------功能3：修改某处防火器材损耗度----------------\n");

printf("-----------------功能4：某地防火器材损坏度查询----------------\n");

printf("-----------------功能5：全部防火器材信息导出----------------\n");

printf("-----------------功能6：校园逃生窗口全部路径可视化---------------\n");

printf("-----------------功能7：校园逃生窗口最短路径查询----------------\n");

printf("-----------------功能8：躲避火源及暴恐路线（随机生成灾害点）----------------\n");

printf("-----------------功能9：最佳灭火器选择地点----------------\n");

printf("----------------- 请输入您所需要查询的信息----------------\n");

}

void read\_file() {

char buf[MAX\_LINE]; /\*缓冲区\*/

FILE \*fp; /\*文件指针\*/

int len; /\*行字符个数\*/

if((fp = fopen("D:\\data\_map.txt","r")) == NULL)

return;

while(fgets(buf,MAX\_LINE,fp) != NULL)

{

len = strlen(buf);

buf[len-1] = '\0'; /\*去掉换行符\*/

printf("%s\n",buf);

}

}

int main() {

fire\_data f;

route r;

int xg,n;

float mod\_data;

system("title 中北安防逃生系统");

system("color f4");

while(1){

menu();

scanf("%d",&n);

switch(n){

case 1:

system("cls");

data\_map();

printf("请输入回车返回菜单\n");

system("pause");

system("cls");

break;

case 2:

system("cls");

read\_file();

printf("请输入回车返回菜单\n");

system("pause");

system("cls");

break;

case 3:

system("cls");

printf("请输入您要修改的位置\n");

scanf("%d",&xg);

printf("请输入您要修改的数值(<=1)\n");

scanf("%f",&mod\_data);

fire\_modify(f,xg,mod\_data);

printf("请输入回车返回菜单\n");

system("pause");

system("cls");

break;

case 4:

system("cls");

printf("请输入您要查询的位置\n");

scanf("%d",&xg);

fire\_find(f,xg);

printf("请输入回车返回菜单\n");

system("pause");

system("cls");

break;

case 5:

system("cls");

fire\_all(f);

printf("请输入回车返回菜单\n");

system("pause");

system("cls");

break;

case 6:

system("cls");

printf("请输入您当前所处位置(以序号坐标处理)\n");

scanf("%d",&xg);

Dijkstra(r,xg);

print\_all\_dist(r,xg);

printf("请输入回车返回菜单\n");

system("pause");

system("cls");

break;

case 7:

system("cls");

printf("请输入您当前所处位置(以序号坐标处理)\n");

scanf("%d",&xg);

Dijkstra(r,xg);

find\_current\_dist(r,xg);

printf("请输入回车返回菜单\n");

system("pause");

system("cls");

break;

case 8:

system("cls");

printf("请输入回车返回菜单\n");

Dijkstra(r,0);

disaster(r,0);

system("pause");

system("cls");

break;

case 9:

system("cls");

printf("请输入回车返回菜单\n");

Dijkstra(r,0);

float temps = best\_hit\_fire(f);

find\_home(f,temps);

system("pause");

system("cls");

break;

}

}

return 0;

}