```
#include"ALGraph.h"

int vl[MAX_VERTEX_NUM];     //事件的最迟发生时间
int ve[MAX_VERTEX_NUM];     //事件的最早发生时间

Status TopologicalOrder(ALGraph G, SqStack &T){
    for(int i=0; i<G.vexnum; i++)           //求每个顶点的入度
        indegree[i] = FindInDegree(G, i);

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

    InitStack_Sq(T);         //初始化栈T
    int count = 0;           //初始化计数器
    for(int i=0; i<G.vexnum; i++)    //初始化事件最早发生数组
        ve[i] = 0;

    int j;
    int k;
    ArcNode *p;
    while(!StackEmpty_Sq(S)){
        //j号顶点(入度为0)的顶点，入栈T
        Pop_Sq(S, j);   Push_Sq(T, j);      count++;
        for(p=G.vertices[j].firstarc; p; p=p->nextarc){
            k = p->adjvex;
            //对j号顶点的每个邻接顶点的入度减1，若减1后顶点的入度为1则该顶点入栈S
            if(--indegree[k]==0)
                Push_Sq(S,k);
            if(ve[j]+*(p->info) > ve[k])
                ve[k] = ve[j]+*(p->info);
        }//for
    }//while
    if(count<G.vexnum)  //网中有回路
        return ERROR;
    else
        return OK;
}//TopologicalOrder

Status CriticalPath(ALGraph G){
    //G为有向网输出G上的各项关键活动

    //拓扑排序判断图中是否有环，有则返回ERROR，无则获得拓扑排序序列T
    SqStack T;
    if(!TopologicalOrder(G,T)){
        cout << "ERROR，图中有环" << endl;
        return ERROR;
    }

    //初始化顶点的最迟发生时间
    int temp = ve[0];
    for(int i=1; i<G.vexnum; i++)
        if(ve[i]>temp)
            temp = ve[i];
    for(int i=0; i<G.vexnum; i++)
        vl[i] = temp;


    //按拓扑排序逆序求各个顶点的vl
    int j, k;
    ArcNode *p;
    int dut;
    while(!StackEmpty_Sq(T))
        for(Pop_Sq(T,j),p=G.vertices[j].firstarc; p; p=p->nextarc){
            k = p->adjvex;
            dut = *(p->info);       //dut<j,k>
            if(vl[k]-dut < vl[j])
                vl[j] = vl[k]-dut;
        }//for

    cout <<
"路径：\t持续时间：\t最早开始时间：\t最晚开始时间：\t“*”标记的为关键路径： " << endl;

    int ee, el;
    char tag;
    //求ee，el，关键路径
    for(j=0; j<G.vexnum; j++)
        for(p=G.vertices[j].firstarc; p; p=p->nextarc){
            k=p->adjvex;    dut = *(p->info);
            ee = ve[j];   el = vl[k]-dut;
            tag = (ee==el)? '*':' ';
            {   //输出所有路径，“*”标记的为关键路径
                cout << G.vertices[j].data << "_" << G.vertices[k].data << "\t    ";
                cout << dut << "\t\t\t";
                cout << ee << "\t\t";
                cout << el << "\t\t" << tag << endl;
            }
        }//for
}//CriticalPath


int main(){
    ALGraph G;
    CreateGraph(G);
    ALGraphShow(G);
    CriticalPath(G);

    return 0;
}

```