

**实验报告**

目录

[**实验报告** 1](#_Toc60167823)

[一、 实验内容 3](#_Toc60167824)

[1.1 问题描述 3](#_Toc60167825)

[1.2基本要求 3](#_Toc60167826)

[1.3 附加内容 3](#_Toc60167827)

[二、 实验环境 3](#_Toc60167828)

[三、 实验过程 4](#_Toc60167829)

[3.1主界面设计 4](#_Toc60167830)

[3. 2 存储结构设计 4](#_Toc60167831)

[3. 3 系统功能设计 4](#_Toc60167832)

[3.3 模块设计 5](#_Toc60167833)

[四、 实验结果 7](#_Toc60167834)

[4.1 系统首页： 7](#_Toc60167835)

[4.2 景点列表： 7](#_Toc60167836)

[4.3 查询路径： 8](#_Toc60167837)

[4.4 更改路径： 8](#_Toc60167838)

[4.5 查看地图： 9](#_Toc60167839)

[五、 实验体会 9](#_Toc60167840)

[六、 源代码 10](#_Toc60167841)

# 实验内容

## 1.1 问题描述

设计一个校园导游程序， 为来访的客人提供信息查询服务。

## 1.2基本要求

（1） 设计学校的校园平面图。 选取若干个有代表性的景点抽象成一个无向带权图（无向网），以图中顶点表示校内各景点，边上的权值表示两景点之间的距离。

（2） 为来访客人提供图中任意景点相关信息的查询。

（3） 为来访客人提供图中任意景点之间的问路查询。最短路径

## 1.3 附加内容

（1） 为来访客人提供图中任意景点之间的问路查询。所有路径

（2） 修改删除

# 实验环境

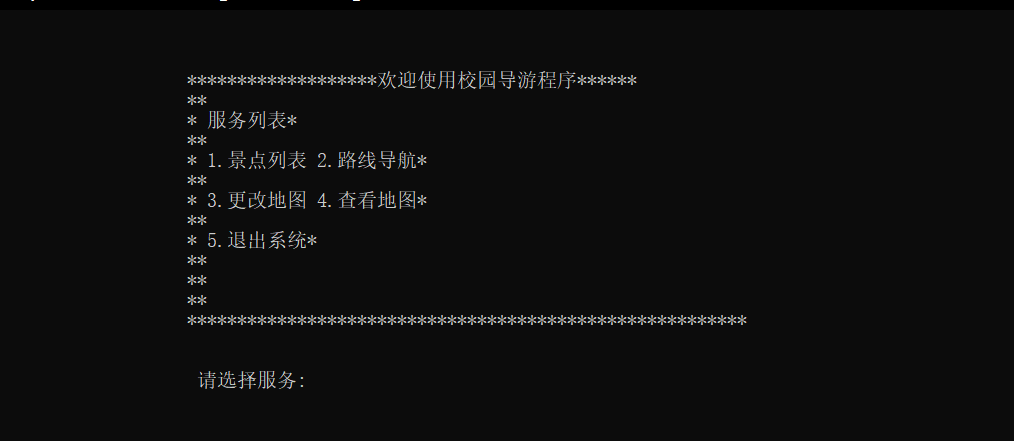
* PC
* Win10 pro
* Visual Studio Code
* C/C++

# 实验过程

## 3.1主界面设计

为了实现校园导游系统各功能的管理。 首先设计一个含有多个菜单选项的主控

菜单子程序以链接系统的各项子功能， 方便用户使用本系统主控菜单运行界面如图：



## 3. 2 存储结构设计

本系统采用图结构类型（mgraph） 存储抽象校园图的信息。 其中， 各景点间的邻接关系用图的邻接矩阵类型（adjmatrix） 存储； 景点（顶点） 信息用结构数组（vexs） 存储， 其中每个数组元素是一个结构变量， 包含景点编号、 景点名称及景点介绍三个分量； 图的顶点个数及边的个数由分量vexnum、arcnum表示， 它们是整型数据。此外， 本系统还设置了三个全局变量：visited[]数组用于存储顶点是否被访问标志； d[]数组用于存放边上权值或存储查找路径顶点的编号； c是一个图结构的全局变量。

## 3. 3 系统功能设计

本系统除了要完成图的初始化功能外还设置了4个子功能菜单。 图的初始化右

函数Chushihua() 实现。 依据读入的图的顶点个数和边的个数， 分别初始化图结构中图的顶点向量数组和图的邻接矩阵。 4个子功能的设计描述如下。

（1） 景点列表

景点列表由函数Viewlist() 实现。 当用户选择该功能， 系统即能输入学校全部

景点的信息： 包括景点编号、 景点名称及景点简介。

（2） 路线导航

路线导航由函数 Guide() 实现， 该功能采用弗洛伊德（Floyd） 和迪杰斯特拉（Dijkstra） 算法实现。 当用户选择该功能， 系统能根据用户输入的起始景点编号和目标景点编号， 求出从该景点到其他景点的最短路径线路及距离， 和其他包含 7个景点以内的可行路径。

（3） 更改地图

更改图的信息功能由主调函数 Changegraph() 及若干子函数完成， 可以实现图的若干基本操作。 如增加新的景点、 删除边、 修改景点简介等。

（4）查看地图

可以查看大致地图

（5） 退出

即退出校园导游系统， 由函数 Exit() 实现。

## 3.3 模块设计

本程序包含 3 个模块： 主程序模块、 工作区模块和无向网操作模块。

各子程序的函数名及功能说明如下：

picture Initialize();            //图的初始化

    void Viewlist();                 //景点列表

    void Guide();                    //景点导航

    void ShortestPath(int a, int b); // 用 Dijktra 算法， 求最短路径

    void AllPath(int a, int b);      //查询所有可行路径

    void Path(int a, int b, int k);

    //被AllPath调用打印两景点间的景点个数小于6的所有路径

    void Changegraph();       //图操作的主调函数

    void DeletePlace();       //删除景点

    void DeletePath();        //删除路径

    void AddPlace();          //增加景点

    void AddPath();           //增加路径

void UpdateInformation(); //更新简介

 void Waitting(long time); // 被Loading(char) 调用， 确定等待时间

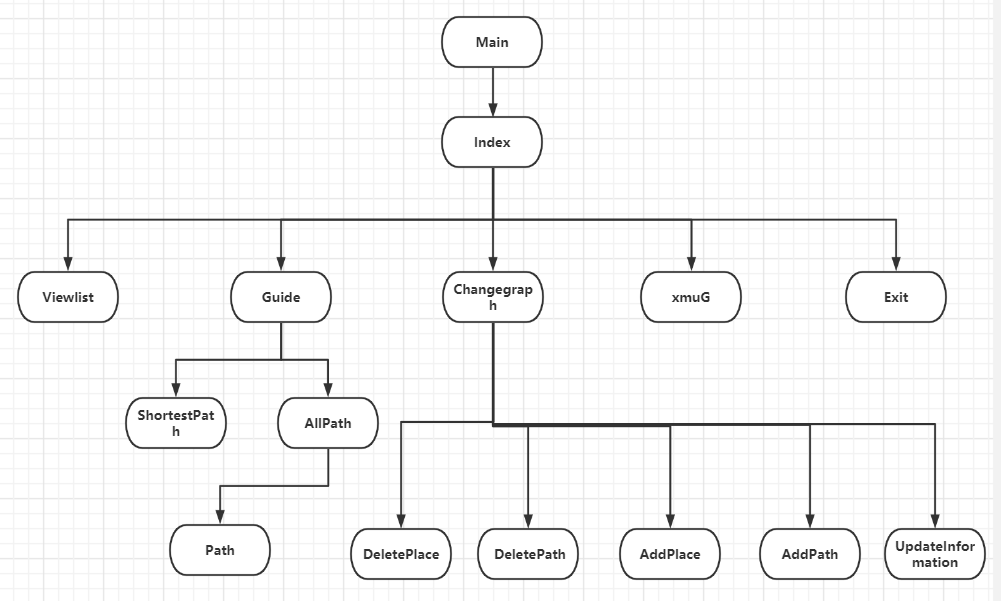
    void Loading(string s);   //页面间的加载界面

    void Index();             //主界面

    void xmuG();              //地图

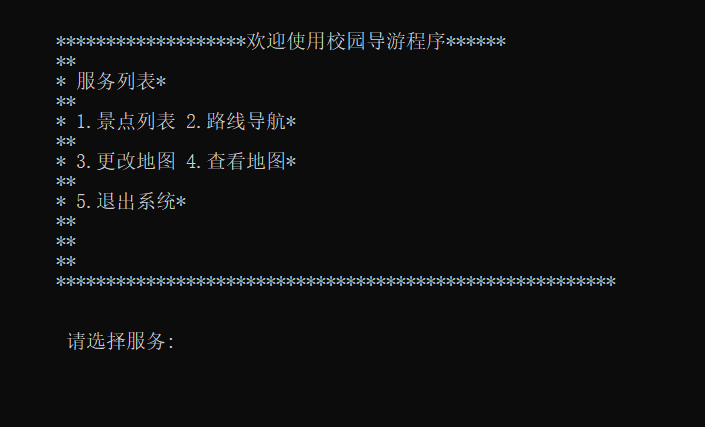
    void Exit();              //退出

校园导游系统 17 个子程序之间的主要调用关系如图所示。

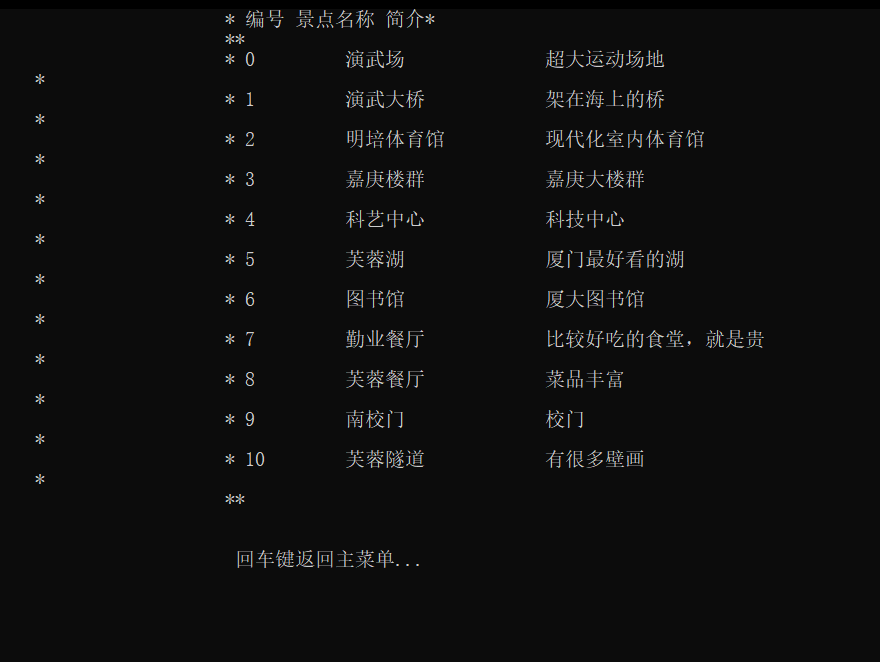


# 实验结果

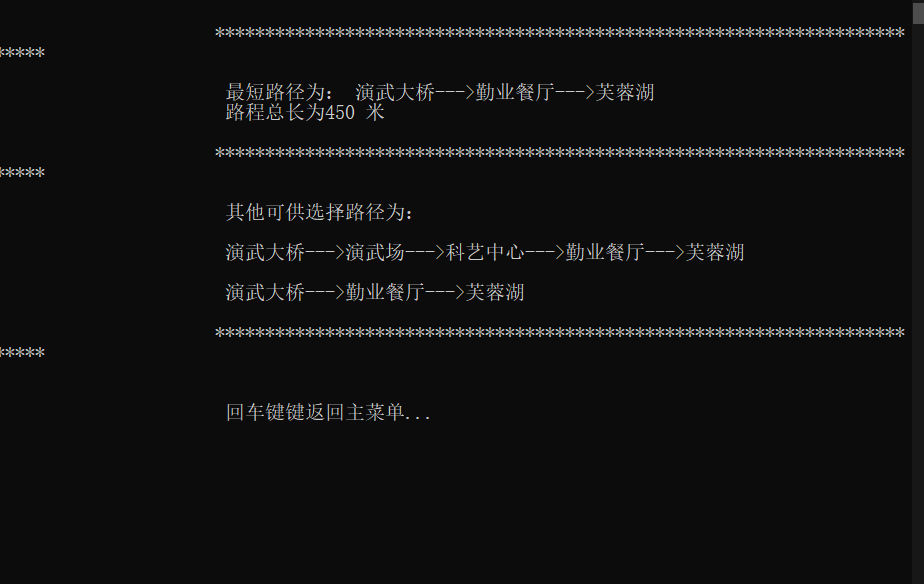
## 4.1 系统首页：



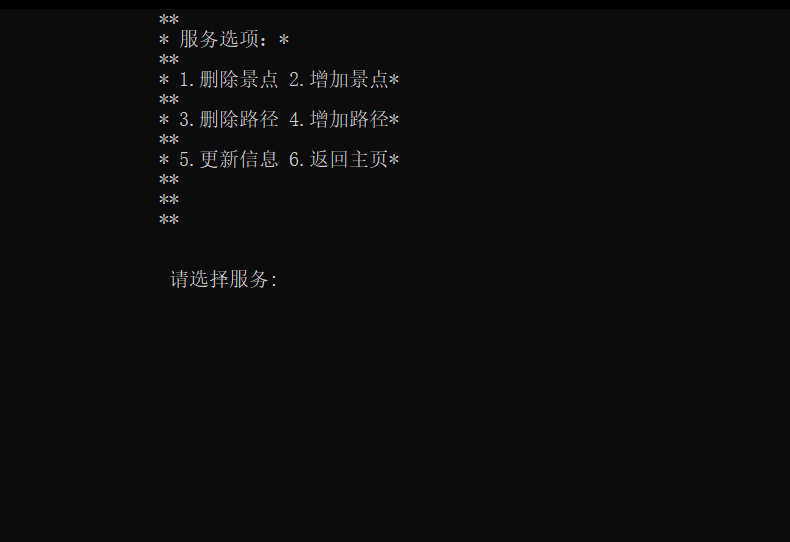
## 4.2 景点列表：



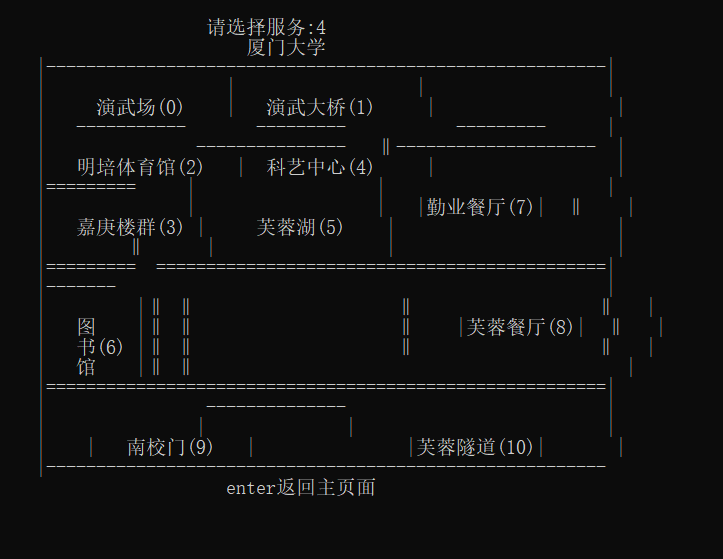
## 4.3 查询路径：



## 4.4 更改路径：



## 4.5 查看地图：



# 实验体会

本次实验圆满完成。在完成课题的过程中不可避免地遇见了很多问题。

尤其是在对图进行操作和迪及斯特拉算法运用的时候最为棘手。在本学期的数据结构课程中，这一单元也是掌握最不牢固，最为薄弱的一个环节，因此运用起来也最为生疏。通过此次课程设计，我对这些算法掌握更为牢固，理解更加深刻。为来走向工作岗位铺下了又一快基石。

# 源代码

#include <cstring>

#include <iostream>

using namespace std;

#define Null 1000

#define MaxVertexNum 20

#define Max 40

char \*my\_strcpy(char \*dest, const char \*src) { //返回值保存

    char \*p = dest;                            //检查入参，使程序健壮

    assert(dest != NULL || src != NULL);       //注意越界

    while ((\*dest++ = \*src++) != '\0')

        ;

    return p; //返回字符指针，使函数可以用于链式表达式，增强可用性

}

typedef struct side //边的信息

{

    int adj;                                   //边的权值

} side, adjmatrix[MaxVertexNum][MaxVertexNum]; //邻接矩阵

typedef struct vertex //顶点信息

{

    int number;             //景点编号

    char name[32];          //景点名称

    char introduction[256]; //景点介绍

} vertex;

typedef struct picture //图

{

    vertex vexs[MaxVertexNum]; //顶点向量（数组）

    adjmatrix arcs;            //邻接矩阵

    int vexnum, arcnum;        //顶点个数 边的个输

} picture;

picture xmu;

int visited[15], d[15];

class Graph {

  public:

    picture Initialize();            //图的初始化

    void Viewlist();                 //景点列表

    void Guide();                    //景点导航

    void ShortestPath(int a, int b); // 用 Dijktra 算法， 求最短路径

    void AllPath(int a, int b);      //查询所有可行路径

    void Path(int a, int b, int k);

    //被AllPath调用打印两景点间的景点个数小于6的所有路径

    void Changegraph();       //图操作的主调函数

    void DeletePlace();       //删除景点

    void DeletePath();        //删除路径

    void AddPlace();          //增加景点

    void AddPath();           //增加路径

    void UpdateInformation(); //更新简介

};

class Other {

  public:

    void Waitting(long time); // 被Loading(char) 调用， 确定等待时间

    void Loading(string s);   //页面间的加载界面

    void Index();             //主界面

    void xmuG();              //地图

    void Exit();              //退出

};

int main() {

    Graph G;

    Other O;

    xmu = G.Initialize();

    O.Index();

    return 0;

    system("pause");

}

picture Graph::Initialize() { //图的初始化

    int i = 0, j = 0;

    picture xmu;

    xmu.vexnum = 11; //顶点个数

    xmu.arcnum = 23; //边的个数

    for (i = 0; i < xmu.vexnum; i++)

        xmu.vexs[i].number = i;

    my\_strcpy(xmu.vexs[0].name, "演武场");

    my\_strcpy(xmu.vexs[0].introduction, "超大运动场地");

    my\_strcpy(xmu.vexs[1].name, "演武大桥");

    my\_strcpy(xmu.vexs[1].introduction, "架在海上的桥");

    my\_strcpy(xmu.vexs[2].name, "明培体育馆");

    my\_strcpy(xmu.vexs[2].introduction, "现代化室内体育馆");

    my\_strcpy(xmu.vexs[3].name, "嘉庚楼群");

    my\_strcpy(xmu.vexs[3].introduction, "嘉庚大楼群");

    my\_strcpy(xmu.vexs[4].name, "科艺中心");

    my\_strcpy(xmu.vexs[4].introduction, "科技中心");

    my\_strcpy(xmu.vexs[5].name, "芙蓉湖");

    my\_strcpy(xmu.vexs[5].introduction, "厦门最好看的湖");

    my\_strcpy(xmu.vexs[7].name, "勤业餐厅");

    my\_strcpy(xmu.vexs[7].introduction, "比较好吃的食堂，就是贵");

    my\_strcpy(xmu.vexs[6].name, "图书馆");

    my\_strcpy(xmu.vexs[6].introduction, "厦大图书馆");

    my\_strcpy(xmu.vexs[8].name, "芙蓉餐厅");

    my\_strcpy(xmu.vexs[8].introduction, "菜品丰富");

    my\_strcpy(xmu.vexs[9].name, "南校门");

    my\_strcpy(xmu.vexs[9].introduction, "校门");

    my\_strcpy(xmu.vexs[10].name, "芙蓉隧道");

    my\_strcpy(xmu.vexs[10].introduction, "有很多壁画");

    for (i = 0; i < xmu.vexnum; i++)

        for (j = 0; j < xmu.vexnum; j++)

            xmu.arcs[i][j].adj = Null;

    //初始化各个边为 1000，表示两个景点之间不可达

    xmu.arcs[0][1].adj = xmu.arcs[1][0].adj = 100;

    xmu.arcs[0][2].adj = xmu.arcs[2][0].adj = 100;

    xmu.arcs[1][7].adj = xmu.arcs[7][1].adj = 200;

    xmu.arcs[3][5].adj = xmu.arcs[5][3].adj = 130;

    xmu.arcs[3][6].adj = xmu.arcs[6][3].adj = 150;

    xmu.arcs[4][0].adj = xmu.arcs[0][4].adj = 200;

    xmu.arcs[4][7].adj = xmu.arcs[7][4].adj = 220;

    xmu.arcs[5][7].adj = xmu.arcs[7][5].adj = 250;

    xmu.arcs[6][12].adj = xmu.arcs[12][6].adj = 100;

    xmu.arcs[6][3].adj = xmu.arcs[3][6].adj = 90;

    xmu.arcs[7][8].adj = xmu.arcs[8][7].adj = 200;

    xmu.arcs[8][9].adj = xmu.arcs[9][8].adj = 500;

    xmu.arcs[8][10].adj = xmu.arcs[10][8].adj = 110;

    xmu.arcs[9][10].adj = xmu.arcs[10][9].adj = 300;

    xmu.arcs[9][6].adj = xmu.arcs[6][9].adj = 100;

    return xmu;

}

void Graph::Viewlist() { //景点列表

    Other O;

    O.Loading("景点列表");

    printf("\t\t\t\* 编号 景点名称 简介\* \n");

    printf("\t\t\t\*\*\n");

    for (int i = 0; i < xmu.vexnum; i++)

        printf("\t\t\t\* %-10d%-20s%-42s\*\n", xmu.vexs[i].number,

               xmu.vexs[i].name, xmu.vexs[i].introduction);

    printf("\t\t\t\*\*\n");

    printf("\n\n\t\t\t 回车键返回主菜单...");

    getchar();

    getchar();

    O.Loading("主页");

    O.Index();

}

void Graph::Guide() { //景点导航

    Other O;

    Graph G;

    int a, b;

    O.Loading("导航界面");

    printf("\n\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\n\t\t\t 请输入您当前位置的代号:");

    cin >> a;

    while (!(a >= 0 && a < xmu.vexnum)) {

        printf("\n\t\t\t 输入信息错误， 请重新输入:");

        cin >> a;

    }

    printf("\n\t\t\t 请输入您目标位置的代号:");

    cin >> b;

    while (!(b >= 0 && b < xmu.vexnum)) {

        printf("\n\t\t\t 输入信息错误， 请重新输入:");

        cin >> b;

    }

    printf("\n\n\n\t\t\t\t\t 正在查询路线\t....");

    for (int i = 0; i < 5; i++) {

        O.Waitting(60000000);

        printf(".");

    }

    system("cls");

    ShortestPath(a, b);

    G.AllPath(a, b);

    printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

           "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\n\n\t\t\t 回车键键返回主菜单...");

    getchar();

    getchar();

    O.Loading("主页");

    O.Index();

}

void Graph::ShortestPath(int a, int b) { // 用 Dijktra 算法， 求最短路径

    int i, v, u, w, d[15][15], p[15][15][15]; // v0 为起始点

    for (v = 0; v < xmu.vexnum; v++) {

        for (w = 0; w < xmu.vexnum; w++) {

            d[v][w] = xmu.arcs[v][w].adj;

            for (u = 0; u < xmu.vexnum; u++)

                p[v][w][u] = 0;

            if (d[v][w] < Null) {

                p[v][w][v] = 1;

                p[v][w][w] = 1;

            }

        }

    }

    for (u = 0; u < xmu.vexnum; u++) {

        for (v = 0; v < xmu.vexnum; v++)

            for (w = 0; w < xmu.vexnum; w++)

                if (d[v][u] + d[u][v] < d[v][w]) {

                    d[v][w] = d[v][u] + d[u][w];

                    for (i = 0; i < xmu.vexnum; i++)

                        p[v][w][i] = p[v][u][i] || p[u][w][i];

                }

    }

    printf("\n\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

           "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\n\t\t\t 最短路径为： ");

    printf("%s", xmu.vexs[a].name);

    for (u = 0; u < xmu.vexnum; u++)

        if (p[a][b][u] && a != u && b != u) {

            printf("--->%s", xmu.vexs[u].name);

        }

    printf("--->%s", xmu.vexs[b].name);

    printf("\n\t\t\t 路程总长为%d 米\n\n", d[a][b]);

    printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

           "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\n\t\t\t 其他可供选择路径为： \n\n");

}

void Graph::AllPath(int a, int b) { //查询所有可行路径

    Graph G;

    d[0] = a; //存储路径起点 m (int d[]数组是全局变量)

    for (int k = 0; k < xmu.vexnum; k++) //全部顶点访问标志初值设为 0

        visited[k] = 0;

    visited[a] = 1; //第 m 个顶点访问标志设置为 1

    G.Path(a, b, 0); //调用(3)。 k=0， 对应起点 d[0]==a。 k 为 d[]数组下标

}

void Graph::Path(int a, int b, int k) {

    Graph G;

    int s;

    int t = k + 1; // t 记载路径上下一个中间顶点在 d[]数组中的下标

    if (d[k] == b && k < 6) {

        // d[k]存储路径顶点。 若 d[k]是终点 n且景点个数<= 8， 则输出该路径

        printf("\t\t\t "); //递归出口， 找到一条路径

        for (s = 0; s < k; s++)

            printf("%s--->", xmu.vexs[d[s]].name);

        //输出该路径。 s=0 时为起点 m

        printf("%s", xmu.vexs[d[s]].name);

        //输出最后一个景点名(即顶点 n 的名字，此时 s==k)

        printf("\n\n");

    } else {

        int s = 0;

        while (s < xmu.vexnum) //从第 m 个顶点， 试探至所有顶点是否有路径

        {

            if ((xmu.arcs[d[k]][s].adj < Null) && (visited[s] == 0)) {

                //初态： 顶点 m 到顶点 s 有边， 且未被访问

                visited[s] = 1;

                d[k + 1] = s; //存储顶点编号 s 至 d[k+1]中

                G.Path(a, b, t);

                //求从下标为 t=k+1 的第d[t]个顶点开始的路径(递归调用)，

                //同时打印出一条 m 至 n的路径

                visited[s] = 0;

                //将找到的路径上顶点的访问标志重新设置为 0，用于试探新的路径

            }

            s++; //试探从下一个顶点 s 开始是否有到终点的路径

        }

    }

}

void Graph::Changegraph() { //图操作的主调函数

    Other o;

    Graph g;

    o.Loading("更改地图界面");

    int yourchoice;

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\* 服务选项：\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\* 1.删除景点 2.增加景点\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\* 3.删除路径 4.增加路径\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\* 5.更新信息 6.返回主页\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\*\*\n");

    printf("\n\n\t\t\t 请选择服务:");

    cin >> yourchoice;

    printf("\n\n");

    while (!(yourchoice == 1 || yourchoice == 2 || yourchoice == 3 ||

             yourchoice == 4 || yourchoice == 5 || yourchoice == 6)) {

        printf("\n\t\t\t 输入信息错误， 请重新输入:");

        cin >> yourchoice;

    }

    while (1) {

        switch (yourchoice) {

        case 1:

            g.AddPlace();

            break;

        case 3:

            g.DeletePath();

            break;

        case 2:

            g.AddPlace();

            break;

        case 4:

            g.AddPath();

            break;

        case 5:

            g.UpdateInformation();

            break;

        case 6:

            o.Loading("主页");

            o.Index();

        }

    }

}

void Graph::DeletePlace() { //删除景点

    Other O;

    int i = 0, j;

    int v;

    system("cls");

    if (xmu.vexnum <= 0) {

        printf("\n\t\t\t\t 图中已无顶点\n");

    }

    printf("\n\t\t\t\t 请输入你要删除的景点编号： ");

    cin >> v;

    while (v < 0 || v > xmu.vexnum) {

        printf("\n\t\t\t\t 输入错误！ 请重新输入:");

        cin >> v;

    }

    printf("\n\t\t\t\t 顶点 %d 已删除\n\n\n", v);

    for (i = v; i < xmu.vexnum - 1; i++) {

        //对顶点信息所在顺序表进行删除v 点的操作

        my\_strcpy(xmu.vexs[i].name, xmu.vexs[i + 1].name);

        my\_strcpy(xmu.vexs[i].introduction, xmu.vexs[i + 1].introduction);

    }

    //对原邻接矩阵， 删除该顶点到其余顶点的邻接关系。 分别删除相应的行和列

    for (i = v; i < xmu.vexnum - 1; i++) //行

        for (j = 0; j < xmu.vexnum; j++) //列

            xmu.arcs[i][j] = xmu.arcs[i + 1][j];

    //二维数组， 从第 m+1行开始依次往前移一行。即删除第 v 行

    xmu.vexnum--;

    printf("\n\n\t\t\t 回车键返回主菜单...");

    getchar();

    getchar();

    O.Loading("主页");

    O.Index();

}

void Graph::DeletePath() { //删除路径

    Other o;

    int v0, v1;

    system("cls");

    printf("\n\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

           "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    if (xmu.arcnum <= 0) {

        printf("\n\t\t\t\t 图中已无边， 无法删除。 ");

    }

    printf("\n\t\t\t\t 请输入你要删除的边的起点和终点编号： ");

    cin >> v0 >> v1;

    while (v0 < 0 || v0 > xmu.vexnum || v1 < 0 || v1 > xmu.vexnum) {

        printf("\n\t\t\t\t 输入错误！ 请重新输入:");

        cin >> v0 >> v1;

    }

    xmu.arcs[v0][v1].adj = Null; //修改邻接矩阵对应的权值

    xmu.arcs[v1][v0].adj = Null;

    xmu.arcnum--;

    printf("\n\t\t\t\t 边(%d,%d)已删除\n\n\n", v0, v1);

    printf("\n\n\t\t\t 回车键返回主菜单...");

    getchar();

    getchar();

    o.Loading("主页");

    o.Index();

}

void Graph::AddPlace() { //增加景点

    Other o;

    int i;

    system("cls");

    printf("\n\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

           "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\n\t\t\t\t 请输入你要增加结点的信息");

    printf("\n\t\t\t\t 编号： ");

    cin >> xmu.vexs[xmu.vexnum].number;

    printf("\n\t\t\t\t 名称： ");

    cin >> xmu.vexs[xmu.vexnum].name;

    printf("\n\t\t\t\t 简介： ");

    cin >> xmu.vexs[xmu.vexnum].introduction;

    xmu.vexnum++;

    for (i = 0; i < xmu.vexnum; i++) //对原邻接矩阵新增加的一行及一列进行初始化

    {

        xmu.arcs[xmu.vexnum - 1][i].adj = Null; //最后一行(新增的一行)

        xmu.arcs[i][xmu.vexnum - 1].adj = Null; //最后一列(新增的一列)

    }

    printf("\n\t\t\t\t 景点%s 增加成功\n\n\n", xmu.vexs[xmu.vexnum - 1].name);

    printf("\n\n\t\t\t 回车键返回主菜单...");

    getchar();

    getchar();

    o.Loading("主页");

    o.Index();

}

void Graph::AddPath() { //增加路径

    int m, n, distance;

    system("cls");

    printf("\n\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

           "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\n\t\t\t\t 请输入边的起点和终点编号,权值： ");

    cin >> m >> n >> distance;

    while (m < 0 || m > xmu.vexnum || n < 0 || n > xmu.vexnum) {

        printf("\n\t\t\t\t 输入错误， 请重新输入： ");

        cin >> m >> n;

    }

    xmu.arcs[m][n].adj = distance;

    xmu.arcs[n][m].adj = xmu.arcs[m][n].adj; //对称赋值

    printf("\n\t\t\t\t 路径（%d,%d） 增加成功\n\n\n", m, n);

    printf("\n\n\t\t\t 回车键返回主菜单...");

    getchar();

    getchar();

    Other o;

    o.Loading("主页");

    o.Index();

}

void Graph::UpdateInformation() { //更新简介

    system("cls");

    int v;

    printf("\n\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

           "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\n\t\t\t\t 请输入你目标景点的编号： ");

    cin >> v;

    while (v < 0 || v > xmu.vexnum) {

        printf("\n\t\t\t\t 输入错误！ 请重新输入:");

        cin >> v;

    }

    printf("\n\t\t\t\t 请输入新的简介： ");

    cin >> xmu.vexs[v].introduction;

    printf("\n\t\t\t\t 景点%s 简介修改成功\n\n\n", xmu.vexs[v].name);

    printf("\n\n\t\t\t 回车键返回主菜单...");

    Other o;

    o.Loading("主页");

    o.Index();

}

void Other::Waitting(long time) { // 被Loading(char) 调用， 确定等待时间

    int i, j;

    for (i = 0; i < 2; i++)

        for (j = 0; j < time; j++)

            ;

}

void Other::Loading(string s) { //页面间的加载界面

    system("cls");

    cout << "\n\n\n\t\t\t\t\t" << s << " 正在加载\t....";

    for (int i = 0; i < 5; i++) {

        Waitting(30000000);

        printf(".");

    }

    system("cls");

}

void Other::Index() { //主界面

    int choice;

    printf("\n\n\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*欢迎使用校园导游程序\*\*\*\*\*\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\* 服务列表\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\* 1.景点列表 2.路线导航\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\* 3.更改地图 4.查看地图\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\* 5.退出系统\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\*\*\n");

    printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\n\n\t\t\t 请选择服务:");

    cin >> choice;

    Graph g;

    Other o;

    while (true) {

        switch (choice) {

        case 1:

            g.Viewlist();

            break;

        case 2:

            g.Guide();

            break;

        case 3:

            g.Changegraph();

            break;

        case 4:

            o.xmuG();

            break;

        case 5:

            o.Exit();

            break;

        default:

            cout << "\n\t\t\t 输入信息错误， 请重新输入:";

            cin >> choice;

            break;

        }

    }

}

void Other::Exit() { //退出

    system("cls");

    printf("\n\n\n\t\t\t\t\t 正在退出\t....");

    for (int i = 0; i < 5; i++) {

        Waitting(30000000);

        printf(".");

    }

    system("cls");

    printf("\n\n\n\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\t\t\t\t\t\* \*\n");

    printf("\t\t\t\t\t\* 感谢您的使用！ 再见！ \*\n");

    printf("\t\t\t\t\t\* \*\n");

    printf("\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n\n");

    exit(0);

}

void Other::xmuG() {

    system("cls");

    Other o;

    printf("\t                     厦门大学                       \n");

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

    printf("\t|                  |                  |                  |\n");

    printf("\t|     演武场(0)    |   演武大桥(1)     |                  |\n");

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

    printf("\t|               ---------------   ‖--------------------  |\n");

    printf("\t|   明培体育馆(2)   |  科艺中心(4)     |                  |\n");

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

    printf("\t|              |                  |   |勤业餐厅(7)|  ‖    |\n");

    printf("\t|   嘉庚楼群(3) |     芙蓉湖(5)    |                      |\n");

    printf("\t|        ‖      |                 |                      |\n");

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

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

    printf("\t|         |‖ ‖                    ‖                  ‖   |\n");

    printf("\t|   图    |‖ ‖                    ‖    |芙蓉餐厅(8)|  ‖   |\n");

    printf("\t|   书(6) |‖ ‖                    ‖                  ‖   |\n");

    printf("\t|   馆    |‖ ‖                                           |\n");

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

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

    printf("\t|               |              |                         |\n");

    printf("\t|    |   南校门(9)   |               |芙蓉隧道(10)|       |\n");

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

    printf("\t                   enter返回主页面                    \n");

    getchar();

    getchar();

    o.Index();

}