```cpp
//before
#include"StackAndQueue.h"

//-----------------------------------------------------图的邻接表法
#include<iostream>
#include<stdio.h>
using namespace std;

//----------------------写在前边
#define InfoType int
#define VertexType char
#define INFINITY  INT_MAX //最大值表示无联通

typedef enum{DG,DN,UDG,UDN}GraphKind;     //枚举型变量：有向图、有向网、无向图、无向网
typedef enum{ERROR,OK}Status;    //枚举型，函数状态变量

#define MAX_VERTEX_NUM 20

typedef struct ArcNode{
    int adjvex;      //该弧所指向的顶点位置
    struct ArcNode *nextarc;     //指向下一条弧的指针

    InfoType *info;              //该弧相关信息的指针
}ArcNode;

typedef struct VNode{
    VertexType data;             //顶点信息
    ArcNode *firstarc;           //指向第一条依附该顶点的弧的指针
}VNode,AdjList[MAX_VERTEX_NUM];

typedef struct{
    AdjList vertices;            //顶点数组
    int vexnum, arcnum;          //图中顶顶及弧的数目
    GraphKind kind;              //图的种类
}ALGraph;

//-------------------------------------------------------------------基本函数
//若G中存在顶点u，则返回u在G中的位置，若没找到则返回-1
int LocateVex(ALGraph G, VertexType v){
    int i=0;
    for(i=0; i<G.vexnum; i++){
        if(G.vertices[i].data==v)
            break;
    }
    if(i<G.vexnum)
        return i;
    else
        return -1;
}

//构造有向图
Status CreateDG(ALGraph &G){
    char IncInfo;        //0/1用来表示弧边是否含有辅助信息
    cout <<"Please input: vexnum(no more than 20) arcnum(no more than vexnum*vexnum) IncInfo(default 0)" << endl;
    cin >> G.vexnum >> G.arcnum >> IncInfo;      //0/1用来表示弧边是否含有辅助信息
    cout << "构造顶点向量" << endl;
    for(int i=0; i<G.vexnum; i++){
//构造顶点向量并初始化第一条依附顶点的指针
        cin >> G.vertices[i].data;
        G.vertices[i].firstarc = NULL;
    }

    VertexType v1,v2; //顶点
    int v1_int,v2_int;
    ArcNode *p;
    cout << "以'起点 终点'的方式依次输入每一条边(例如：ab\t起点：a，终点：b)：" << endl;
    for(int i=0; i<G.arcnum; i++){                //构造邻接表
        cin >> v1 >> v2;
        //定位v1,v2在G中的位置
        v1_int=LocateVex(G,v1);
        v2_int=LocateVex(G,v2);
        p = new ArcNode;
        p->adjvex = v2_int;
        p->info = NULL;
        p->nextarc = G.vertices[v1_int].firstarc;//以头插法的方式讲P插入到v1的第一个弧边上
        G.vertices[v1_int].firstarc = p;
            // if(IncInfo)
            //      Inuput(*G.arcs[i][j].info);
            //      若弧边上有辅助信息，则输入
    }
    return OK;
}

```

```cpp
//构造有向网
Status CreateDN(ALGraph &G){
    char IncInfo;       //0/1用来表示弧边是否含有辅助信息
    cout <<"Please input: vexnum(no more than 20) arcnum(no more than vexnum*vexnum) IncInfo(default 0)" << endl;
    cin >> G.vexnum >> G.arcnum >> IncInfo;     //0/1用来表示弧边是否含有辅助信息
    cout << "构造顶点向量" << endl;
    for(int i=0; i<G.vexnum; i++){
//构造顶点向量并初始化第一条依附顶点的指针
        cin >> G.vertices[i].data;
        G.vertices[i].firstarc = NULL;
    }

    VertexType v1,v2; //顶点
    int v1_int,v2_int;
    InfoType w;         //顶点信息
    ArcNode *p;
    cout << "以'起点 终点 权值'的方式依次输入每一条边(例如：ab9\t起点：a，终点：b，权值：9)：" << endl;
    for(int i=0; i<G.arcnum; i++){              //构造邻接表
        cin >> v1 >> v2 >> w;
        //定位v1,v2在G中的位置
        v1_int=LocateVex(G,v1);
        v2_int=LocateVex(G,v2);
        p = new ArcNode;
        p->adjvex = v2_int;
        p->info = new InfoType;
        *p->info = w;
        p->nextarc = G.vertices[v1_int].firstarc;//以头插法的方式讲p插入到v1的第一个弧边上
        G.vertices[v1_int].firstarc = p;
            // if(IncInfo)
            //      Input(*G.arcs[i][j].info);
            //      若弧边上有辅助信息，则输入
    }
    return OK;
}

//构造无向图
Status CreateUDG(ALGraph &G){
    char IncInfo;       //0/1用来表示弧边是否含有辅助信息
    cout <<"Please input: vexnum(no more than 20) arcnum(no more than vexnum*vexnum) IncInfo(default 0)" << endl;
    cin >> G.vexnum >> G.arcnum >> IncInfo;     //0/1用来表示弧边是否含有辅助信息
    cout << "构造顶点向量" << endl;
    for(int i=0; i<G.vexnum; i++){
//构造顶点向量并初始化第一条依附顶点的指针
        cin >> G.vertices[i].data;
        G.vertices[i].firstarc = NULL;
    }

    VertexType v1,v2; //顶点
    int v1_int,v2_int;
    ArcNode *p;
    cout << "以'起点 终点'的方式依次输入每一条边(例如：ab\t起点：a，终点：b)：" << endl;
    for(int i=0; i<G.arcnum; i++){              //构造邻接表
        cin >> v1 >> v2;
        //定位v1,v2在G中的位置
        v1_int=LocateVex(G,v1);
        v2_int=LocateVex(G,v2);
        {//无向图中单方向
            p = new ArcNode;
            p->adjvex = v2_int;
            p->info = NULL;
            p->nextarc = G.vertices[v1_int].firstarc;//以头插法的方式讲p插入到v1的第一个弧边上
            G.vertices[v1_int].firstarc = p;
                // if(IncInfo)
                //      Input(*G.arcs[i][j].info);
                //      若弧边上有辅助信息，则输入
        }
        {//无向图中反方向
            p = new ArcNode;
            p->adjvex = v1_int;
            p->info = NULL;
            p->nextarc = G.vertices[v2_int].firstarc;//以头插法的方式讲p插入到v1的第一个弧边上
            G.vertices[v2_int].firstarc = p;
                // if(IncInfo)
                //      Input(*G.arcs[i][j].info);
                //      若弧边上有辅助信息，则输入
        }
    }
    return OK;
}
```

```
160
161 //构造无向网
162 Status CreateUDN(ALGraph &G){
163     char IncInfo;        //0/1用来表示弧边是否含有辅助信息
164     cout <<"Please input: vexnum(no more than 20) arcnum(no more than vexnum*vexnum)
    IncInfo(default 0)" << endl;
165     cin >> G.vexnum >> G.arcnum >> IncInfo;      //0/1用来表示弧边是否含有辅助信息
166     cout << "构造顶点向量" << endl;
167     for(int i=0; i<G.vexnum; i++){
    //构造顶点向量并初始化第一条依附顶点的指针
168         cin >> G.vertices[i].data;
169         G.vertices[i].firstarc = NULL;
170     }
171
172     VertexType v1,v2; //顶点
173     int v1_int,v2_int;
174     InfoType w;          //顶点信息
175     ArcNode *p;
176     cout << "以\起点 终点
    权值'的方式依次输入每一条边(例如: ab9\t起点: a, 终点: b, 权值: 9): " << endl;
177     for(int i=0; i<G.arcnum; i++){               //构造邻接表
178         cin >> v1 >> v2 >>w;
179         //定位v1,v2在G中的位置
180         v1_int=LocateVex(G,v1);
181         v2_int=LocateVex(G,v2);
182         {//无向图中单方向
183             p = new ArcNode;
184             p->adjvex = v2_int;
185             p->info = new InfoType;
186             *p->info = w;
187             p->nextarc =
    G.vertices[v1_int].firstarc;//以头插法的方式讲p插入到v1的第一个弧边上
188             G.vertices[v1_int].firstarc = p;
189                 // if(IncInfo)
190                 //      Inuput(*G.arcs[i][j].info);
191                 //      若弧边上有辅助信息, 则输入
192         }
193         {//无向图中反方向
194             p = new ArcNode;
195             p->adjvex = v1_int;
196             p->info = new InfoType;
197             *p->info = w;
198             p->nextarc =
    G.vertices[v2_int].firstarc;//以头插法的方式讲p插入到v1的第一个弧边上
199             G.vertices[v2_int].firstarc = p;
200                 // if(IncInfo)
201                 //      Inuput(*G.arcs[i][j].info);
202                 //      若弧边上有辅助信息, 则输入
203         }
204     }
205     return OK;
206 }
207
208 //创建ALGraph
209 Status CreateGraph(ALGraph &G){
210     cout << "选择图的类型:     " << endl;
211     cout << "DG,DN,UDG,UDN   有向图、有向网、无向图、无向网" << endl;
212     scanf("%d",&G.kind);
213     switch(G.kind){
214         case DG: return CreateDG(G);     //构造有向图
215         case DN: return CreateDN(G);     //构造有向网
216         case UDG: return CreateUDG(G);     //构造无向图
217         case UDN: return CreateUDN(G);     //构造无向网
218         default: return ERROR;
219     }
220 }
221
222 //打印邻接表
223 Status ALGraphShow(ALGraph G){
224
225     cout << "ALGraph 顶点信息: " << endl;
226     for(int i=0; i<G.vexnum; i++)
227         cout << G.vertices[i].data << "\t";
228     cout << endl << endl;
229
230     cout << "ALGraph 邻接链表: " << endl;
231     ArcNode *p;
232     switch(G.kind){
233         case DG:
234         case UDG:
235             {
236                 for(int i=0; i<G.vexnum; i++){
237                     p=G.vertices[i].firstarc;
238                     cout << G.vertices[i].data << "  |\t";
```

```
                    while(p){
                        cout << G.vertices[p->adjvex].data;
                        if( p->nextarc )
                            cout << " --> ";
                        p=p->nextarc;
                    }
                    cout << endl;
                }
                break;
            }
        case DN:
        case UDN:
            {
                for(int i=0; i<G.vexnum; i++){
                        p=G.vertices[i].firstarc;
                        cout << G.vertices[i].data << "  |\t";
                        while(p){
                            cout << G.vertices[p->adjvex].data  << "_" << *p->info;
                            if( p->nextarc )
                                cout << " --> ";
                            p=p->nextarc;
                        }
                        cout << endl;
                    }
            }

        default: return ERROR;
    }
    return OK;
}

//求某个顶点的入度
int FindInDegree(ALGraph G, int v){
    ArcNode *p;
    int indegree=0;
    for(int i=0; i<G.vexnum; i++){
        p = G.vertices[i].firstarc;
        while(p){
            if(p->adjvex == v)
                indegree++;
            p=p->nextarc;
        }
    }
    return indegree;
}

//-------------------------------------------------------------------------拓扑排序

//入度数组，存放每个顶点的入度
int indegree[MAX_VERTEX_NUM];

//对有向图进行拓扑排序，图G采用邻接表法的存储方式
//图中无回路，返回OK，并输出一条拓扑排序序列，若有回路则返回ERROR
Status TopologicalSort(ALGraph G){
    for(int i=0; i<G.vexnum; i++)                   //求每个顶点的入度
        indegree[i] = FindInDegree(G, i);

    SqStack S;
    InitStack_Sq(S);
    for(int i=0; i<G.vexnum; i++)
        if(indegree[i]==0)
            Push_Sq(S, i);                  //入度为0者进栈

    int count = 0;  //对输出顶点计数
    int i,k;
    ArcNode *p;
    while(!StackEmpty_Sq(S)){    //栈不为空
        Pop_Sq(S, i);
        //输出第i个顶点并计数
        cout << i << ": " << G.vertices[i].data << endl;
        count ++;
        for(p=G.vertices[i].firstarc; p; p=p->nextarc){
            k = p->adjvex;
            if(0==(--indegree[k]))   //k的入度不为0，k入栈
                Push_Sq(S, k);
        }
    }
    if(count<G.vexnum)
        return ERROR;
    else
        return OK;
}
```