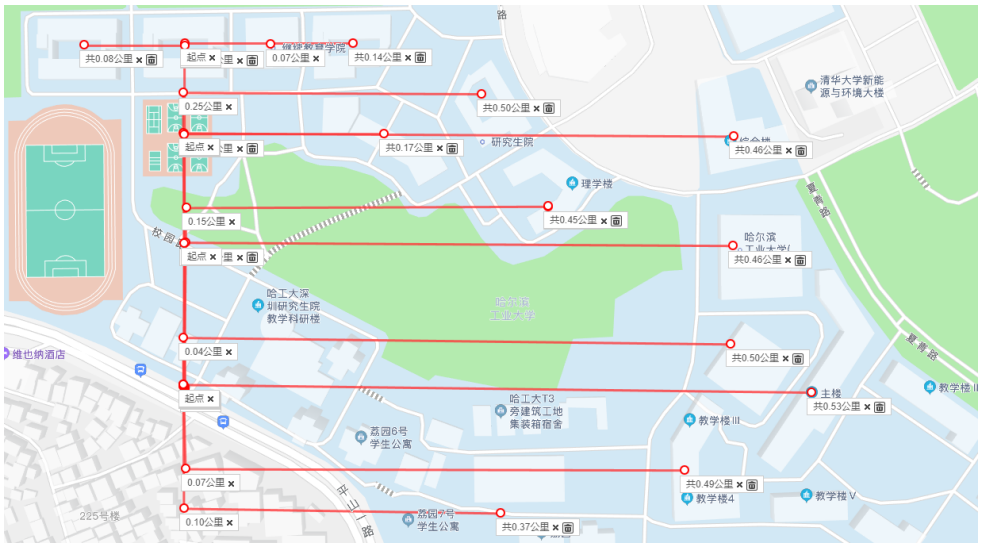
数据结构与算法(C++)大作业实验报告

姓名： _____ **学号：** _____

学院： _____ **专业：** _____

一、 校园导游地图简图

部分坐标测量（以平山一路大门为坐标原点建立坐标系）



部分路径长度测量

二、 运行截图（完整测试题目所要求的功能）

所要求的全部功能已经实现，整个程序非常方便查询和退出

功能一：校园简图
为了保证视觉效果，请将运行窗口调大一点

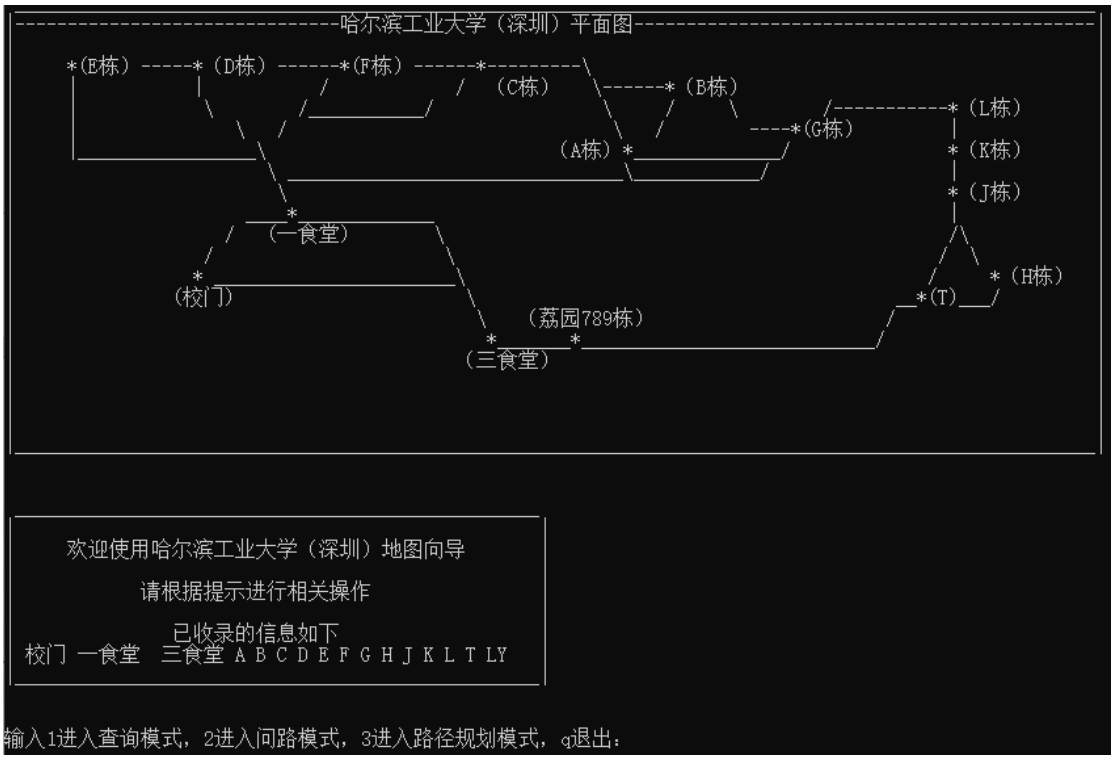


图 1 地图测试

功能二：实现已经收录的各建筑信息查询，如果输入的不存在（没有被收录），提示重新输入

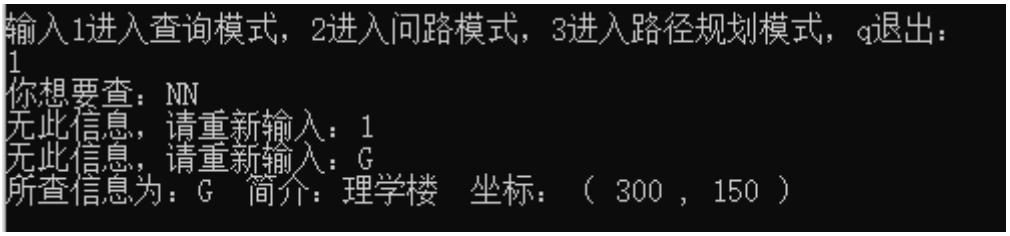


图 2 建筑信息查询功能

功能三：实现从某个起点到其他所有地方的最短路径规划（floyd 算法），并且按照路径长短排序输出，也有输入错误提示功能

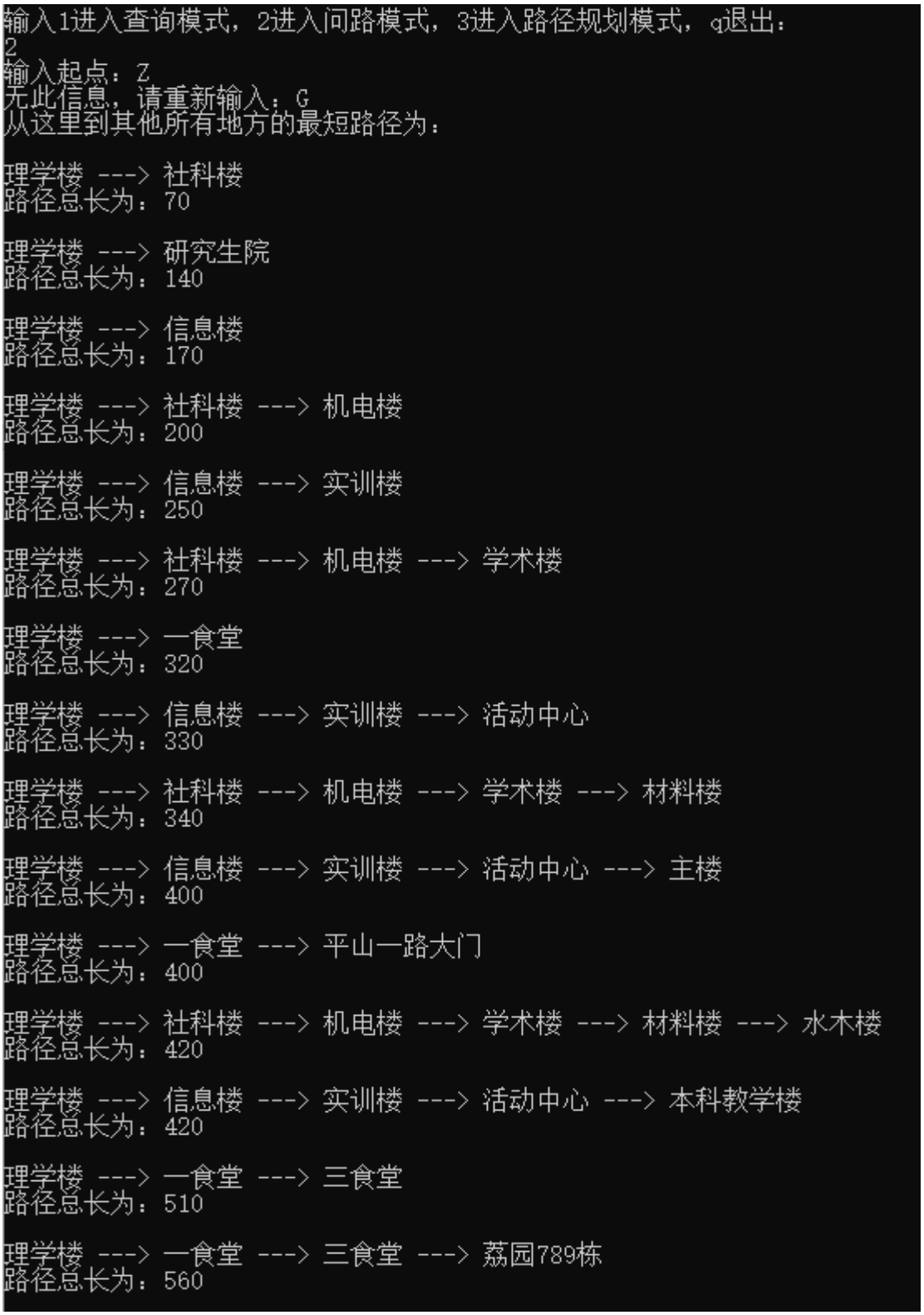


图 3 问路功能

功能四：实现指定起点到终点、指定经过多少建筑的所有路径(DFS 递归)，按路径长度排序，并给出推荐最短路径的导航信息（方向+距离），也有建筑输入错误提示。当输入所要求的途径点数量没有找到路径时，提示没有符合要求的路径

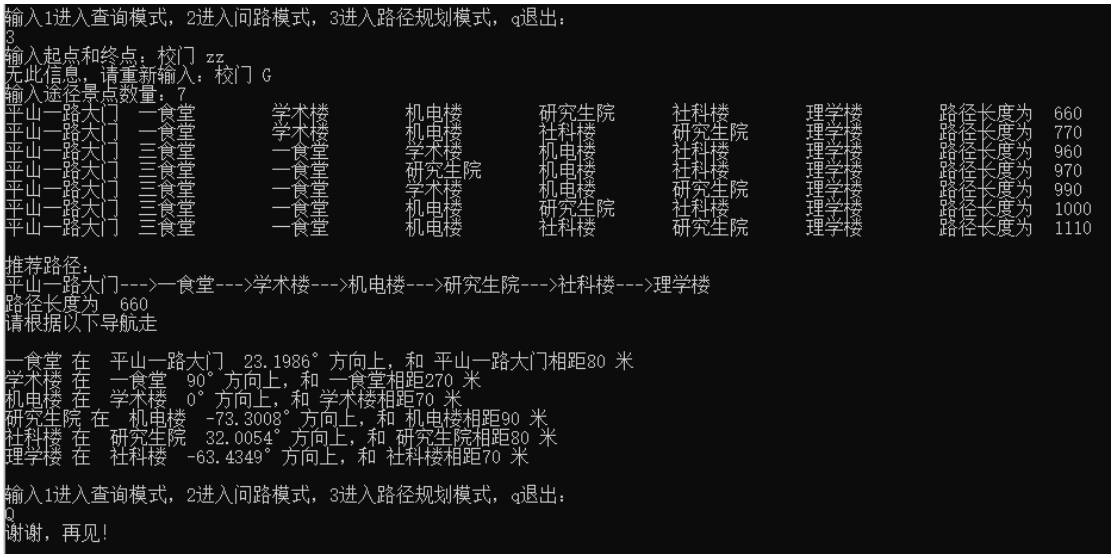


图 4 路径规划功能

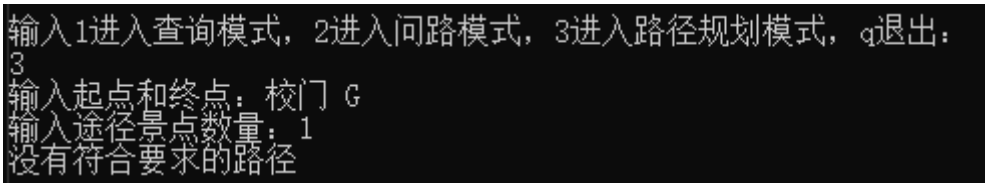


图 5 运行结果提示

三、 程序代码

```
1.  #include <iostream>
2.  #include <cmath>
3.  #include <string>
4.  #include <vector>
5.  #include <iomanip>
6.
7.  using namespace std;
8.
9.
```

```cpp
struct building//建筑信息结构体
{
    string name;//建筑名称
    int x;//坐标
    int y;
    string information;//建筑信息
};

const int BUILDING_NUMBER=16;

//结构数组存储所有建筑信息
building arr[BUILDING_NUMBER] =
{
  //序号，名称，坐标，信息
    {"校门",0,0,"平山一路大门"},
    {"一食堂",70,30,"一食堂"},
    {"三食堂",210,-100,"三食堂"},
    {"A",170,200,"研究生院"},
    {"B",250,250,"社科楼"},
    {"C",140,300,"机电楼"},
    {"D",0,300,"材料楼"},
    {"E",-80,300,"水木楼"},
    {"F",70,300,"学术楼"},
    {"G",300,150,"理学楼"},
    {"H",0,530,"主楼"},
    {"J",460,40,"活动中心"},
    {"K",460,120,"实训楼"},
    {"L",460,200,"信息楼"},
    {"T",420,-70,"本科教学楼"},
    {"LY",270,-100,"荔园 789 栋"},

};

//各点路径长度
const int MAX_NUMBER = 9999;//定义一个很长的路径代表不通,直接用 INT_MAX 会导致后续计算溢出
 int weight[BUILDING_NUMBER][BUILDING_NUMBER] = {
    {MAX_NUMBER,   80   ,       230,        MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER},
    {80,            MAX_NUMBER, 190,        260,        MAX_NUMBER, 340,        MAX_NUMBER, 310,        270,        320,        MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER},
```

```
48.     {230,           190,        MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMB
    ER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER,
    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, 50},
49.     {MAX_NUMBER,    260,        MAX_NUMBER, MAX_NUMBER, 80,         90,
        MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, 140,        MAX_NUMBER, MAX_NUMBER,
    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER},
50.     {MAX_NUMBER,    MAX_NUMBER, MAX_NUMBER, 80,         MAX_NUMBER, 130,
        MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, 70,         MAX_NUMBER, MAX_NUMBER,
    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER},
51.     {MAX_NUMBER,    340,        MAX_NUMBER, 90,         130,        MAX_NUMB
    ER, MAX_NUMBER, MAX_NUMBER, 70,         MAX_NUMBER, MAX_NUMBER, MAX_NUMBER,
    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER},
52.     {MAX_NUMBER,    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMB
    ER, MAX_NUMBER, 80,         70,         MAX_NUMBER, MAX_NUMBER, MAX_NUMBER,
    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER},
53.     {MAX_NUMBER,    310,        MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMB
    ER, 80,         MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER,
    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER},
54.     {MAX_NUMBER,    270,        MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, 70,
        70,         MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER,
    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER},
55.     {MAX_NUMBER,    320,        MAX_NUMBER, 140,        70,         MAX_NUMB
    ER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER,
    MAX_NUMBER, 170,        MAX_NUMBER, MAX_NUMBER},
56.     {MAX_NUMBER,    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMB
    ER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, 70,
    MAX_NUMBER, MAX_NUMBER, 170,        MAX_NUMBER},
57.     {MAX_NUMBER,    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMB
    ER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, 70,         MAX_NUMBER,
    80,         MAX_NUMBER, 90,         MAX_NUMBER},
58.     {MAX_NUMBER,    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMB
    ER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, 80,
    MAX_NUMBER, 80,         MAX_NUMBER, MAX_NUMBER},
59.     {MAX_NUMBER,    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMB
    ER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, 170,        MAX_NUMBER, MAX_NUMBER,
    80,         MAX_NUMBER, MAX_NUMBER, MAX_NUMBER},
60.     {MAX_NUMBER,    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMB
    ER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, 170,        90,
    MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, 180},
61.     {MAX_NUMBER,    MAX_NUMBER, 50,         MAX_NUMBER, MAX_NUMBER, MAX_NUMB
    ER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER, MAX_NUMBER,
    MAX_NUMBER, MAX_NUMBER, 180,        MAX_NUMBER},
62.
63. };
```

```
64.
65. int path_matrix[BUILDING_NUMBER][BUILDING_NUMBER];//最短路径矩阵
66. int short_path_table[BUILDING_NUMBER][BUILDING_NUMBER];//路径长度矩阵
67.
68. void campus_maps();//地图
69. double angle(int front, int behind);//计算两点方向
70. int find_sequence(string name);//找到存储建筑所在数组序号
71.
72. void short_path_floyd(int weight[][BUILDING_NUMBER], int path_matrix[][BUILD
    ING_NUMBER], int short_path_table[][BUILDING_NUMBER]);//计算所有最短路径
73. void path_print(int start, int end, int path_arr[]);//打印路径
74. void direction_print(int end,int path_arr[]);//打印方向
75. void length_sort_all(int sequence,int end_arr[], int path_matrix[][BUILDING_
    NUMBER], int short_path_table[][BUILDING_NUMBER]);//按路径长度排序
76. //----------------------------------------
77. bool visited[BUILDING_NUMBER];
78. void dfs(int start,int stop,int node_number);//dfs 递归搜索路径
79. int stack[MAX_NUMBER], m = 1;///存储路径,m 为栈指针
80. int path_length = 0;//路径长度
81. int path_amount = 0;//路径数量
82. struct all_path
83. {
84.     int path_arr[BUILDING_NUMBER];//储存途径信息
85.     int node_amount;//途径点数量
86.     int path_total_length;//本条路径长度
87. };
88. all_path path_vector[MAX_NUMBER];//结构数组，储存每条路径信息
89. void dfs_sort_print(all_path path_vector[], int node_number, int path_amount
    , int fit_path_number);//将 DFS 搜索后的结果排序并输出
90. //------------------------------------------------
91. int main()
92. {
93.     //campus_maps();//校园地图全貌
94.     short_path_floyd(weight, path_matrix, short_path_table);//求取所有最短路
    径
95.     char mode;
96.     int a=1;//循环标志
97.     while (a==1)
98.     {
99.         cout << "输入 1 进入查询模式，2 进入问路模式，3 进入路径规划模式，q 退出：
    " << endl;
100.
101.         cin >> mode;
102.
```

```
        while (mode != '1' && mode != '2' && mode != '3' && mode != 'q' && mode != 'Q')
        {
            cout << "输入有误，请重新输入！\n 输入 1 进入查询模式，2 进入问路模式，3 进入路径规划模式，q 退出：" << endl;
            cin >> mode;
        }
        //-----------------------------------------------
        while (mode == '1')//查询模式
        {
            string find_name;
            cout << "你想要查：";
            cin >> find_name;
            int sequence = find_sequence(find_name);
            while (sequence == -1)
            {
                cout << "无此信息，请重新输入：";
                cin >> find_name;
                sequence = find_sequence(find_name);
            }
            cout << "所查信息为：" << arr[sequence].name << "  简介：" << arr[sequence].information << "  坐标：
            （ " << arr[sequence].x << " , " << arr[sequence].y << " ） " << endl;
            cout << endl;
            break;
        };
        //-----------------------------------------------
        while (mode == '2')//问路模式,查询指定点到所有点的最短路径
        {
            string start_name;
            cout << "输入起点：";
            cin >> start_name;
            int sequence = find_sequence(start_name);
            while (sequence == -1)
            {
                cout << "无此信息，请重新输入：";
                cin >> start_name;
                sequence = find_sequence(start_name);
            }

            int end_arr[BUILDING_NUMBER - 1];//定义一个终点数组
            int j = 0;
            for (int i = 0; i < BUILDING_NUMBER; i++)
            {
```

```
143.                 //int j = 0;
144.                 if (i != sequence)
145.                 {
146.                     end_arr[j] = i;
147.                     j++;
148.                 }
149.
150.             }
151.             length_sort_all(sequence, end_arr, path_matrix, short_path_tabl
    e);//对给定起点到所有终点按路径长度进行排序
152.             cout << "从这里到其他所有地方的最短路径为：" << endl << endl;
153.             for (int i = 0; i < BUILDING_NUMBER - 1; i++)
154.             {
155.                 int path_arr[BUILDING_NUMBER - 1];
156.                 path_print(sequence, end_arr[i], path_arr);
157.                 cout << endl;
158.             }
159.             cout << endl;
160.             break;
161.         }
162.         //-------------------------------------------------
163.         while (mode == '3')//路径规划模式
164.         {
165.             cout << "输入起点和终点：";
166.
167.             string front, behind;
168.             cin >> front >> behind;
169.             int front_sequence = find_sequence(front);
170.             int behind_sequence = find_sequence(behind);
171.             while (front_sequence == -1 || behind_sequence == -1)
172.             {
173.                 cout << "无此信息，请重新输入：";
174.                 cin >> front >> behind;
175.                 front_sequence = find_sequence(front);
176.                 behind_sequence = find_sequence(behind);
177.             }
178.
179.             cout << "输入途径景点数量：";
180.             int node_number;
181.             cin >> node_number;
182.             stack[0] = front_sequence;//将起点入栈
183.             for (int i = 0; i < BUILDING_NUMBER; i++)//初始化所有未访问
184.                 visited[i] = false;
185.             visited[front_sequence]=true;//初始化起点为访问
```

```
186.             dfs(front_sequence, behind_sequence, node_number);//求解
187.             int fit_path_number=0;//符合要求的路径数目
188.
189.             dfs_sort_print(path_vector, node_number, path_amount, fit_path_
    number);
190.             cout << endl;
191.             path_amount = 0;//路径数重置为0
192.             break;
193.
194.         };
195.         while (mode == 'q' || mode == 'Q')//退出
196.         {
197.             cout << "谢谢，再见！\n";
198.             a = 0;
199.             cout << endl;
200.             break;
201.         };
202.
203.     }
204. }
205. void dfs_sort_print(all_path path_vector[], int node_number, int path_amoun
    t, int fit_path_number )
206. {
207.     //vector<all_path> path[path_amount];
208.     for (int i = 0; i < path_amount; i++)//统计符合所要求结点数量的路径
209.     {
210.         if (path_vector[i].node_amount == node_number)
211.         {
212.             path_vector[fit_path_number] = path_vector[i];
213.             fit_path_number++;
214.         }
215.     }
216.     if (fit_path_number == 0)
217.         cout << "没有符合要求的路径" << endl;
218.     else//排序并输出
219.     {
220.         for (int i = 0; i < fit_path_number; i++)//排序
221.         {
222.             for (int j = i; j < fit_path_number; j++)
223.                 if (path_vector[i].path_total_length > path_vector[j].path_
    total_length)
224.                     swap(path_vector[i], path_vector[j]);
225.
226.         }
```

```
227.         for (int i = 0; i < fit_path_number; i++)
228.         {
229.             for (int j = 0; j < path_vector[i].node_amount; j++)
230.                 cout <<left<<setw(12)<< arr[path_vector[i].path_arr[j]].inf
    ormation << "  ";
231.             cout << "路径长度为  " << path_vector[i].path_total_length;
232.             cout << endl;
233.         }
234.         cout << endl;
235.         cout << "推荐路径: " << endl;
236.         for (int i = 0; i < path_vector[0].node_amount-1; i++)
237.             cout << arr[path_vector[0].path_arr[i]].information << "--
    ->";
238.         cout << arr[path_vector[0].path_arr[path_vector[0].node_amount- 1]
    ].information << endl;
239.         cout << "路径长度为  " << path_vector[0].path_total_length<<endl;
240.         cout << "请根据以下导航走" << endl;
241.         direction_print(path_vector[0].path_arr[path_vector[0].node_amount
    - 1], path_vector[0].path_arr);
242.
243.     }
244. }
245. //-----------------------------------------
246. void campus_maps()//打印地图
247. {
248.     cout << " _____
    _____" << endl;
249.     cout << "|------------------------------哈尔滨工业大学（深圳）平面图----
    ----------------------------------------|" << endl;
250.     cout << "|
                                                |" << endl;
251.     cout << "|     *(E 栋）-----*（D 栋）------*(F 栋）------*---------
    \\                                              |" << endl;
252.     cout << "|     |           |           /            /   （C 栋）    \\----
    --*（B 栋）                                    |" << endl;
253.     cout << "|     |            \\        /__________/                \\
      /     \\        /-----------*（L 栋）       |" << endl;
254.     cout << "|     |               \\   /                                \\
     /        ----*(G 栋）        |             |" << endl;
255.     cout << "|     |_________________\\                              （A
    栋) *_____________/              *（K 栋）       |" << endl;
256.     cout << "|                        \\\ _____________________________\\\
    ___________/                |             |" << endl;
```

```
257.     cout << "|                          \\
                                *（J 栋）       |" << endl;
258.     cout << "|                        ____*___________
                               |             |" << endl;
259.     cout << "|                     /   (一食
    堂)        \\                                        /\\
    |" << endl;
260.     cout << "|                   /                    \\
                              /  \\           |" << endl;
261.     cout << "|                  * _____________________\\\\
                             /     *（H 栋）   |" << endl;
262.     cout << "|                (校
    门)                    \\                                __*(T)___
    /         |" << endl;
263.     cout << "|                                          \\\\    （荔园 789
    栋）                    /                   |" << endl;
264.     cout << "|                                          *_______*_______
    _____________________/                    |" << endl;
265.     cout << "|                                              （三食
    堂）                                        |" << endl;
266.     cout << "|
                                              |" << endl;
267.     cout << "|
                                              |" << endl;
268.     cout << "|
                                              |" << endl;
269.     cout << "|_____________________________________________________________
    ____________________________________|" << endl;
270.     cout << endl;
271.     cout << endl;
272.     cout << " _____________________________________________________"<<endl;
273.     cout << "|                                                    |" << endl;

274.     cout << "|      欢迎使用哈尔滨工业大学（深圳）地图向导        |" << endl;
275.     cout << "|                                                    |" << endl;

276.     cout << "|            请根据提示进行相关操作                  |" << endl;
277.     cout << "|                                                    |" << endl;

278.     cout << "|                已收录的信息如
    下                  |" << endl;
279.     cout << "| 校门 一食堂  三食
    堂 A B C D E F G H J K L T LY   |" << endl;
```

```
280.     cout << "|_____|" << endl;
281.     cout << endl;
282.     cout << endl;
283. }
284. //按路径长度排序
285. void length_sort_all(int sequence, int end_arr[], int path_matrix[][BUILDIN
    G_NUMBER], int short_path_table[][BUILDING_NUMBER])
286. {
287.     for (int i = 0; i < BUILDING_NUMBER - 1; i++)
288.     {
289.         for (int j = i + 1; j < BUILDING_NUMBER - 1; j++)
290.         {
291.             if (short_path_table[sequence][end_arr[i]] > short_path_table[s
    equence][end_arr[j]])
292.                 swap(end_arr[i], end_arr[j]);
293.
294.         }
295.     }
296. }
297. //-------------------------------------------------------------------------
    -
298. //Floyd 算法求最短路径
299. void short_path_floyd(int weight[][BUILDING_NUMBER], int path_matrix[][BUIL
    DING_NUMBER], int short_path_table[][BUILDING_NUMBER])
300. {
301.     int v, w, k;
302.     for (v = 0; v < BUILDING_NUMBER; v++)//初始
    化  path_matrix  和  short_path_table
303.     {
304.         for (w = 0; w < BUILDING_NUMBER; w++)
305.         {
306.             short_path_table[v][w] = weight[v][w];
307.             path_matrix[v][w] = w;
308.         }
309.     }
310.     for (k = 0; k < BUILDING_NUMBER; k++)
311.     {
312.         for (v = 0; v < BUILDING_NUMBER; v++)
313.         {
314.             for (w = 0; w < BUILDING_NUMBER; w++)
315.             {
316.                 if (short_path_table[v][w] > short_path_table[v][k] + short
    _path_table[k][w])
```

```
317.                 {
318.                     short_path_table[v][w] = short_path_table[v][k] + short
    _path_table[k][w];
319.                     path_matrix[v][w] = path_matrix[v][k];
320.                 }
321.             }
322.         }
323.     }
324.     //打印两个矩阵
325.     /*
326.     for (int i = 0; i < BUILDING_NUMBER; i++)
327.     {
328.         for (int j = 0; j < BUILDING_NUMBER; j++)
329.         {
330.             cout << weight[i][j] << "    ";
331.         }
332.         cout << endl;
333.     }
334.     for (int i = 0; i < BUILDING_NUMBER; i++)
335.     {
336.         for (int j = 0; j < BUILDING_NUMBER; j++)
337.         {
338.             cout << short_path_table[i][j]<<"    ";
339.         }
340.         cout << endl;
341.     }
342.     for (int i = 0; i < BUILDING_NUMBER; i++)
343.     {
344.         for (int j = 0; j < BUILDING_NUMBER; j++)
345.         {
346.             cout << path_matrix[i][j]<<"    ";
347.         }
348.         cout << endl;
349.     }
350.     */
351. }
352. //------------------------------------------------------------------------
353. void path_print(int start, int end,int path_arr[]) //打印最短路径
354. {
355.
356.     cout << arr[start].information << " ---> ";//打印起点
357.     int i = 0;
358.     path_arr[i] = start;
359.     int k = path_matrix[start][end];
```

```
    while (k != end)
    {
        i++;
        path_arr[i] = k;
        cout << arr[k].information << " ---> ";
        k = path_matrix[k][end];
    }
    i++;
    path_arr[i] = k;
    cout << arr[end].information << endl;
    cout << "路径总长为：" << short_path_table[start][end] << endl;

}
//------------------------------------------------------------------------
void direction_print(int end,int path_arr[])//打印路径方位
{
    cout << endl;
    int i = 0;
    while (path_arr[i+1] != end)
    {

        int k = path_arr[i];
        int j = path_arr[i + 1];
        double direction = angle(k, j);
        cout <<  arr[j].information <<" 在 " << " " << arr[k].information <<"  "<<direction<<"°方向上，和" << " " << arr[k].information <<"相距 "<<weight[k][j]<<" 米\n";
        i++;
    }
    int k = path_arr[i];
    int j = path_arr[i + 1];
    double direction = angle(k, j);
    cout << arr[j].information << " 在 "  << " " << arr[k].information << "  " << direction << "°方向上，和" <<  " " << arr[k].information << "相距 " << weight[k][j] << " 米\n";

}

//------------------------------------------------------------------------
double angle(int front, int behind)//求方位角
{
    double x1 = arr[front].x;
    double y1 = arr[front].y;
```

```
399.     double x2 = arr[behind].x;
400.     double y2 = arr[behind].y;
401.
402.     double angle_rad=atan( (y1 - y2)/ (x1 - x2));//计算两点间的方向（弧度表
    示）
403.     angle_rad = angle_rad * 180 / acos(-1);//返回角度表示
404.     if (x2 <= x1 && y2 > y1)
405.         return angle_rad + 180;
406.     else
407.         if (x2 <= x1 && y2 < y1)
408.             return angle_rad - 180;
409.         else
410.             if (x2 < x1 && y2 == y1)
411.                 return angle_rad + 180;
412.             else
413.                 if (x2 > x1&& y2 == y1)
414.                     return 0;
415.                 else
416.                     return angle_rad;
417.
418.
419. }
420. //------------------------------------------------------
421. int find_sequence(string name)
422. {
423.     int i = 0;
424.
425.     for (i; i < BUILDING_NUMBER; i++)
426.     {
427.         if (arr[i].name == name)
428.         {
429.
430.             return i;
431.             break;
432.
433.         };
434.
435.     };
436.     return -1;//没找到返回-1
437. }
438. //-------------------------------------------------
439. void dfs(int start,int stop,int node_number)//深度优先搜索路径
440. {
441.     int i, j;
```

```
    for (i = 0; i < BUILDING_NUMBER; i++)
    {
        if (weight[start][i] != MAX_NUMBER&& m<node_number&&visited[i]==false)
        {
            if (i == stop)//如果深搜到了终点，就输出刚才经过的路径
            {
                path_length += weight[start][i];

                for (j = 0; j < m; j++)
                {
                    //printf("%-5d", stack[j]);
                    path_vector[path_amount].path_arr[j] = stack[j];//存路径

                }
                //printf("%-5d\n", stop);
                path_vector[path_amount].path_arr[m] = stop;//存终点
                path_vector[path_amount].node_amount = m + 1;//存途径点的数量
                //cout << "此条路径长度为" << path_length << endl;
                path_vector[path_amount].path_total_length = path_length;//存长度
                path_amount++;//每到一次终点，路径数目+1
                path_length -= weight[start][i];
            }
            else///如果该点不是终点
            {
                visited[i] = true;
                path_length += weight[start][i];
                int length = weight[i][start];//暂存路长信息
                weight[start][i] = MAX_NUMBER;
                weight[i][start] = MAX_NUMBER;
                stack[m] = i;//将该点存起来
                m++;
                dfs(i,stop,node_number);//接着深搜
                visited[i] = false;
                weight[start][i] = length;
                weight[i][start] = length;
                path_length -= weight[start][i];
                m--;
            }
        }
    }
```

```
482. }
```