课 程 设 计 报 告

设计题目：校园导航
学生姓名：刘铭源
专　　业：软件工程
班　　级：１８－４
指导教师：张晶
完成日期：２０１９.６.２４

合肥工业大学软件学院

一、需求分析

给定校园内 10 个以上的地点及路径图，给出推荐路径。

要求：

构建这些地点之间的带权图，权重设置为距离或时间；

任意输入两个地点，给出推荐路径，需给出自行车和步行两种方案。

二、设计

设计一个对象地图 AdjacencyLis 包含初始化地图信息，创建地图，计算各个顶点的最短路径，输出路径长度和信息、三个结构体 EdgeNode、VertexNode、GraphAdjList 分别存储下标，权值，数据，顶点，边数信息。

为了使校园导航地图使用方便，设计了图片显示和后台输入，当程序运行时，系统把校园平面地图显示出来，在后台输入需要的查看的路径。

图片显示：

1. 首先我用高德地图截图一张合肥工业大学的图片。

2. 在地图旁边备注系统所需要的顶点信息，根据信息进行排序。

3. 使用 system("mspaint 地图名字.png")和 while{1}使地图一直循环使用一直显示在我们屏幕上。

地图(弗洛伊德算法)

算法思想：

Floyd 算法是一个经典的动态规划算法。用通俗的语言来描述的话，首先我们的目标是寻找从点 i 到点 j 的最短路径。从任意节点 i 到任意节点 j 的最短路径不外乎 2 种可能,1 是直接从 i 到 j，2 是从 i 经过若干个节点 k 到 j。所以，我们假设 Dis(i,j)为节点 u 到节点 v 的最短路径的距离，对于每一个节点 k，我们检查 Dis(i,k) + Dis(k,j) < Dis(i,j)是否成立，如果成立，证明从 i 到 k 再到 j 的路径比 i 直接到 j 的路径短,我们便设置 Dis(i,j) = Dis(i,k) + Dis(k,j)，这样一来，当我们遍历完所有节点 k，Dis(i,j)中记录的便是 i 到 j 的最短路径的距离。

步骤：

初始状态：Dis 是记录各个顶点间最短路径的矩阵。

第 1 步：初始化 D。

矩阵 D 中顶点 a[i][j]的距离为顶点 i 到顶点 j 的权值；如果 i 和 j 不相邻，则 a[i][j]=∞。实际上，就是将图的原始矩阵复制到 S 中。

注:a[i][j]表示矩阵 D 中顶点 i(第 i 个顶点)到顶点 j(第 j 个顶点)的距离。

第 2 步：以顶点 A(第 1 个顶点)为中介点，若 a[i][j] > a[i][0]+a[0][j]，则设置 a[i][j]=a[i][0]+a[0][j]。

以顶点 a[1]6，上一步操作之后，a[1][6]=∞；而将 A 作为中介点时，(B,A)=12，(A,G)=14，因此 B 和 G 之间的距离可以更新为。同理，依次将顶点 B,C,D,E,F,G 作为中介点，并更新 a[i][j]的大小。

地图具体设计

1. 第一步设立三个结构体 EdgeNode、VertexNode、GraphAdjList 保存信息

2. 构建地图类，包含构造函数（初始化地图）、显示函数，计算各个顶点的最短路径，输出各个顶点的最短路径。

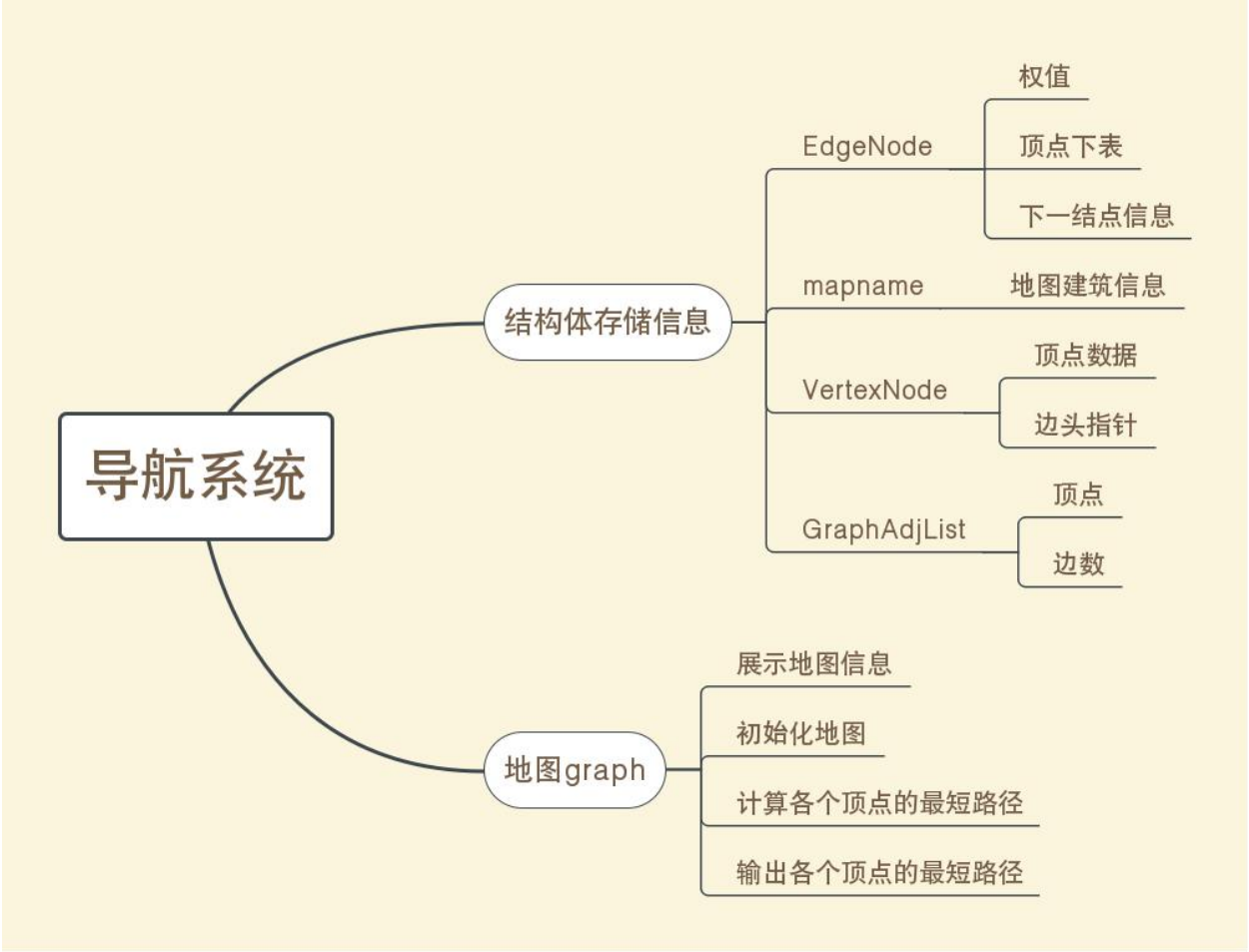
3. 创建地图：建立顶点表，读入顶点信息，将边表置为空表。第二步使用头插法建立边表，计算各个顶点之间的最短路径

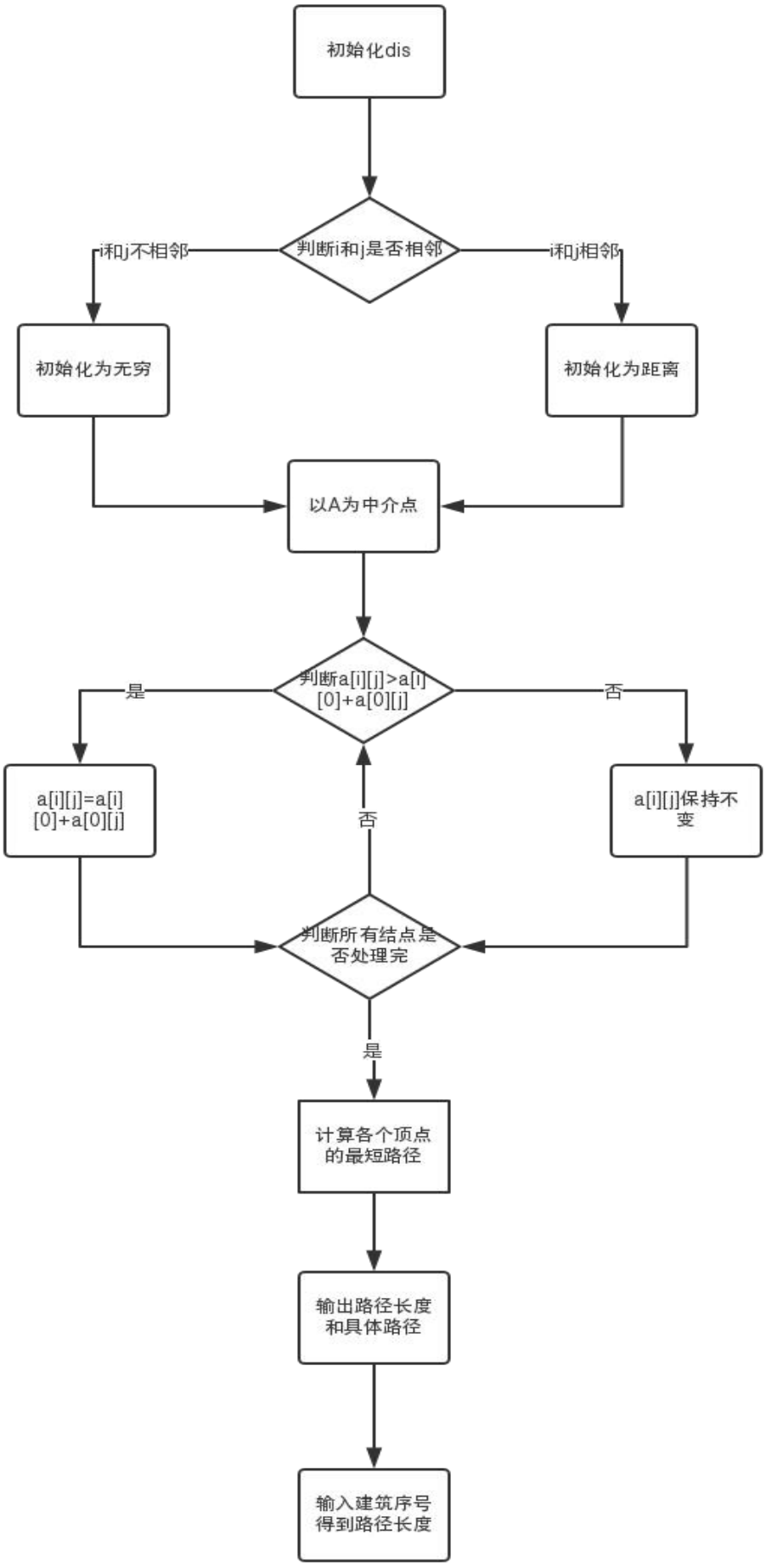
4. 初始化地图数据，输入定点数和边数

5. 输出路径长度和具体路径，初始化 path 与 dis。

6. 输入地点序号，关闭地图，显示路径长度和具体路径。

三、系统框架图：

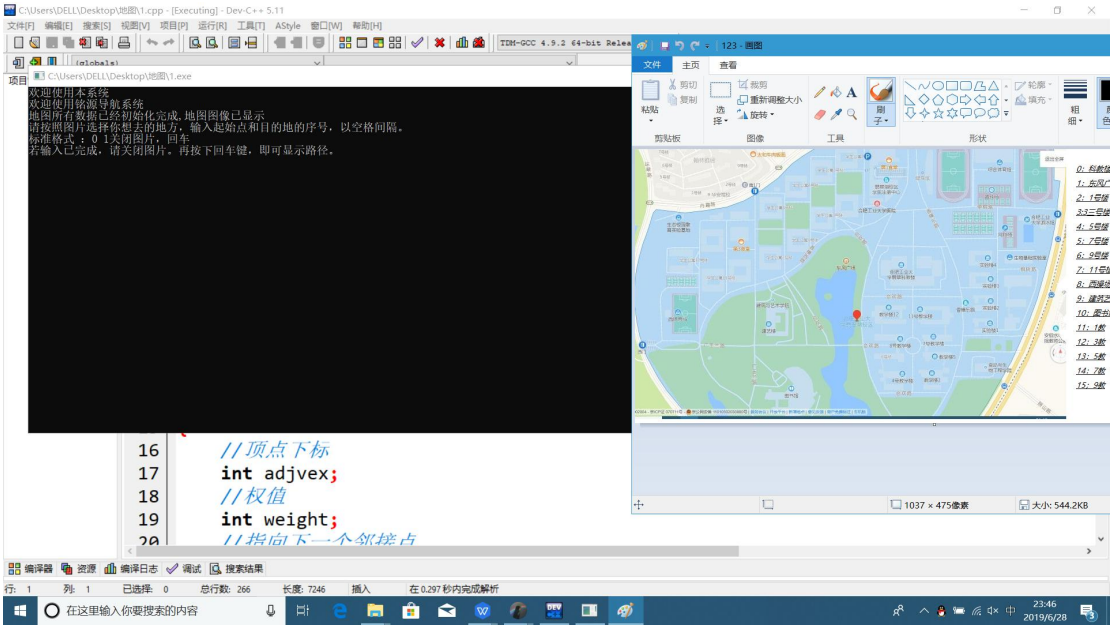

导航系统
结构体存储信息
EdgeNode
权值
顶点下表
下一结点信息
mapname
地图建筑信息
VertexNode
顶点数据
边头指针
GraphAdjList
顶点
边数
地图graph
展示地图信息
初始化地图
计算各个顶点的最短路径
输出各个顶点的最短路径

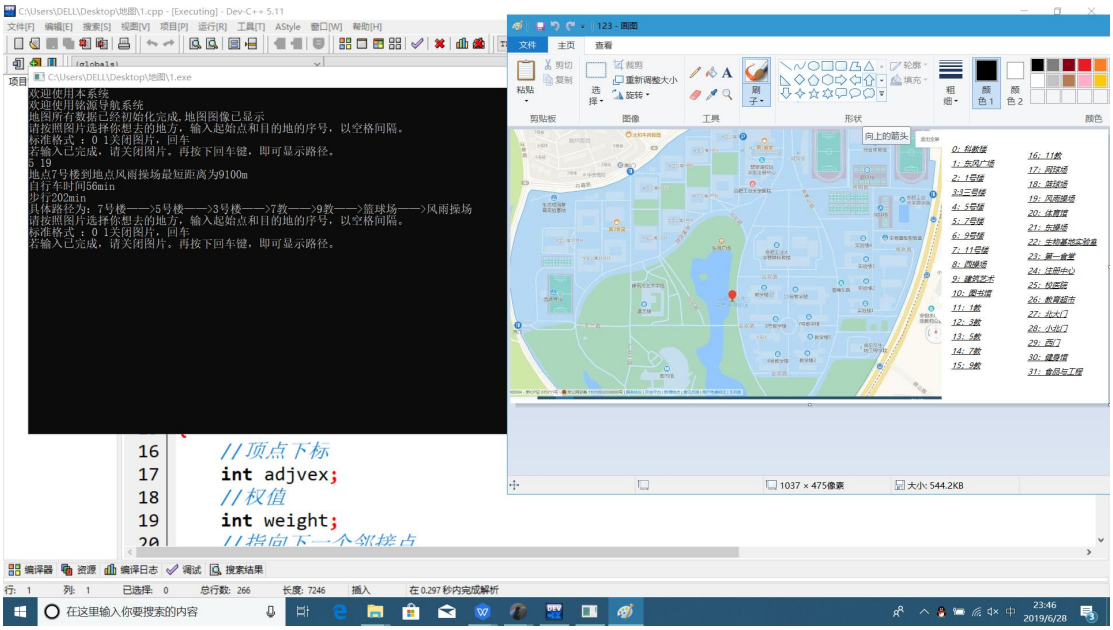

初始化dis
判断i和j是否相邻
i和j不相邻
i和j相邻
初始化为无穷
初始化为距离
以A为中介点
判断a[i][j]>a[i][0]+a[0][j]
是
否
a[i][j]=a[i][0]+a[0][j]
a[i][j]保持不变
否
判断所有结点是否处理完
是
计算各个顶点的最短路径
输出路径长度和具体路径
输入建筑序号得到路径长度

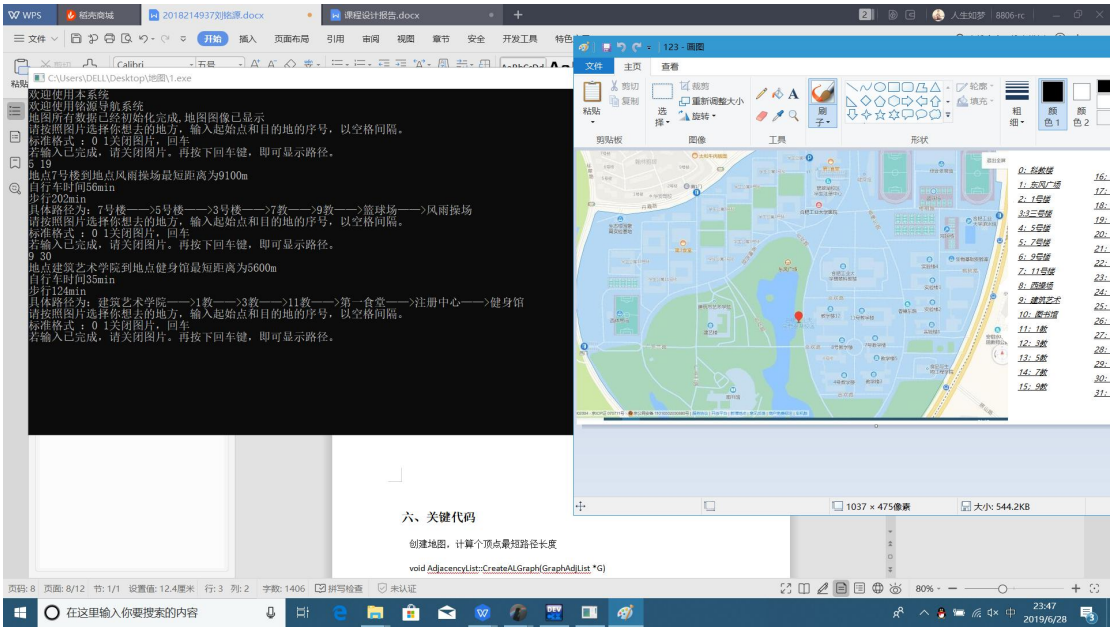
四、属性与方法定义

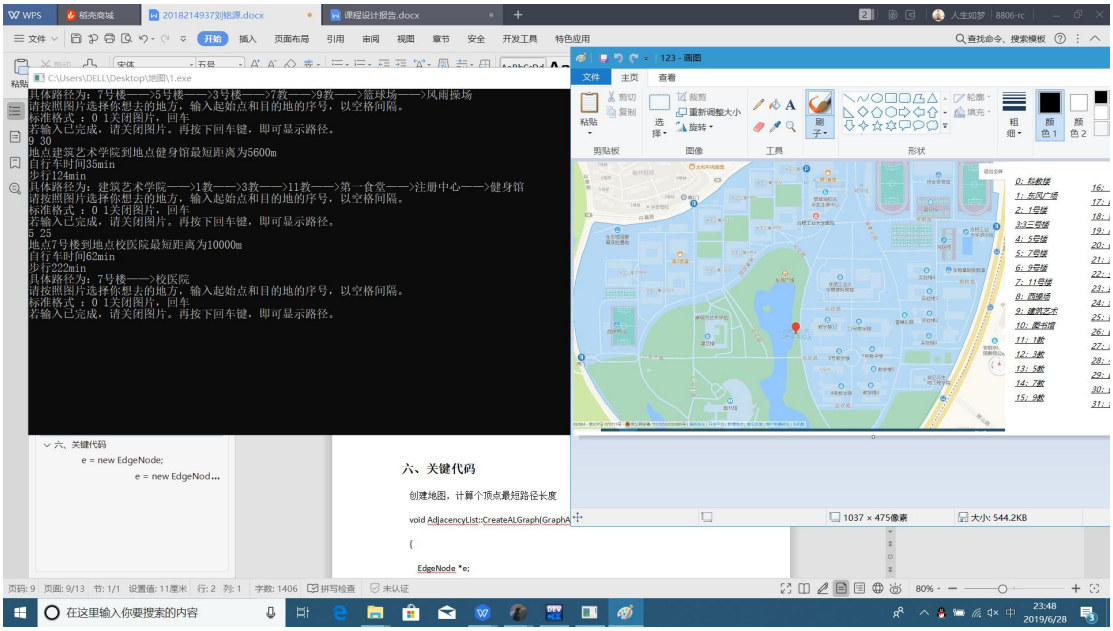
类/结构	函数名/变量	作用
EdgeNode	Adjvex、weight、*next	存储边的信息
VertexNode	data[50]、*firstedge	顶点表结点
GraphAdjList	adjList	顶点和边数
AdjacencyList	ShowALGraph	展示地图信息
AdjacencyList	Test	显示界面
AdjacencyList	InitMap	初始化地图
AdjacencyList	CreateALGraph	计算各个顶点信息
AdjacencyList	ShortestPath_Floyd	输出各个顶点信息

五、调试与测试


欢迎使用本系统
欢迎使用铭源导航系统
地图所有数据已经初始化完成,地图图像已显示
请按照图片选择你想去的地方，输入起始点和目的地的序号，以空格间隔。
标准格式：0 1关闭图片，回车
若输入已完成，请关闭图片。再按下回车键，即可显示路径。
//顶点下标
int adjvex;
//权值
int weight;


欢迎使用本系统
欢迎使用铭源导航系统
地图所有数据已经初始化完成,地图图像已显示
请按照图片选择你想去的地方，输入起始点和目的地的序号，以空格间隔。
标准格式：0 1关闭图片，回车
若输入已完成，请关闭图片。再按下回车键，即可显示路径。
6 19
地点7号楼到地点风雨操场最短距离为9100m
自行车时间56min
步行202min
具体路径为：7号楼——>5号楼——>3号楼——>7教——>9教——>篮球场——>风雨操场
请按照图片选择你想去的地方，输入起始点和目的地的序号，以空格间隔。
标准格式：0 1关闭图片，回车
若输入已完成，请关闭图片。再按下回车键，即可显示路径。
//顶点下标
int adjvex;
//权值
int weight;





六、参考文献

Cdsn--弗洛伊德算法

Csdn--弗洛伊德实现

Csdn--如何缓存图片

七、感受感悟

通过这次数据结构课程设计，让我明白了的有向图和无向图的区别，通过此次课程设计帮助我复习了弗洛伊德算法、动态规划的思想、迪杰斯特拉算法在求最短路径时如何设计算法完成对最短路径的求解，我查找了资料进一步明白了动态规划的思想和完善了对图的认识。

八、关键代码

创建地图，计算个顶点最短路径长度

```
void AdjacencyList::CreateALGraph(GraphAdjList *G)
{
   EdgeNode *e;
   //读入顶点信息，建立顶点表
    for (int i = 0; i < G->numVertexes; i++)
    {
      //读入顶点信息
            strcpy(G->adjList[i].data, _mapName[i]);
            //将边表置为空表
            G->adjList[i].firstedge = NULL;
   }
   //建立边表（头插法）
   for (int i = 0; i < G->numVertexes; i++)
      {
```

```
for (int j = 0; j < i; j++)
{
 int temp;
      if (_distance[i][j] != 0 || _distance[j][i] != 0)
      {
       if(_distance[i][j] != 0)
       {
       temp = _distance[i][j];
            _distance[j][i] = _distance[i][j];
       }
       else
       {
            temp = _distance[j][i];
            _distance[i][j] = _distance[j][i];

       }
       e = new EdgeNode;
       e->adjvex = j;
       e->next = G->adjList[i].firstedge;
       e->weight = temp;
            G->adjList[i].firstedge = e;
            e = new EdgeNode;
```

```
                    e->adjvex = i;
                    e->next = G->adjList[j].firstedge;
                    e->weight = temp;
                    G->adjList[j].firstedge = e;
                }
            }
        }
}
```

输出最短路径长度和具体路径

```
void AdjacencyList::ShortestPath_Floyd(GraphAdjList *G, int P[32][32], int D[32][32])
{
  //初始化 d 与 p
  for(int v = 0; v < G->numVertexes; ++v)
     {
          for (int w = 0; w < G->numVertexes; ++w)
          {
            if(_distance[v][w]==0&&v!=w)
            {
                 _distance[v][w] = 10000;
              }
```

```
                    D[v][w] = _distance[v][w];

                    P[v][w] = w;

            }

    }

    for (int k = 0; k < G->numVertexes; ++k)

    {

            for (int v = 0; v < G->numVertexes; ++v)

            {

              for (int w = 0; w < G->numVertexes; ++w)

              {

                    if (D[v][w] > D[v][k] + D[k][w])

                    {

                        D[v][w] = D[v][k] + D[k][w];

                              P[v][w] = P[v][k];

                    }

               }

            }

    }

}
```