**《数据结构》**

**课程设计报告**

|  |  |
| --- | --- |
| 课程名称： | 《数据结构》课程设计 |
| 课程设计题目： | 校园导游咨询 |
| 姓 名： | 杨新莹 |
| 院 系： | 计算机学院 |
| 专 业： | 计算机科学与技术 |
| 班 级： | 22052316 |
| 学 号： | 22050605 |
| 指导教师： | 张灵均 |

2023年12月15日

### 需求分析

#### 功能需求：

（1）设计校园平面图，其中所含景点不少于10个。以图中顶点表示校内各景点，存放景点名称、代号、简介等信息；以边表示路径，存放路径长度等相关信息；

（2）为来访客人提供图中任意景点相关信息的查询；

（3）为来访客人提供途中任意景点的问路查询，即查询任意两个景点之间的一条最短的路径和所有可能的简单路径。

#### 界面需求：

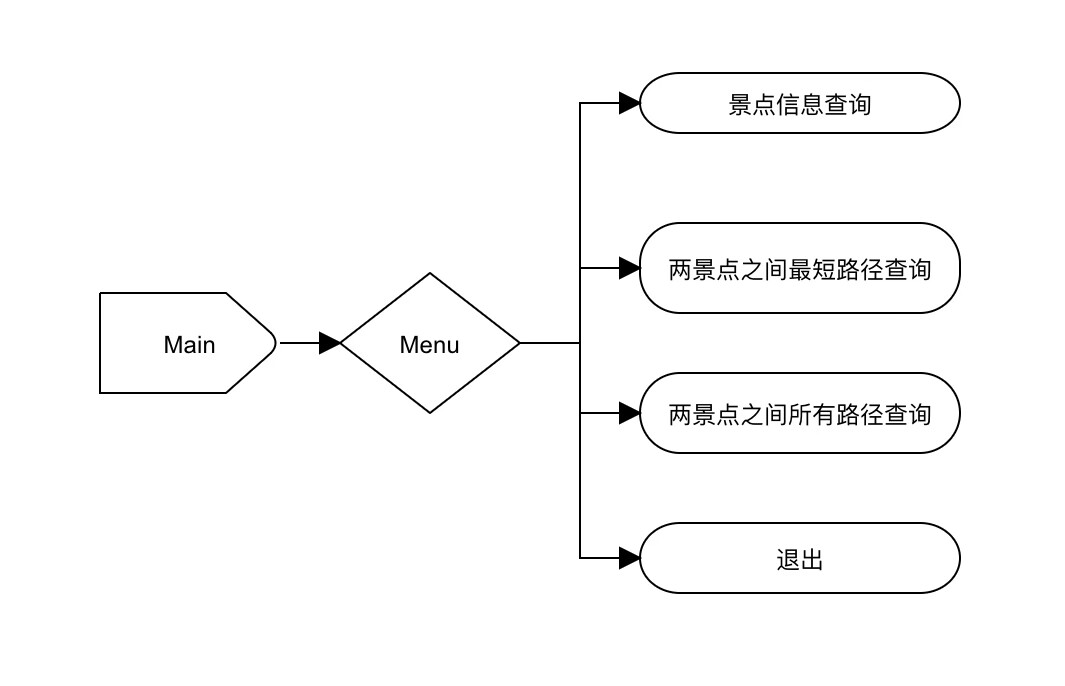
1）打印菜单页面，供用户进行选择

2）打印仿真校园平面图

3）输出用户所查询的信息

### 概要设计

#### 模块层次结构设计



#### 接口设计

void GotoXY(int x, int y);

void Dis\_map();

void Dis\_map1();

void Creatspot();

void Search\_spot\_info();

void Floyd();

void Floyd\_print(int s, int e);

void Dfs\_allpath(int s, int e);

int menu();

int main();

**界面设计**

·菜单页面

system("cls");

GotoXY(38, 8);

printf("欢迎使用杭州电子科技大学导游咨询系统");

GotoXY(45, 10);

printf("1.景点信息查询");

GotoXY(45, 12);

printf("2.两景点之间最短路查询");

GotoXY(45, 14);

printf("3.两景点间所有路径查询");

GotoXY(45, 16);

printf("4. 退出");

GotoXY(40, 18);

printf("请输入您需要进行的操作[ ]\b\b");

·输出校园平面图

void Dis\_map()

{

GotoXY(25, 1);

printf("|----------------------------\*杭州电子科技大学校园平面图\*------------------------------------");

GotoXY(25, 2);

printf("| 1)图书馆 13）学生活动中心 ");

GotoXY(25, 3);

printf("| 7）六教 ◎ 8）七教 ◎ ");

GotoXY(25, 4);

printf("| ◎ ◎---------------| ");

GotoXY(25, 5);

printf("| |------------| | ");

GotoXY(25, 6);

printf("| 2）体育馆 12）十二教 11）十一教 ");

GotoXY(25, 7);

printf("| ◎ ◎ ◎ ");

GotoXY(25, 8);

printf("| 6）四教 5）三教 | 10）十教 9）九教 ");

GotoXY(25, 9);

printf("| ◎ ◎ ◎ ◎ ");

GotoXY(25, 10);

printf("| -------|------------------—-----|----------------| ");

GotoXY(25, 11);

printf("| | | | ");

GotoXY(25, 12);

printf("| | | | ");

GotoXY(25, 13);

printf("| 4）二教 3）一教 ");

GotoXY(25, 14);

printf("| ◎ ◎ ");

GotoXY(25, 15);

printf("| | | ");

GotoXY(25, 16);

printf("| 14）行政楼 15）月雅湖 ");

GotoXY(25, 17);

printf(" ◎ ◎ ");

GotoXY(25, 18);

printf(" 16）南大门 ");

GotoXY(25, 19);

printf(" ◎ ");

}

#### 数据结构设计

struct spot//景点信息结构体

{

int num;//景点编号

string name;//景点名称

string info;//景点介绍

};

struct maps

{

int n;//点数

int m;//边数

spot v[M];//顶点

int e[M][M];//邻接矩阵

} g; //景点图的结构体

### 详细设计

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<stack>

#include<iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include<Windows.h>

#include <sstream>

#define INF 99999

#define M 20

int dist[M][M];///距离

int path[M][M];///路径

int Stack[M];//路径栈

int top=0;//栈顶

int counts=1;//记录路径数

int visited[M];//标记数组

int dis;

using namespace std;

struct spot//景点信息结构体

{

int num;//景点编号

string name;//景点名称

string info;//景点介绍

};

struct maps

{

int n;//点数

int m;//边数

spot v[M];//顶点

int e[M][M];//邻接矩阵

} g; //景点图的结构体

void GotoXY(int x, int y) {

HANDLE hout;

COORD cor;

hout = GetStdHandle(STD\_OUTPUT\_HANDLE);

cor.X = x;

cor.Y = y;

SetConsoleCursorPosition(hout, cor);

}

void Dis\_map()

{

GotoXY(25, 1);

printf("|----------------------------\*杭州电子科技大学校园平面图\*------------------------------------");

GotoXY(25, 2);

printf("| 1)图书馆 13）学生活动中心 ");

GotoXY(25, 3);

printf("| 7）六教 ◎ 8）七教 ◎ ");

GotoXY(25, 4);

printf("| ◎ ◎---------------| ");

GotoXY(25, 5);

printf("| |------------| | ");

GotoXY(25, 6);

printf("| 2）体育馆 12）十二教 11）十一教 ");

GotoXY(25, 7);

printf("| ◎ ◎ ◎ ");

GotoXY(25, 8);

printf("| 6）四教 5）三教 | 10）十教 9）九教 ");

GotoXY(25, 9);

printf("| ◎ ◎ ◎ ◎ ");

GotoXY(25, 10);

printf("| -------|------------------—-----|----------------| ");

GotoXY(25, 11);

printf("| | | | ");

GotoXY(25, 12);

printf("| | | | ");

GotoXY(25, 13);

printf("| 4）二教 3）一教 ");

GotoXY(25, 14);

printf("| ◎ ◎ ");

GotoXY(25, 15);

printf("| | | ");

GotoXY(25, 16);

printf("| 14）行政楼 15）月雅湖 ");

GotoXY(25, 17);

printf(" ◎ ◎ ");

GotoXY(25, 18);

printf(" 16）南大门 ");

GotoXY(25, 19);

printf(" ◎ ");

}

void Dis\_map1()

{

GotoXY(15, 1);

printf("|----------------------------\*杭州电子科技大学校园平面图\*------------------------------------");

GotoXY(15, 2);

printf("| 1)图书馆 13）学生活动中心 ");

GotoXY(15, 3);

printf("| 7）六教 ◎ 8）七教 ◎ ");

GotoXY(15, 4);

printf("| ◎ ◎---------------| ");

GotoXY(15, 5);

printf("| |------------| | ");

GotoXY(15, 6);

printf("| 2）体育馆 12）十二教 11）十一教 ");

GotoXY(15, 7);

printf("| ◎ ◎ ◎ ");

GotoXY(15, 8);

printf("| 6）四教 5）三教 | 10）十教 9）九教 ");

GotoXY(15, 9);

printf("| ◎ ◎ ◎ ◎ ");

GotoXY(15, 10);

printf("| -------|------------------—-----|----------------| ");

GotoXY(15, 11);

printf("| | | | ");

GotoXY(15, 12);

printf("| | | | ");

GotoXY(15, 13);

printf("| 4）二教 3）一教 ");

GotoXY(15, 14);

printf("| ◎ ◎ ");

GotoXY(15, 15);

printf("| | | ");

GotoXY(15, 16);

printf("| 14）行政楼 15）月雅湖 ");

GotoXY(15, 17);

printf("| ◎ ◎ ");

GotoXY(15, 18);

printf("| 16）南大门 ");

GotoXY(15, 19);

printf("| ◎ ");

}

void Creatspot()

{

ifstream File("校园平面图.txt");

if (!File.is\_open()) {

cout << "无法打开文件" << endl;

return;

}

File >> g.n;

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

{

File >> g.v[i].num >> g.v[i].name >> g.v[i].info;

}

File >> g.m;

for (int i = 0; i < g.n; i++) ///初始化邻接矩阵

{

for (int j = 0; j < g.n; j++)

{

g.e[i][j] = INF;

}

}

for (int i = 0; i < g.m; i++)

{

int a, b,d;

File >> a >> b>>d;

g.e[a-1][b-1] = g.e[b-1][a-1] = d;

}

}

void Search\_spot\_info()

{

while (1)

{

system("cls");

int your\_search;

printf("杭州电子科技大学的景点有：\n");

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

{

cout << setw(3) << g.v[i].num << " " << g.v[i].name << endl;

}

Dis\_map();

printf("\n\n\n请输入你想要查询的景点编号(按0退出)：\n");

cin >> your\_search;

getchar();

if (your\_search == 0)

{

break;

}

else if (your\_search < 0 || your\_search>g.n)

{

printf("输入有误，请重新输入！！！\n\n");

continue;

}

else

{

cout << g.v[your\_search - 1].num <<" " <<g.v[your\_search - 1].name <<"--" <<g.v[your\_search - 1].info << endl;

system("pause");

}

}

return;

}

void Floyd()

{

int i, j, k;

for (i = 0; i < g.n; i++) ///初始化距离与路径矩阵

{

for (j = 0; j < g.n; j++)

{

dist[i][j] = g.e[i][j];

if (i != j && dist[i][j] < INF)

{

path[i][j] = i;

}

else

{

path[i][j] = -1;///-1代表不可达

}

}

}

for (k = 0; k < g.n; k++)//中间顶点

{

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

{

for (j = 0; j < g.n; j++)

{

if ( (dist[i][k] + dist[k][j])< dist[i][j] )

{

dist[i][j] = dist[i][k] + dist[k][j];///更新

path[i][j] = k; ///path用于记录最短路径上的结点\*/

}

}

}

}

return;

//最后就会得到距离矩阵和路径矩阵

}

void Floyd\_print(int s, int e)

{

if (path[s][e] == -1 || path[s][e] == e || path[s][e] == s)///递归终止条件，不可到达/到达终点或者起点

{

return;

}

else

{

Floyd\_print(s, path[s][e]);// 递归调用，打印从顶点 s 到 path[s][e] 的最短路径

cout << g.v[path[s][e]].name << "->";

Floyd\_print(path[s][e], e);// 递归调用，打印从 path[s][e] 到顶点 e 的最短路径

}

}

//查询任意两个景点之间所有可能的简单路径。深度优先搜索

void Dfs\_allpath(int s, int e)

{

dis = 0;//初始化路径长度

int i, j;

Stack[top] = s;

top++; //栈顶指针的移动

visited[s] = 1;//标记入栈

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

{

if (g.e[s][i] > 0 && g.e[s][i] != INF && !visited[i])

{

//两点可达且未被访问

if (i == e)///DFS到了终点，打印路径

{

printf("第%d条路:", counts++);

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

{

cout << g.v[Stack[j]].name << "->";

if (j < top - 1)///统计路径长度

{

dis = dis + g.e[Stack[j]][Stack[j + 1]];

}

}

dis = dis + g.e[Stack[top - 1]][e];

cout << g.v[e].name << endl;

printf("总长度是：%dm\n\n", dis);

}

else///不是终点接着DFS

{

Dfs\_allpath(i, e);

top--;///支路全被访问一遍,顶点出栈

visited[i] = 0;///出栈点标记为已出栈，允许下次访问

}

}

}

}

int menu()

{

system("cls");

GotoXY(38, 8);

printf("欢迎使用杭州电子科技大学导游咨询系统");

GotoXY(45, 10);

printf("1.景点信息查询");

GotoXY(45, 12);

printf("2.两景点之间最短路查询");

GotoXY(45, 14);

printf("3.两景点间所有路径查询");

GotoXY(45, 16);

printf("4. 退出");

GotoXY(40, 18);

printf("请输入您需要进行的操作[ ]\b\b");

char ch;

int result = 4;

ch = getchar();

switch (ch) {

case '1': result = 1; break;

case '2': result = 2; break;

case '3': result = 3; break;

case '4': result = 4; break;

default:

result = -1;

GotoXY(38, 24);

cout << "无效的选择，请重新输入。\n";

GotoXY(38, 26);

system("pause");

}

getchar();

system("cls");

return result;

}

int main()

{

int i, n;

int start, ends;

Creatspot();

int end = 1;

int result = -1;

while (end) {

result = menu();

switch (result)

{

case 1:

Search\_spot\_info();

break;

case 2:

Dis\_map1();

printf("\n----------------------------------------------------------------------------------------------------");

printf("\n请输入当前所在的景点(景点前方的编号）：\n");

scanf\_s("%d", &start);

printf("请输入终点的景点：\n");

scanf\_s("%d", &ends);

printf("\n----------------------------------------------------------------------------------------------------");

if (start<1 || start>g.n || ends<1 || ends>g.n) {

cout << endl<<"输入错误"<<endl;

system("pause");

getchar();

continue;

}

Floyd();///弗洛伊德

cout <<endl<< "从 ◎" << g.v[start - 1].name << " 到◎ " << g.v[ends - 1].name << " 最短距离是：" << dist[start - 1][ends - 1] << endl;

cout << "路径是：";

cout << g.v[start - 1].name << "->";

Floyd\_print(start - 1, ends - 1);

cout << g.v[ends - 1].name << endl<<endl;

system("pause");

getchar();

break;

case 3:

dis = 0;

counts = 1;

top = 0;

Dis\_map1();

printf("\n----------------------------------------------------------------------------------------------------");

printf("\n请输入当前所在的景点(景点前方的编号）：\n");

scanf\_s("%d", &start);

printf("请输入终点的景点：\n");

scanf\_s("%d", &ends);

printf("\n----------------------------------------------------------------------------------------------------\n");

Dfs\_allpath(start - 1, ends - 1);

system("pause");

getchar();

break;

case 4:

cout << "正在退出系统......";

end = 0;

break;

default:

break;

}

}

return 0;

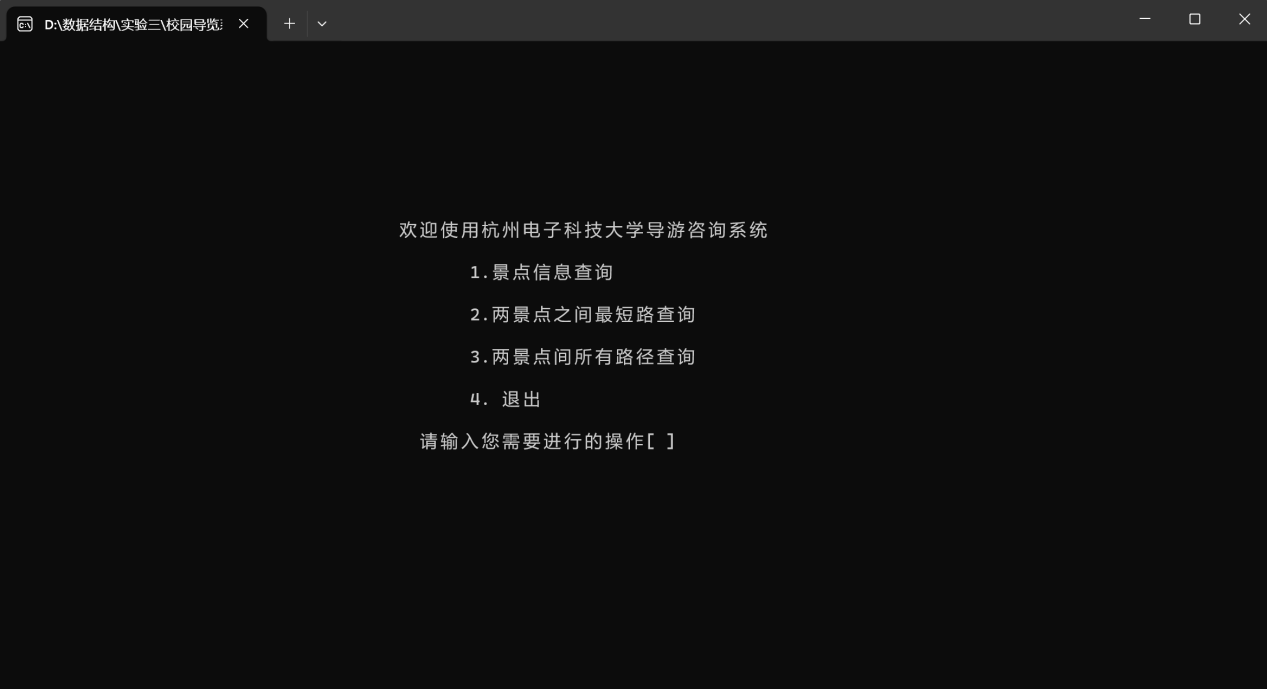
}

### 调试分析

1. 本次作业要求设计一个校园导览系统，一是要求实现找到两个景点之间的最短路径，核心在于采用floyd算法，通过一定的规则逐步更新每个节点的最短路径估计值，直到每个节点的最短路径估计值收敛为止。二是要求实现找出两个景点之家的所有简单路径，核心是采用深度优先算法，相对于起点选择一条路走到底，发现不行就返回选择的节点换一条路试，直到试出一条能到达终点的路。
2. 在实现最短路径算法时，需要注意算法的收敛条件和路径更新逻辑是否正确，以及算法是否能够正确处理各个节点之间的路径关系。
3. 在实现深度优先算法时，需确保算法能够正确地遍历所有可能的路径，并能够正确处理路径的选择和回溯，以获取所有简单路径。

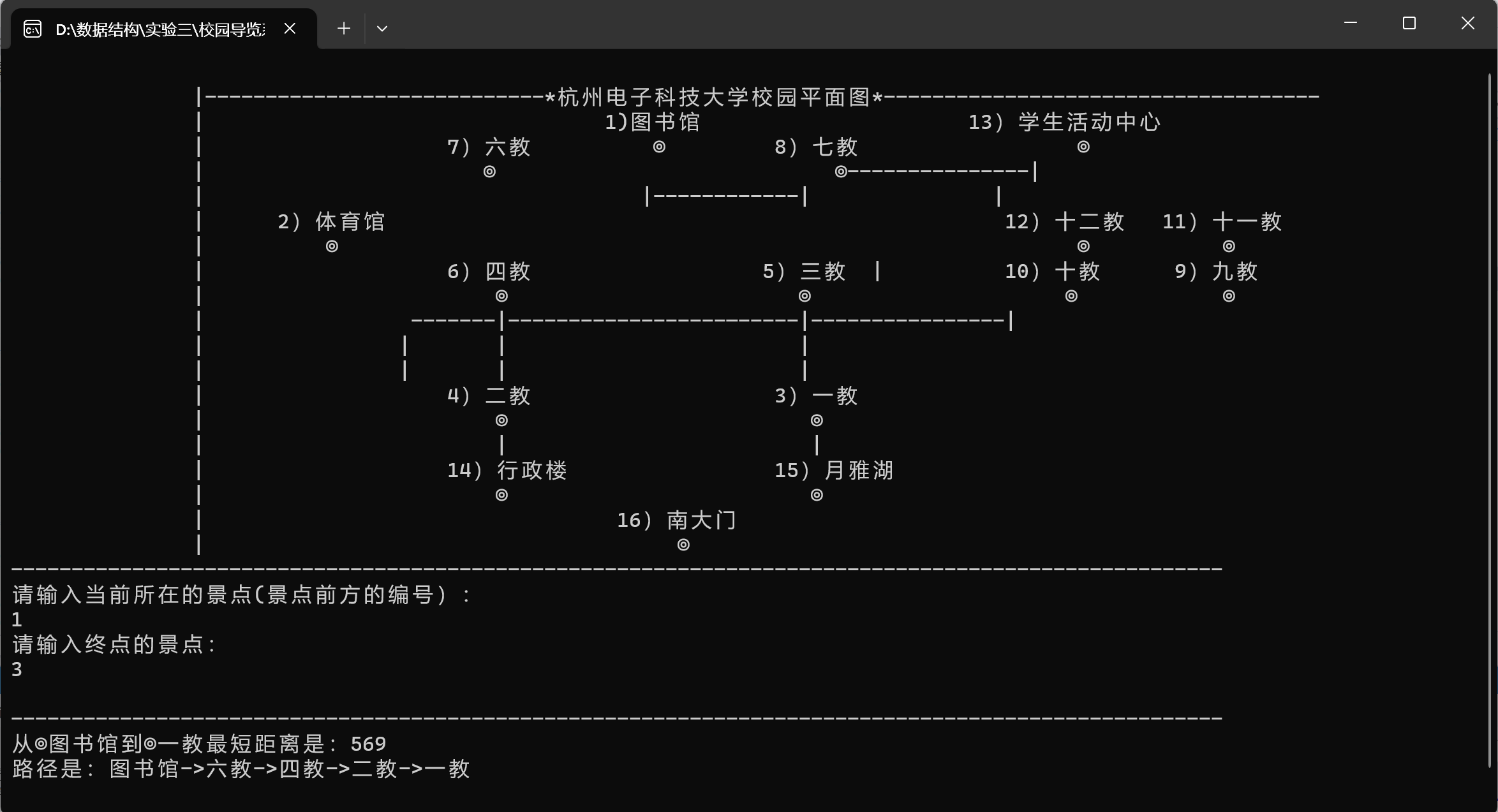
### 用户手册

1. 本程序的执行文件为：校园导览系统.exe
2. 进入演示程序后，将显示如下的菜单界面，进行选择



1. 景点信息查询

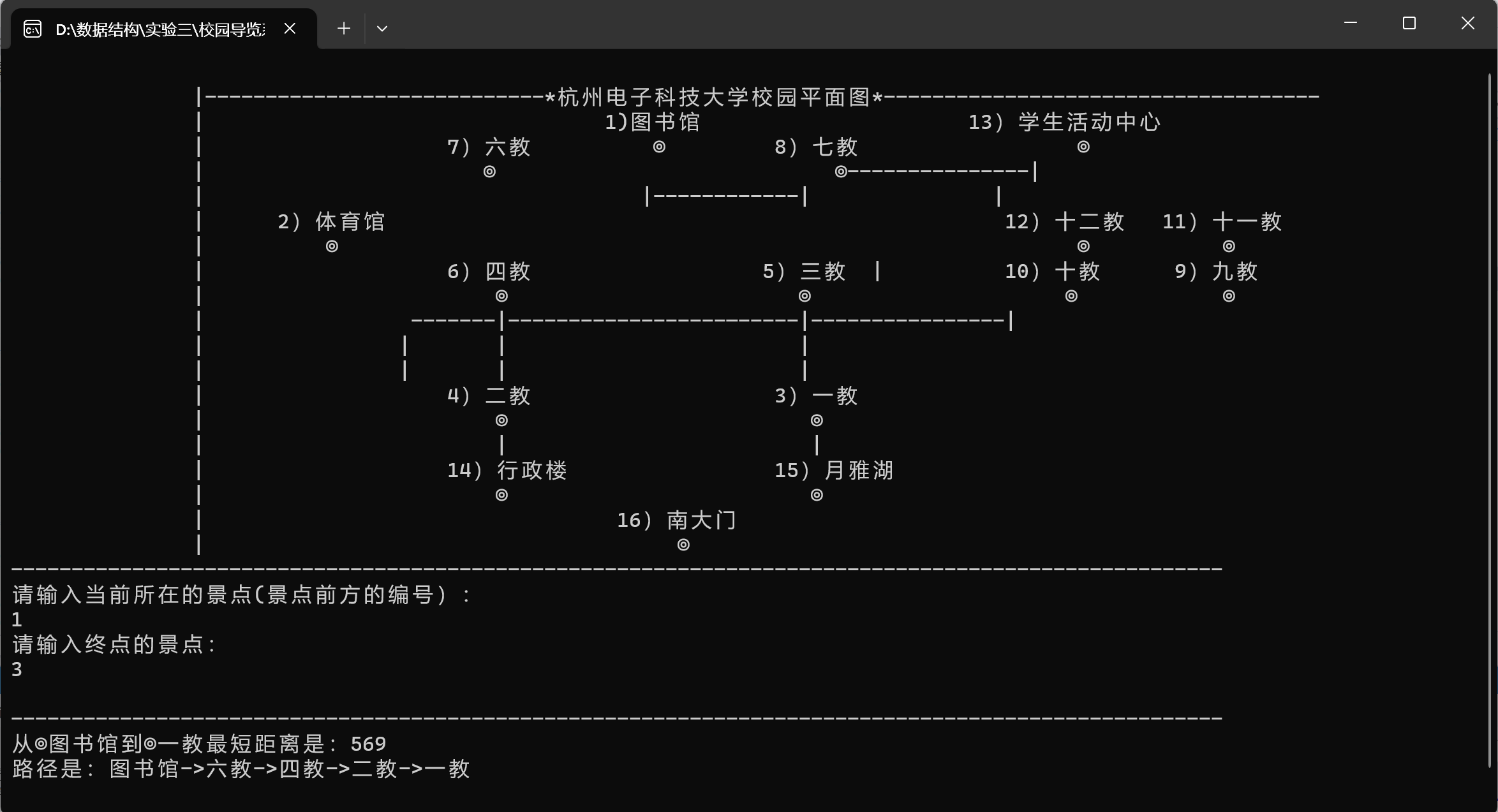


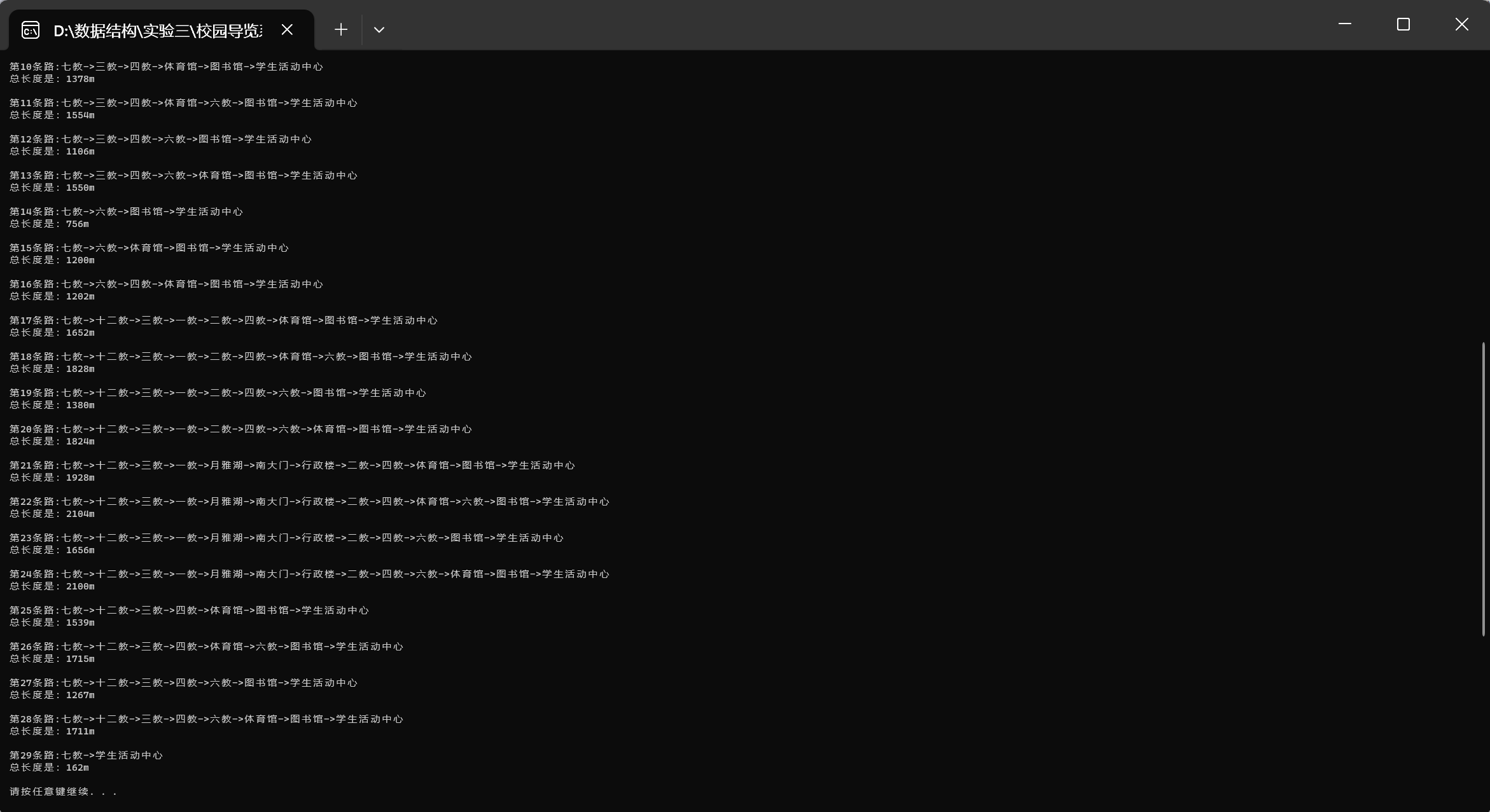
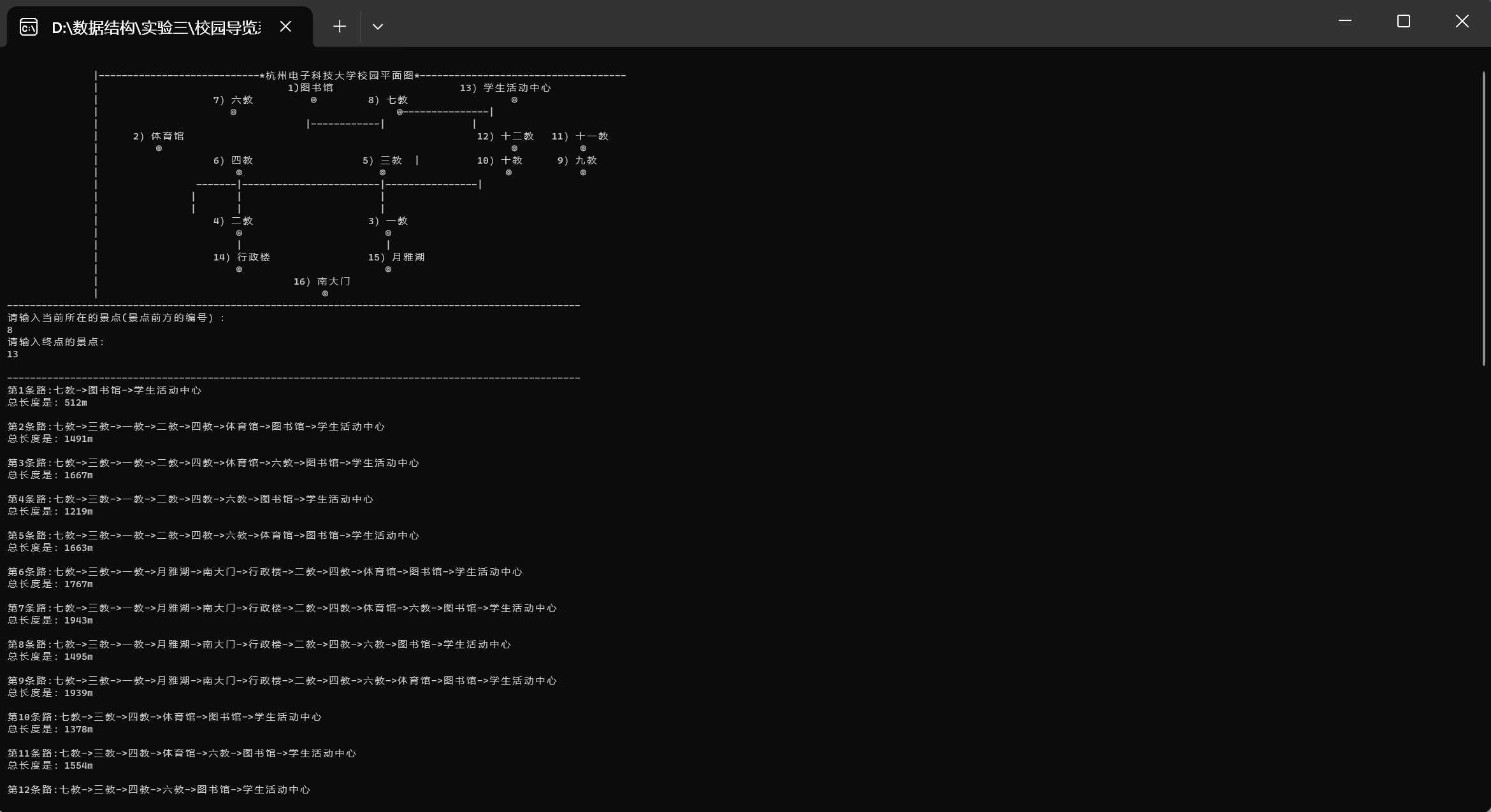
1. 两点之间最短路径查询，先后输入出发点和终点进行路径查询（两点之间的所有简单路径查询也是一样的）

### 测试结果

1. 景点信息查询，左侧显示杭电有的景点，右侧显示校园的仿真平面图，输入查询的景点编号即可查询得到相应的景点信息



1. 查询两个景点之间的最短路径，上方显示校园的仿真平面图，依次输入出发点和终点
2. 查询两个景点之间的所有简单路径，上方显示校园的仿真平面图，依次输入出发点和终点，输出所有简单路径



1. 退出系统

PS：详细的运行结果请打开校园导览系统.exe文件

### 附录

源程序文件名清单：main.cpp，校园导览系统.exe