代码：

#include"stdio.h"

#include"stdlib.h"

#include"string.h"

#define M 30 /\*预定义图的最大顶点数\*/

//拓扑排序

typedef struct node { //边结点类型定义

int adjvex;

struct node \*next;

}edgenode;

typedef struct de { //待定点入度的头节点定义

edgenode\* FirstEdge;

char vertex;

int id; //顶点的入度域

}vertexnode;

typedef struct { //AOV网的邻接表结构

vertexnode adjlist[M];

int n, e;

}AovGraph;

//类似于邻接表的常见方式，不同的是在这里从文件多读入了一个顶点的入度域

void createTop(AovGraph \*g, char \*filename, int c)

{

int i, j, k;

edgenode \*s;

FILE \*fp;

fp = fopen(filename, "r");

if (fp)

{

fscanf(fp, "%d%d\n", &g->n, &g->e);

for (i = 0; i < g->n; i++)

{

fscanf(fp, "%c%d ", &g->adjlist[i].vertex, &g->adjlist[i].id);

g->adjlist[i].FirstEdge = NULL;

}

for (k = 0; k < g->e; k++)

{

fscanf(fp, "%d%d", &i, &j);

s = (edgenode \*)malloc(sizeof(edgenode));

s->adjvex = j;

s->next = g->adjlist[i].FirstEdge;

g->adjlist[i].FirstEdge = s;

if (c == 0) //无向图

{

s = (edgenode \*)malloc(sizeof(edgenode));

s->adjvex = i;

s->next = g->adjlist[j].FirstEdge;

g->adjlist[i].FirstEdge = s;

}

}

fclose(fp); //关闭文件流

}

else

{

g->n = 0;

printf("文件打开失败！\n");

}

}

//拓扑排序的算法

int TopSort(AovGraph g)

{

int k = 0, i, j, v, flag[M];

int queue[M]; //定义队列

int front, rear;

edgenode \*p;

front = rear = 0; //初始化队列

for (i = 0; i < g.n; i++)

flag[i] = 0; //访问标记初始化

for (i = 0; i < g.n; i++)

{

if (g.adjlist[i].id == 0 && flag[i] == 0)

{

queue[rear++] = i;

flag[i] = 1;

}

}

printf("\n该AOV网的拓扑排序为：\n");

while (front < rear) // 如果当前队列不为空

{

v = queue[front++]; //队列首位元素出列

printf("%c ", g.adjlist[v].vertex);

k++; //计数器加1

p = g.adjlist[v].FirstEdge;

while (p) //将所有于v邻接的顶点的入度减1

{

j = p->adjvex;

if (--g.adjlist[j].id == 0 && flag[j] == 0) //如果入度为0则将进队

{

queue[rear++] = j;

flag[j] = 1; //标记已经被访问过

}

p = p->next;

}

}

return k; //返回输出的结点个数

}

int main()

{

AovGraph g;

char filename[20] = "D:\\Desktop\\Test.txt";

createTop(&g, filename, 1);

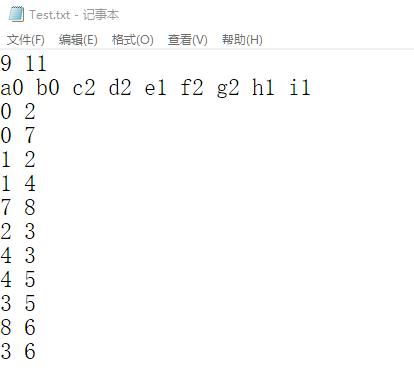
printf("拓扑排序的节点个数：%d\n", TopSort(g));

system("pause");

return 0;

}

文件结构：



结果：