姓名：岳宇轩

学号：19020011038

科目：数据结构与算法

指导老师：纪筱鹏

/\*因为编译器的原因，中文和中文注释编译报错，试过几个方案没有解决，所以景点信息都用的英文，没有添加注释，关键算法的解释写在实验报告中了\*/

1. **实验题目**

校园导游咨询

1. **实验目的**

掌握图的存储方法和最短路经算法

1. **实验要求**
2. 设计所在学校的校园平面图，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、为来访客人提供图中任意景点的纹路查询，即查询任意两个景点之间的一条最短的简单路径。

1. **实验内容和实验步骤**

**4.1实验内容**

设计一个校园导游程序，为来访客人提供各种信息查询服务。测试数据根据实际情况指定。一般情况下，校园的道路是双向通行的，可设校园平面图是一个无向图。顶点和边均含有相关信息。

**4.2实验步骤**

**4.2.1数据结构**

#define MAX\_VERTEX\_NUM 10  
using namespace std;  
  
typedef enum {  
 *DG*, *DN*, *UDG*, *UDN*} GraphKind;  
  
typedef struct {  
 string name;  
 int code;  
 string description;  
} VertexType;  
  
typedef struct {  
 VertexType vexs[MAX\_VERTEX\_NUM];  
 int arcs[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];  
 int vexnum, arcnum;  
 GraphKind kind;  
} MGraph;

**4.2.2函数声明：**

**void init();**

初始化校园信息对应的邻接矩阵

**void PrintHint();**

打印控制台输入提示

**void CheckView(MGraph &G);**

景点信息查询,参数为校园信息的邻接矩阵，函数功能：输入景点编号，输出相应景点的有关信息。

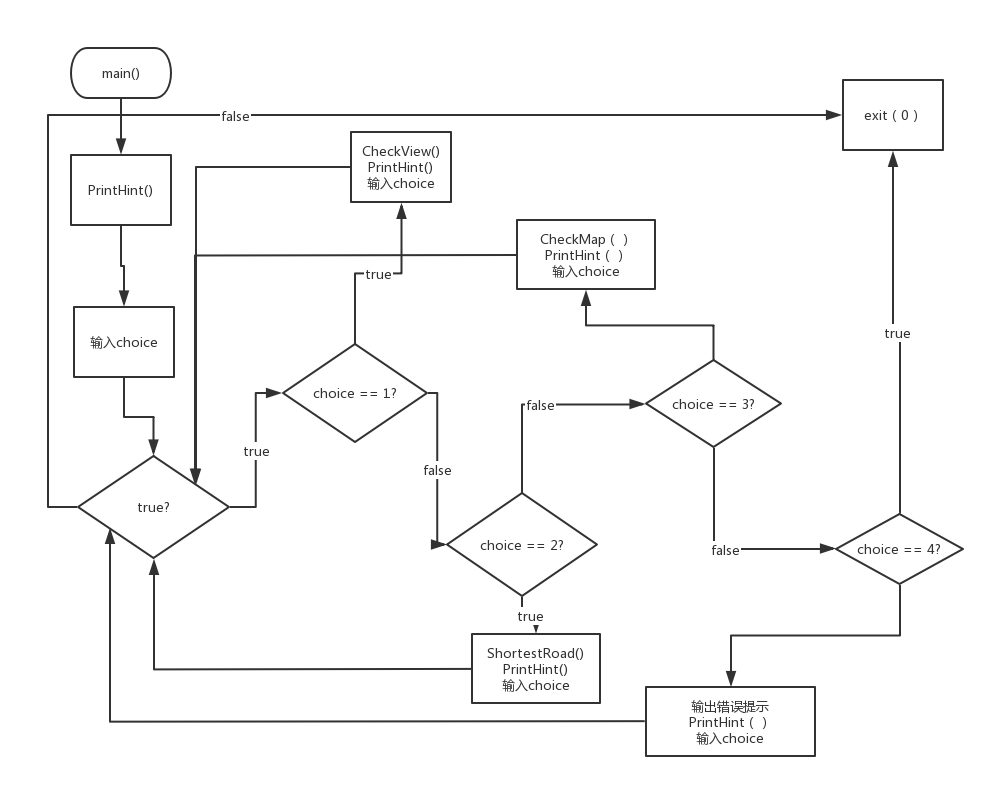
**void ShortestRoad(MGraph &G);**

景点之间最短路径的查询，参数为校园信息的邻接矩阵，函数功能：输入起点和终点的景点编号，输出从起点到终点的最短路径和最短路径长度。

**void CheckMap(MGraph &G);**

控制台输出校园地图，包括景点名称，编号，路径和路径长度。

**4.2.3主函数调用关系**

****

**4.3主要功能函数分析（其它函数的实现都比较简单，主要写一下FLOYD算法求最短路径的函数）（时间复杂度O（n^3））：**

**第一步：初始化D矩阵和P矩阵**

CheckMap(G);  
cout << "Please enter the code of the start point:" << endl;  
int start;  
cin >> start;  
while (start < 1 || start > 10) {  
 cout << "Wrong input! Please retry!" << endl;  
 cin >> start;  
}  
cout << "Please enter the code of the end point:" << endl;  
int end;  
cin >> end;  
while (end < 1 || end > 10) {  
 cout << "Wrong input! Please retry!" << endl;  
 cin >> end;  
}  
start -= 1;  
end -= 1;  
// Flory  
int D[10][10];  
bool P[10][10][10];  
for (int v = 0; v < G.vexnum; ++v) {  
 for (int w = 0; w < G.vexnum; ++w) {  
 if (G.arcs[v][w] == 0) {  
 D[v][w] = 999;  
 } else {  
 D[v][w] = G.arcs[v][w];  
 }  
 if (v == w)  
 D[v][w] = 0;  
 for (int u = 0; u < G.vexnum; ++u) {  
 P[v][w][u] = false;  
 }  
 if (D[v][w] < 999) {  
 P[v][w][v] = true;  
 P[v][w][w] = true;  
 }  
 }  
}

**方法：**

**D矩阵**（D[i][j]表示从i到j的最短路径的长度）：

一：将邻接矩阵赋值给D矩阵

二：D矩阵中元素为0的位置全部赋值999

p.s.这里有个疑惑，不知道为什么用INFINITY不行，写像999这样的大数可以

三：D矩阵对角线置0

**P矩阵**（P[i][j][k]表示k是否是从i到j的最短路径上的点）：

一：全部置false

二：如果D[i][j]不为999，则说明i到j有路径，i和j都是从i到j的最短路径上的点，故P[i][j][i]和P[i][j][j]都应为true

**第二步：不断更新D矩阵和P矩阵**

for (int u = 0; u < G.vexnum; ++u) {  
 for (int v = 0; v < G.vexnum; ++v) {  
 for (int w = 0; w < G.vexnum; ++w) {  
 if (D[v][u] + D[u][w] < D[v][w]) {  
 D[v][w] = D[v][u] + D[u][w];  
 for (int i = 0; i < G.vexnum; ++i) {  
 P[v][w][i] = P[v][u][i] || P[u][w][i];  
 }  
 }  
 }  
 }  
}

**方法：**

先以u为中间结点，分析从v到w：

如果从v到u的路径长度+从u到w的路径长度 < 从v到w的路径长度，则有：

**D矩阵的更新**：

更新从v到w的路径长度为从v到u的路径长度+从u到w的路径长度，因此更新D[v][w]为D[v][u] +D[u][w].

**P矩阵的更新：**

最短路径的中间结点为u，则可知v→u→w一定是v到w最短路径上的路，那么对于任意一个结点i，要更新、判断它是不是新的最短路径上的结点，只要看它在不在vu段或者uw段即可，因此更新P[v][w][i]为P[v][u][i] || P[u][w][i].

**第三步：根据D矩阵输出最短路径长度，根据P矩阵输出最短路径**

cout << "The shortest path is :" << endl;  
if (start != end) {  
 int length = 0;  
 int past = -1;  
 int temp = start;  
 while (temp != end) {  
 cout << G.vexs[temp].name << "-->";  
 int vex\_on\_road[10], num = 0;  
 for (int i = 0; i < G.vexnum; i++) {  
 if (G.arcs[temp][i] != 0 && P[start][end][i] && i != temp && i != past) {  
 vex\_on\_road[num++] = i;  
 }  
 }  
 int min = vex\_on\_road[0];  
 for (int j = 1; j < num; j++) {  
 if (G.arcs[temp][vex\_on\_road[j]] < min)  
 min = vex\_on\_road[j];  
 }  
 past = temp;  
 temp = min;  
 length += G.arcs[past][temp];  
 }  
 cout << G.vexs[end].name << endl;  
 cout << "The length of the road is : " << length << endl;  
} else {  
 cout << G.vexs[start].name << "-->" << G.vexs[end].name << endl;  
 cout << "The length of the road is : " << 0 << endl;  
}

**一：当起点和重点是同一点时：**

直接输出

**二：当起点和终点不是同一点时：**

①P[起点][终点]数组中保存着路径，比如，P[7][5][3] == true表示景点4是从景点8到景点6的最短路径上的点（因为景点编号是其在数组中的下标+1）

②用一个temp表示当前输出结点，初始为start，用一个past表示上一次输出的结点（防止输出上一次已经输出的结点），初始为-1。用一个length记录最短路径长度，初始为0.

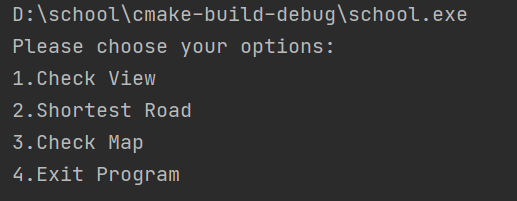
③当当前输出结点不是end结点时，先输出当前结点的信息，然后将所有与当前结点相邻的、在从start到end的最短路径上的、不是当前结点、不是上一次输出的结点的结点放到一个数组中，然后看当前结点到这个数组中的哪一个的距离最短，将它作为下一个输出结点。

④更新temp，past，length。

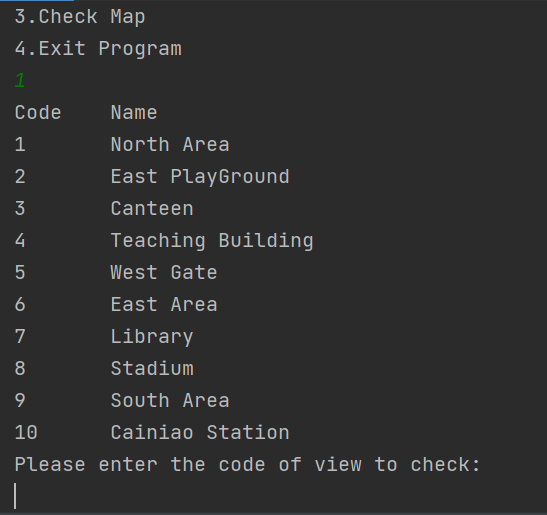
1. **实验用测试数据和相关结果分析**

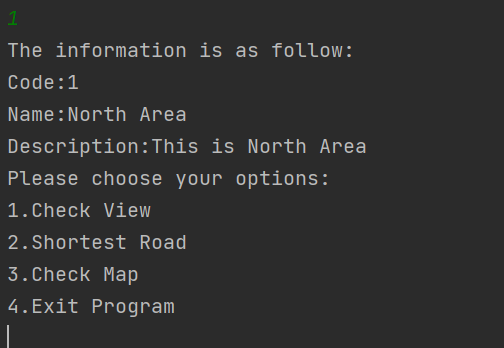
**5.1实验结果**

输入提示：



**查看景点信息**

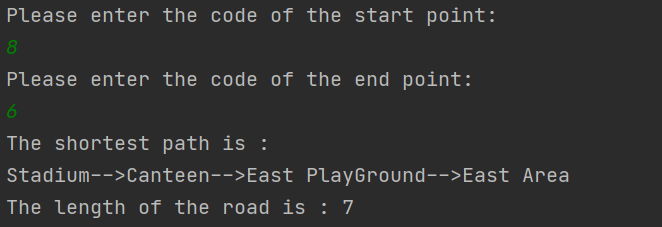


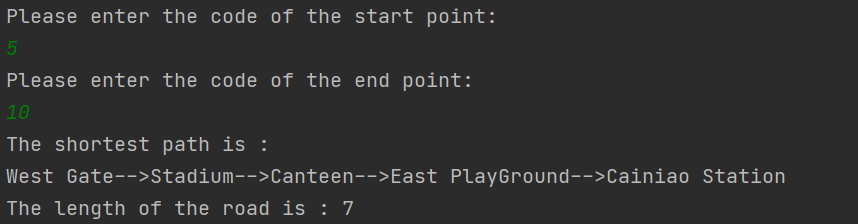


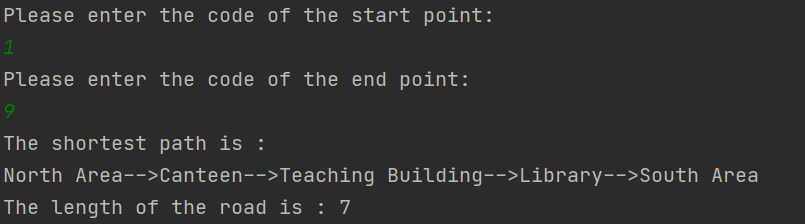
**查看地图**

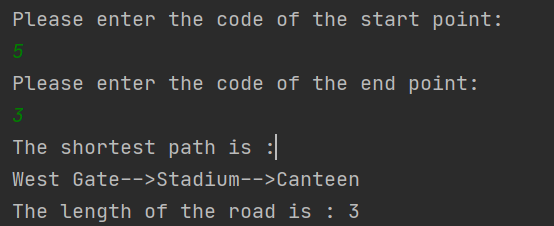


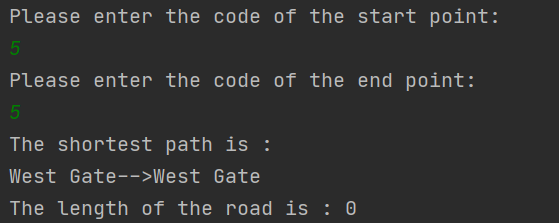
**最短路径**



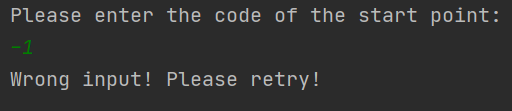








**错误输入的相应提示**



5.2实验心得

通过本次实验，我更加了解了FLOYD算法求任意两点之间最短路径的原理。

在构建D数组和P数组的过程中，我在构建D数组的P数组的时候都分别踩雷了。

构建D数组的时候，我一开始就直接把图的邻接矩阵赋值给D数组了，后来才意识到不对，于是把邻接矩阵的0换成了999，后来还是不对，还要把主对角线元素改成0.这里还有一个未解决的问题，就是为什么INFINITY不行（用INFINITY效果和用0 是一样的不知道为什么）。因为任意简单路径的长度都是小于999的，所以对于这次实验我自己给定的数据是没有问题的，当然这是很不严谨的。

构建P数组的时候，我一开始用的并不是课本上写的算法7.16，而是课后作业里构建P数组的那种方法，就是int P[10][10].如果P[v][w] != w, 则P[v][w]是从v到w的最短路径的路径中w的前一个结点。另k = P[v][w]，再判断p[v][k]是否等于k，直至p[v][k] == v。用这个方法是没问题的，不过P数组的初始化应该是P[v][w] = v。但是我一开始根据P输出路径写错了，改完之后发现还是不对，有一个问题是：只要最短路径的中间结点的序号比最短路径的中间结点的序号的最大值小，它就输出不出来，这显然是不合理的。因为我当时的写法是：

for (int u = 0; u < G.vexnum; ++u) {  
 for (int v = 0; v < G.vexnum; ++v) {  
 for (int w = 0; w < G.vexnum; ++w) {  
 if (D[v][u] + D[u][w] < D[v][w]) {  
 D[v][w] = D[v][u] + D[u][w];  
 P[v][w] = u;  
 }  
 }  
 }  
}

问题就出在P[v][w] = u这里。这么写的话最后保留的是u最大的那一次的循环结果。正确的写法应该是P[v][w] = P[v][u]。当时没有发现这个问题，也一直没做出来，才改用课本上的方法，后来才想明白的。也多亏如此，我对于FLOYD算法的理解也更深一步。

根据P矩阵输出路径的过程也有坑。一开始我只想到了要记录上一步的输出，判断是否相邻、判断是否在最短路径上。但这样会有一个问题，就是这种情况下无法判断哪一个结点是下一步要输出的结点：就是如果有多个点既与当前结点相邻，又在最短路径上。这时只要取当前结点到这些点中距离最短的那个点就行了。

1. **代码**

#include <stdio.h>  
#include <iostream>  
#include <string>  
  
#define MAX\_VERTEX\_NUM 10  
using namespace std;  
  
typedef enum {  
 *DG*, *DN*, *UDG*, *UDN*} GraphKind;  
  
typedef struct {  
 string name;  
 int code;  
 string description;  
} VertexType;  
  
typedef struct {  
 VertexType vexs[MAX\_VERTEX\_NUM];  
 int arcs[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];  
 int vexnum, arcnum;  
 GraphKind kind;  
} MGraph;  
  
void init(MGraph &G) {  
 G.kind = *UDG*;  
 G.vexs[0].name = "North Area";  
 G.vexs[1].name = "East PlayGround";  
 G.vexs[2].name = "Canteen";  
 G.vexs[3].name = "Teaching Building";  
 G.vexs[4].name = "West Gate";  
 G.vexs[5].name = "East Area";  
 G.vexs[6].name = "Library";  
 G.vexs[7].name = "Stadium";  
 G.vexs[8].name = "South Area";  
 G.vexs[9].name = "Cainiao Station";  
 for (int i = 0; i < 10; i++) {  
 G.vexs[i].code = i;  
 G.vexs[i].description = "This is " + G.vexs[i].name;  
 for (int j = 0; j < 10; j++) {  
 G.arcs[i][j] = 0;  
 }  
 }  
 G.vexnum = 10;  
 G.arcnum = 14;  
 G.arcs[0][2] = 1;  
 G.arcs[0][7] = 3;  
 G.arcs[1][2] = 3;  
 G.arcs[1][5] = 2;  
 G.arcs[1][9] = 1;  
 G.arcs[2][0] = 1;  
 G.arcs[2][1] = 3;  
 G.arcs[2][3] = 2;  
 G.arcs[2][4] = 4;  
 G.arcs[2][7] = 2;  
 G.arcs[3][2] = 2;  
 G.arcs[3][4] = 3;  
 G.arcs[3][6] = 3;  
 G.arcs[3][9] = 5;  
 G.arcs[4][2] = 4;  
 G.arcs[4][3] = 3;  
 G.arcs[4][7] = 1;  
 G.arcs[5][1] = 2;  
 G.arcs[5][9] = 3;  
 G.arcs[6][3] = 3;  
 G.arcs[6][8] = 1;  
 G.arcs[7][0] = 3;  
 G.arcs[7][2] = 2;  
 G.arcs[7][4] = 1;  
 G.arcs[8][6] = 1;  
 G.arcs[9][1] = 1;  
 G.arcs[9][3] = 5;  
 G.arcs[9][5] = 3;  
}  
  
void PrintHint() {  
 cout << "Please choose your options:" << endl;  
 cout << "1.Check View" << endl;  
 cout << "2.Shortest Road" << endl;  
 cout << "3.Check Map" << endl;  
 cout << "4.Exit Program" << endl;  
}  
  
void CheckView(MGraph &G) {  
 cout << "Code\tName" << endl;  
 for (int i = 0; i < 10; i++) {  
 cout << G.vexs[i].code + 1 << "\t" << G.vexs[i].name << endl;  
 }  
 cout << "Please enter the code of view to check:" << endl;  
 int choice;  
 cin >> choice;  
 while (choice < 1 || choice > 10) {  
 cout << "Wrong input! Please retry!" << endl;  
 cin >> choice;  
 }  
 cout << "The information is as follow:" << endl;  
 cout << "Code:" << choice << endl;  
 cout << "Name:" << G.vexs[choice - 1].name << endl;  
 cout << "Description:" << G.vexs[choice - 1].description << endl;  
}  
void CheckMap(MGraph &G) {  
 cout << " NorthArea 1\n"  
 " / |\n"  
 " 3/ 1|\n"  
 " / |\n"  
 " / |\n"  
 " / 2 | 3 2\n"  
 " Stadium 8 - - - - Canteen 3 - - - - - - - - EastPlayground 2 - - - EastArea 6\n"  
 " | / | |1 /2\n"  
 " | / | | /\n"  
 " 1| /4 | CainiaoStation 10\n"  
 " | / | 2 /\n"  
 " | / | /5\n"  
 " | / 3 | /\n"  
 " WestGate 5 - - - - - TeachingBuilding 4\n"  
 " |\n"  
 " |3\n"  
 " Library 7\n"  
 " |1\n"  
 " SouthArea 9\n";  
}  
void ShortestRoad(MGraph &G) {  
 CheckMap(G);  
 cout << "Please enter the code of the start point:" << endl;  
 int start;  
 cin >> start;  
 while (start < 1 || start > 10) {  
 cout << "Wrong input! Please retry!" << endl;  
 cin >> start;  
 }  
 cout << "Please enter the code of the end point:" << endl;  
 int end;  
 cin >> end;  
 while (end < 1 || end > 10) {  
 cout << "Wrong input! Please retry!" << endl;  
 cin >> end;  
 }  
 start -= 1;  
 end -= 1;  
 // Flory  
 int D[10][10];  
 bool P[10][10][10];  
 for (int v = 0; v < G.vexnum; ++v) {  
 for (int w = 0; w < G.vexnum; ++w) {  
 if (G.arcs[v][w] == 0) {  
 D[v][w] = 999;  
 } else {  
 D[v][w] = G.arcs[v][w];  
 }  
 if (v == w)  
 D[v][w] = 0;  
 for (int u = 0; u < G.vexnum; ++u) {  
 P[v][w][u] = false;  
 }  
 if (D[v][w] < 999) {  
 P[v][w][v] = true;  
 P[v][w][w] = true;  
 }  
 }  
 }  
  
 for (int u = 0; u < G.vexnum; ++u) {  
 for (int v = 0; v < G.vexnum; ++v) {  
 for (int w = 0; w < G.vexnum; ++w) {  
 if (D[v][u] + D[u][w] < D[v][w]) {  
 D[v][w] = D[v][u] + D[u][w];  
 for (int i = 0; i < G.vexnum; ++i) {  
 P[v][w][i] = P[v][u][i] || P[u][w][i];  
 }  
 }  
 }  
 }  
 }  
  
 //Flory end  
 cout << "The shortest path is :" << endl;  
 if (start != end) {  
 int length = 0;  
 int past = -1;  
 int temp = start;  
 while (temp != end) {  
 cout << G.vexs[temp].name << "-->";  
 int vex\_on\_road[10], num = 0;  
 for (int i = 0; i < G.vexnum; i++) {  
 if (G.arcs[temp][i] != 0 && P[start][end][i] && i != temp && i != past) {  
 vex\_on\_road[num++] = i;  
 }  
 }  
 int min = vex\_on\_road[0];  
 for (int j = 1; j < num; j++) {  
 if (G.arcs[temp][vex\_on\_road[j]] < min)  
 min = vex\_on\_road[j];  
 }  
 past = temp;  
 temp = min;  
 length += G.arcs[past][temp];  
 }  
 cout << G.vexs[end].name << endl;  
 cout << "The length of the road is : " << length << endl;  
 } else {  
 cout << G.vexs[start].name << "-->" << G.vexs[end].name << endl;  
 cout << "The length of the road is : " << 0 << endl;  
 }  
  
}  
  
  
int main() {  
 MGraph G;  
 init(G);  
 int choice;  
 PrintHint();  
 cin >> choice;  
 while (true) {  
 switch (choice) {  
 case 1:  
 CheckView(G);  
 PrintHint();  
 cin >> choice;  
 break;  
 case 2:  
 ShortestRoad(G);  
 PrintHint();  
 cin >> choice;  
 break;  
 case 3:  
 CheckMap(G);  
 PrintHint();  
 cin >> choice;  
 break;  
 case 4:  
 exit(0);  
 default:  
 cout << "Wrong input! Please retry!" << endl;  
 PrintHint();  
 cin >> choice;  
 break;  
 }  
 }  
 return 0;  
}