**北京科技大学实验报告**

学院：计通学院 专业：物联网工程 班级：物联1602

姓名： 周康能 学号：41624657 实验日期： 2018年 6 月 11 日

**实验名称：《数据结构》实验5图的应用**

**实验目的：掌握图的存储结构，理解有向无环图和拓扑排序的概念和意义，熟悉图的基本操作及应用。**

**实验内容：**

建立有向图的十字链表存储结构，利用拓扑排序方法判断该图是否为有向无环图。

要求：有向图的顶点信息和弧信息可以从键盘或txt文件输入，输入格式自拟。

**问题分析与算法思路：**

运用带有向图的十字链表结构，并通过拓扑排序的算法来判断是否有环，可以从键盘和txt文件输入顶点数和弧结点。

还用了链式的堆栈来存储。

本程序算法简单，亮点在txt输入的处理上，用到了标志位。

本程序时间复杂度和空间复杂度都较合适。

**算法描述：**

void Creatid(vexNode G[], int id[]) //求入度

{

for (int i = 0; i < n; i++) //用逆邻接表求

{

count = 0;

p = G[i].fin;

while (p)

{

count++;

p = p->hlink;

}

id[i] = count;

}

}

此为求入度的函数

void TopSort(vexNode G[]) //十字链表，拓扑排序

{

for (int i = 0; i < n; i++)

{

if (id[i] == 0)

{

Push(mid, i);

}

}

count = 0;

printf("\n");

while (mid->next)

{

j = GetTop(mid);

Pop(mid);

//printf("%c \t", G[j].date);

count++;

p = G[j].fout;

while (p)

{

k = p->head;

id[k]--;

if (id[k] == 0)

{

Push(mid, k);

}

p = p->tlink;

}

}

if (count == n)

{

printf("This graph has no cycles.\n");

}

else

{

printf("This graph has cycles.\n");

}

}

此函数用到链式堆栈来求拓扑排序，以此来求是否有环。

void createList(vexNode G[])

{

//----------------------------顶点-------------------------------//

键盘或txt输入顶点

//---------------------------------------------------------------//

//----------------------------弧---------------------------------//

键盘或txt输入弧

具体见程序

}

**程序实现：**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Copyright(c) :Ellis

\* All rights reserved.

\*

\* 文件名称:USTB2018\_data\_structure\_test5

\* 简要描述:

\* 要求：题目：图的存储结构及拓扑排序

\* 从键盘或文件读入有向图的顶点信息和弧信息（输入格式自拟）；

\* 建立有向图的十字链表存储结构；

\* 利用拓扑排序方法判断该图是否为有向无环图。

\*

\*

\* 创建日期:20180501\_16:18

\* 作者:Ellis

\* 说明:完成所有任务

\*

\* 修改日期:

\* 作者:Ellis

\* 说明:

\* 注意：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 宏定义、头文件、命名空间区

\* 创建日期: 20180501

\* 作者 : Ellis

\* 说明 : 无

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define \_CRT\_SECURE\_NO\_DEPRECATE //规避安全检查

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include <string>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 结构体定义区

\* 创建日期: 20180501

\* 作者 : Ellis

\* 说明 : 链栈 + 顶点 + 弧结点

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct Node

{

int date;

struct Node \*next;

}NodeStack,\*LinkStack;

typedef char vtype; //一种新类型

typedef struct Anode

{

int head;

int tail;

struct Anode \*hlink, \*tlink;

}arcNode;

typedef struct Vnode

{

vtype date;

arcNode \*fin, \*fout;

}vexNode;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 全局变量区

\* 创建日期: 20180501

\* 作者 : Ellis

\* 说明 :

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int n; //记录顶点

vexNode G[100]; //顶点表

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 函数定义区

\* 创建日期: 20180501

\* 作者 : Ellis

\* 说明 : 定义各种函数,集成化，容易看

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool IsEmpty(LinkStack s); //栈判空

LinkStack CreateStack(); //创建一个栈

void Push(LinkStack top, int input); //入栈

void Pop(LinkStack top); //出栈

char GetTop(LinkStack top); //取栈顶

int loactevex(vexNode G[], char u); //求序号

void createList(vexNode G[]); //创建十字链表

void Creatid(vexNode G[], int id[]); //求入度

void TopSort(vexNode G[]); //十字链表，拓扑排序

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void main()

Author :Ellis

Date :20180501

Description :

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{

printf("Enter 'Y' to start.\n");

char a = getchar();

getchar();

while (a == 'Y')

{

createList(G);

TopSort(G);

printf("Continue or not ?\nEnter 'Y' to continue,'N' to exit.\n");

char s = getchar();

getchar();

while (1)

{

if (s == 'Y')

{ //---------------

//continue; //Y继续，N退出

break; //其他继续输入

} //---------------

else if (s == 'N')

{

return;

}

else

{

printf("Invalid!\nPlease try again.\n");

s = getchar();

getchar();

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:bool IsEmpty(LinkStack s)

Author :Ellis

Date :20180501

Description :判断堆栈是否为空

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool IsEmpty(LinkStack s)

{

return s->next ? false : true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:LinkStack CreateStack()

Author :Ellis

Date :20180501

Description :新建一个堆栈

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

LinkStack CreateStack()

{

LinkStack top = (LinkStack)malloc(sizeof(NodeStack));

top->next = NULL;

return top;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void Push(LinkStack top, int input)

Author :Ellis

Date :20180501

Description :push

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Push(LinkStack top, int input)

{

LinkStack node = (LinkStack)malloc(sizeof(NodeStack));

node->date = input;

node->next = top->next;

top->next = node;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void Pop(LinkStack top)

Author :Ellis

Date :20180501

Description :pop

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Pop(LinkStack top)

{

if (IsEmpty(top))

{

return;

//return 0;

}

else

{

LinkStack temp = top->next;

top->next = temp->next;

//int x = temp->date;

free(temp);

//return x;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:char GetTop(LinkStack top)

Author :Ellis

Date :20180501

Description :取栈顶

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetTop(LinkStack top)

{

if (IsEmpty(top))

{

return false;

}

else

{

return top->next->date;

}

//return IsEmpty(top) ? false : top->next->num\_operator;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:int loactevex(vexNode G[], char u)

Author :Ellis

Date :20180501

Description :

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int loactevex(vexNode G[], char u)

{

for (int i = 0; i < 100; i++)

{

if (

//(&G[i])->date == u

G[i].date == u

)

{

return i;

}

}

printf("Error!\n");

return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void createList(vexNode G[])

Author :Ellis

Date :20180501

Description :

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void createList(vexNode G[])

{

arcNode \*p;

vtype ch, u, v;

n = 0; //全局变量，计数，计顶点数

int i = 0;

int j;

//----------------------------顶点-------------------------------//

printf("Please enter your vertexs.\n");

printf("# to end.\n");

while (1)

{

char d;

printf("Enter '1' to use keyboard,2 to use txt.\n");

d = getchar();

getchar();

if (d == '1')

{

ch = getchar();

getchar();

while (ch != '#')

{

n++; //计顶点数

G[i].date = ch;

G[i].fin = NULL; //初始化

G[i].fout = NULL; //初始化

i++;

ch = getchar();

getchar();

}

break;

}

else if (d == '2')

{

FILE \*file;

char name\_of\_txt[20];

printf("Please enter the name of the txt...\n");

scanf("%s", name\_of\_txt);

file = fopen(name\_of\_txt, "r");

if (!file)

{

printf("Could not open this txt!\n");

system("pause");

}

else

{

printf("Open the txt successfully!\n");

}

int k = 0;

while (

(ch = fgetc(file)) != EOF //逐个字符去读取

)

{

if (ch == ' ')

{

continue;

}

else if (ch != '#')

{

n++; //计顶点数

G[i].date = ch;

G[i].fin = NULL; //初始化

G[i].fout = NULL; //初始化

i++;

}

}

getchar();

break;

}

else

{

printf("Error! try again!\n");

}

}

//---------------------------------------------------------------//

//----------------------------弧---------------------------------//

printf("Please enter yout arcs.\n");

printf("Example: a,b");

printf("# to end.\n");

while (1)

{

char d;

printf("Enter '1' to use keyboard,2 to use txt.\n");

d = getchar();

getchar();

if (d == '1')

{

scanf("%c,%c", &u, &v); //顶点u，v输入

getchar();

while (u != '#')

{

i = loactevex(G, u);

j = loactevex(G, v);

p = (arcNode\*)malloc(sizeof(arcNode)); //申请弧结点

p->tail = i;

p->head = j;

p->hlink = G[j].fin;

G[j].fin = p;

p->tlink = G[i].fout;

G[i].fout = p;

scanf("%c,%c", &u, &v); //顶点u，v输入

getchar();

}

break;

}

else if (d == '2')

{

FILE \*file;

char name\_of\_txt[20];

printf("Please enter the name of the txt...\n");

scanf("%s", name\_of\_txt);

file = fopen(name\_of\_txt, "r");

if (!file)

{

printf("Could not open this txt!\n");

system("pause");

}

else

{

printf("Open the txt successfully!\n");

}

int k = 0;

int flag = 0;

while (

(ch = fgetc(file)) != EOF //逐个字符去读取

)

{

if (ch == ' ')

{

flag = 0;

continue;

}

else if (ch != '#')

{

if (flag == 0)

{

u = ch;

flag = 1;

}

else if (flag == 1)

{

flag = 2;

}

else if (flag == 2)

{

v = ch;

i = loactevex(G, u);

j = loactevex(G, v);

p = (arcNode\*)malloc(sizeof(arcNode)); //申请弧结点

p->tail = i;

p->head = j;

p->hlink = G[j].fin;

G[j].fin = p;

p->tlink = G[i].fout;

G[i].fout = p;

}

}

}

break;

}

else

{

printf("Error! try again!\n");

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void Creatid(vexNode G[], int id[])

Author :Ellis

Date :20180501

Description :求入度

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Creatid(vexNode G[], int id[]) //求入度

{

int count;

arcNode \*p;

for (int i = 0; i < n; i++) //用逆邻接表求

{

count = 0;

p = G[i].fin;

while (p)

{

count++;

p = p->hlink;

}

id[i] = count;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void TopSort(vexNode G[])

Author :Ellis

Date :20180501

Description :

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TopSort(vexNode G[]) //十字链表，拓扑排序

{

int j;

int k;

int count;

int id[100];

LinkStack mid = CreateStack();

arcNode \*p;

Creatid(G, id);

for (int i = 0; i < n; i++)

{

if (id[i] == 0)

{

Push(mid, i);

}

}

count = 0;

printf("\n");

while (mid->next)

{

j = GetTop(mid);

Pop(mid);

//printf("%c \t", G[j].date);

count++;

p = G[j].fout;

while (p)

{

k = p->head;

id[k]--;

if (id[k] == 0)

{

Push(mid, k);

}

p = p->tlink;

}

}

printf("\n");

if (count == n)

{

printf("This graph has no cycles.\n");

}

else

{

printf("This graph has cycles.\n");

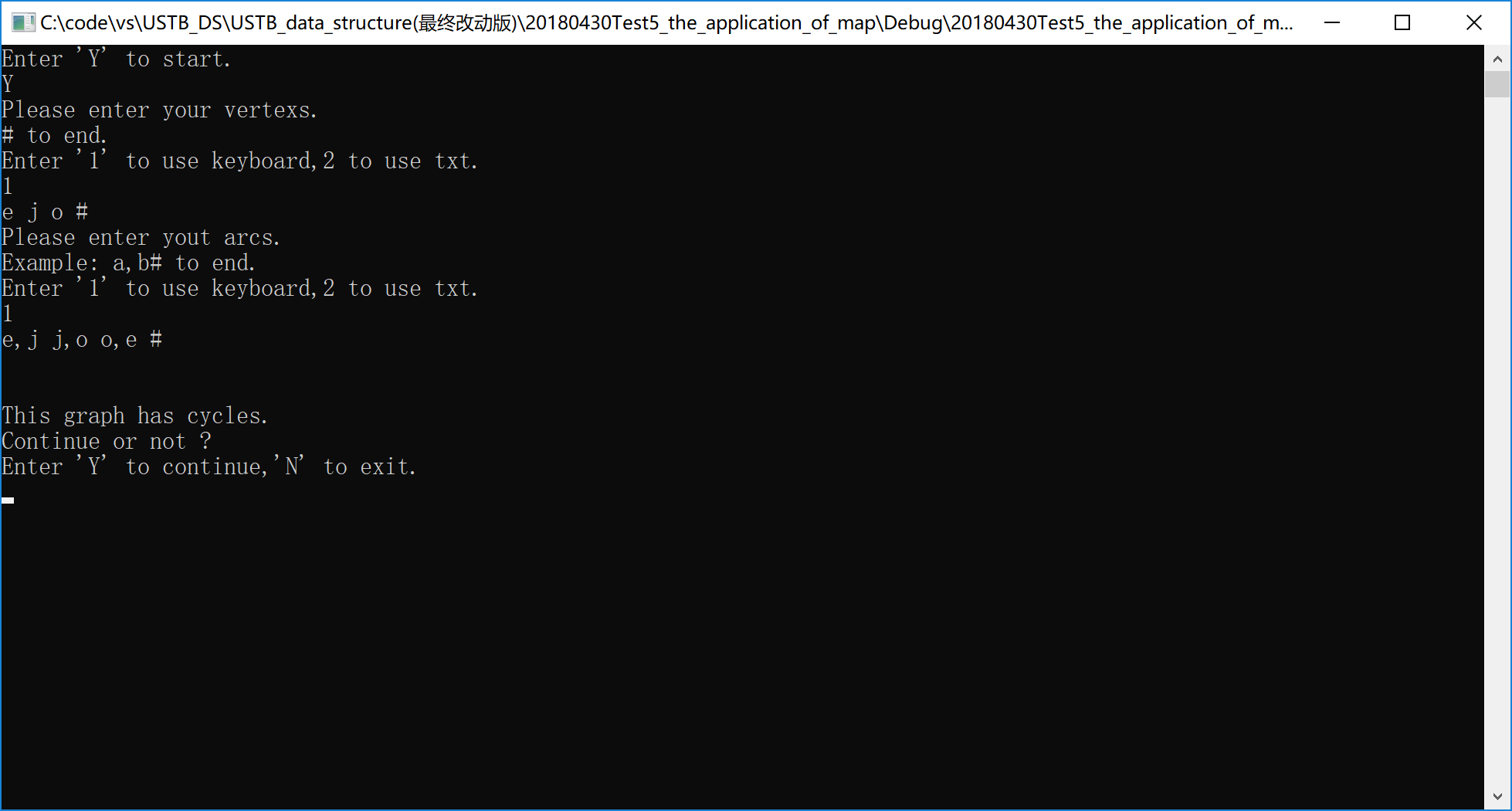
}

}

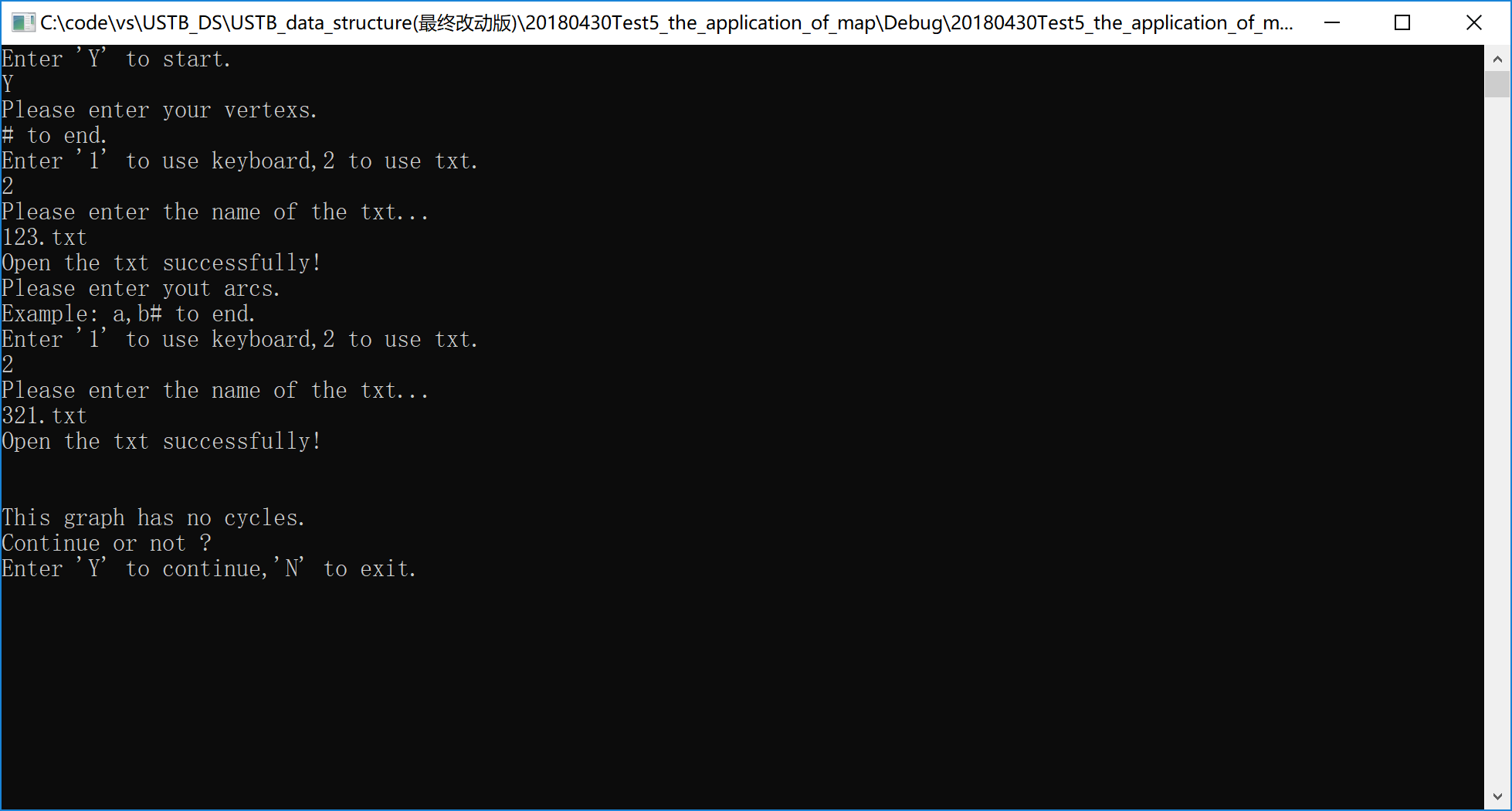
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

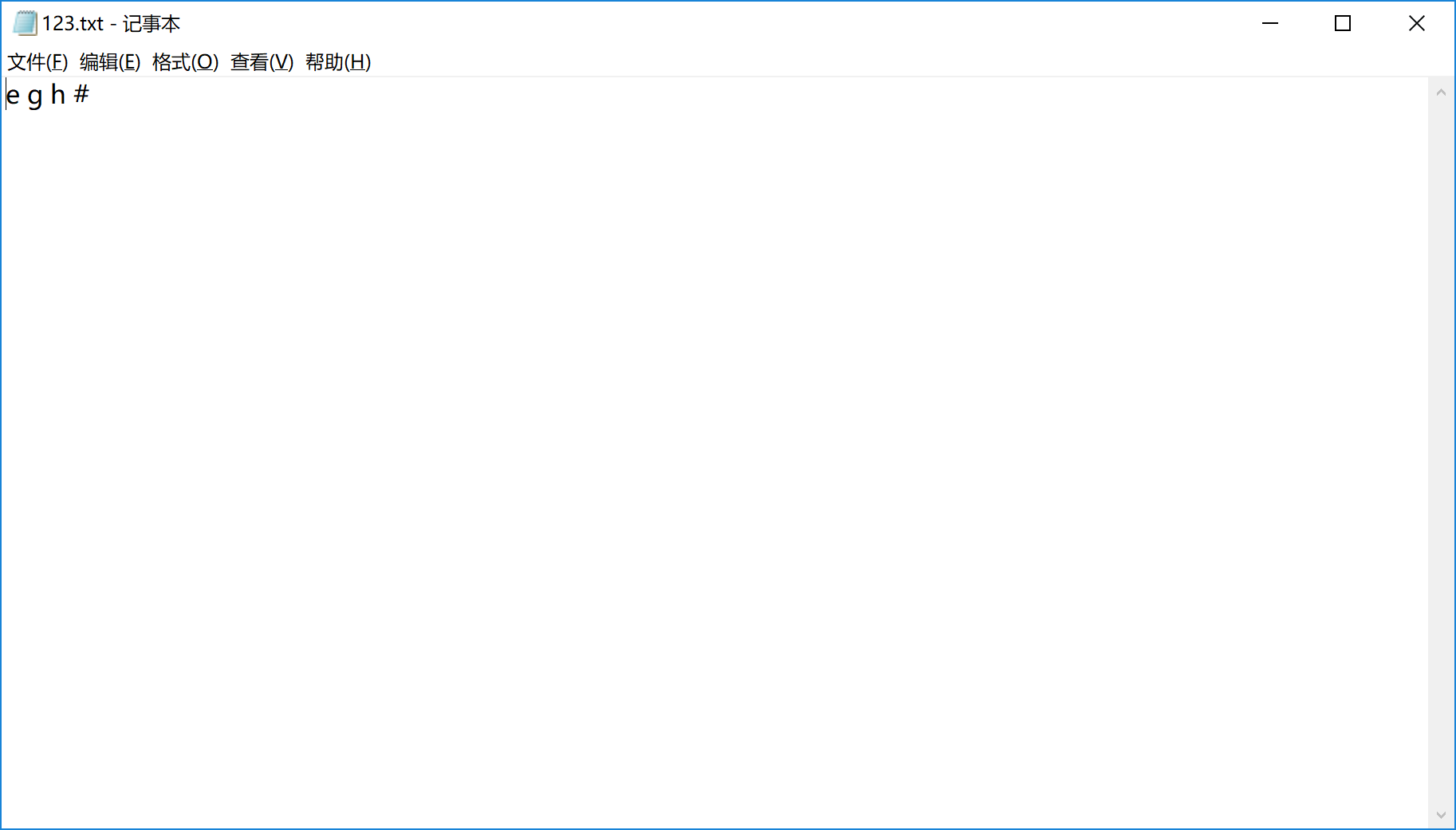
**测试：**

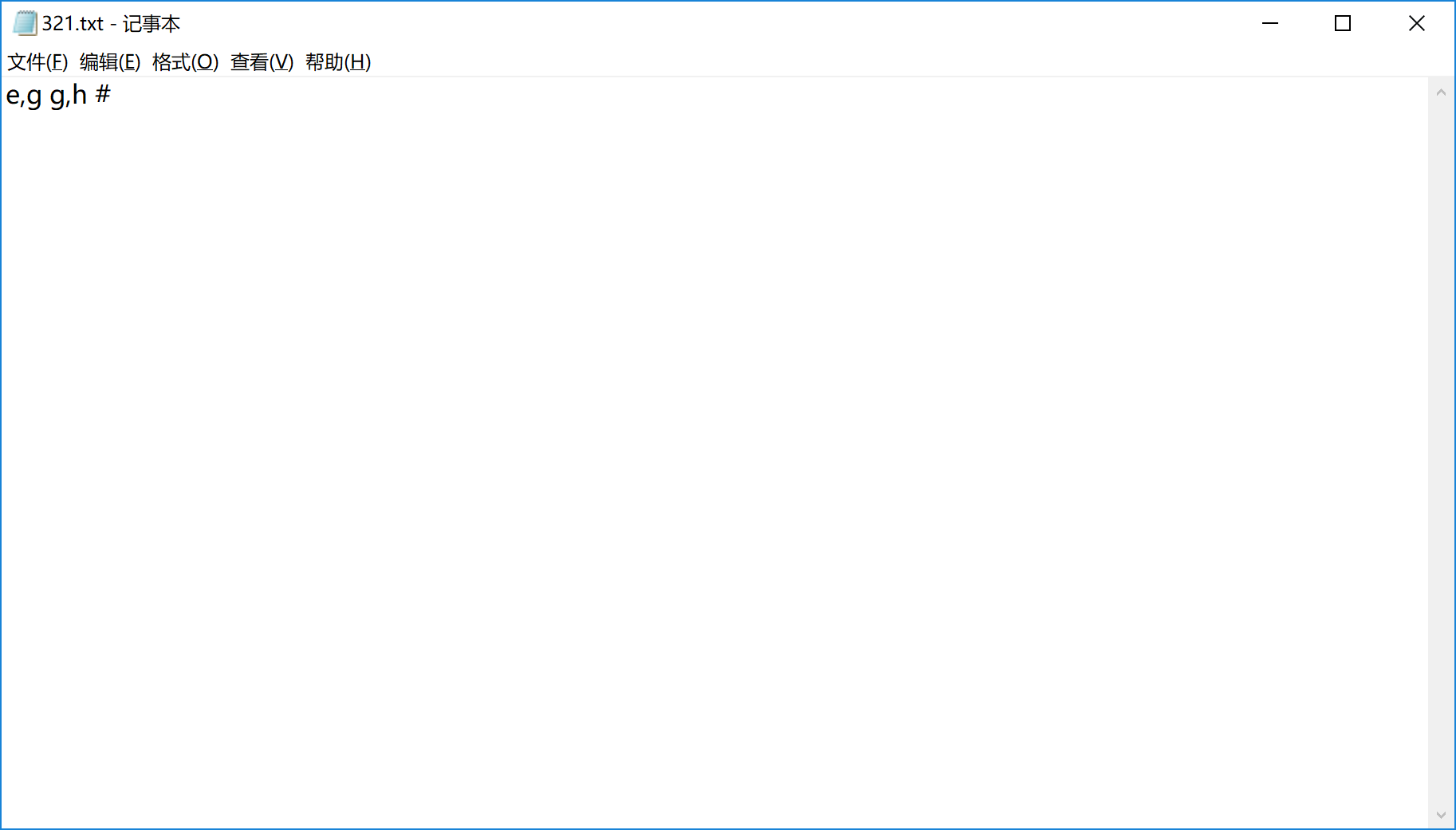
测试键盘输入：



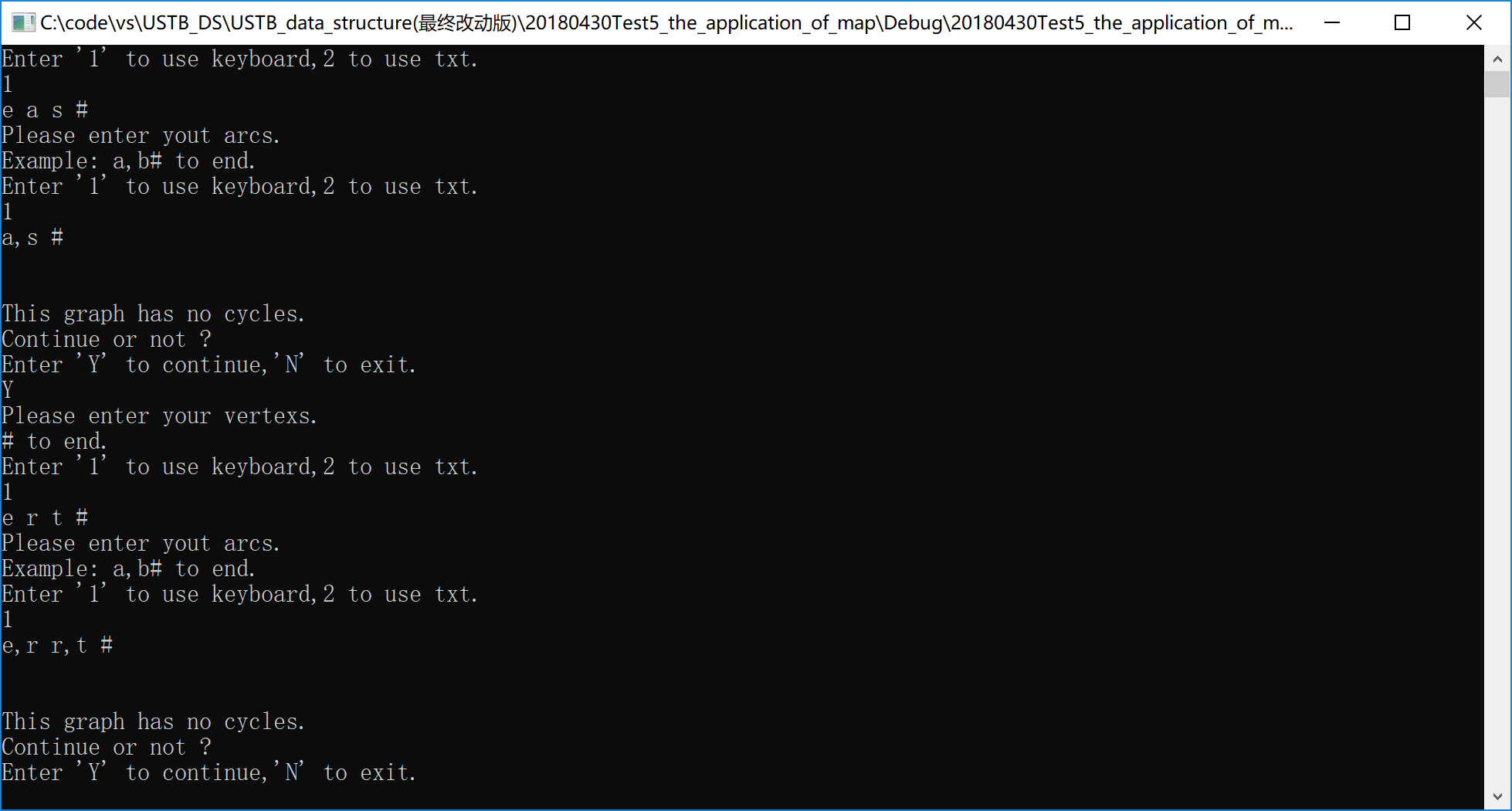
测试txt输入：



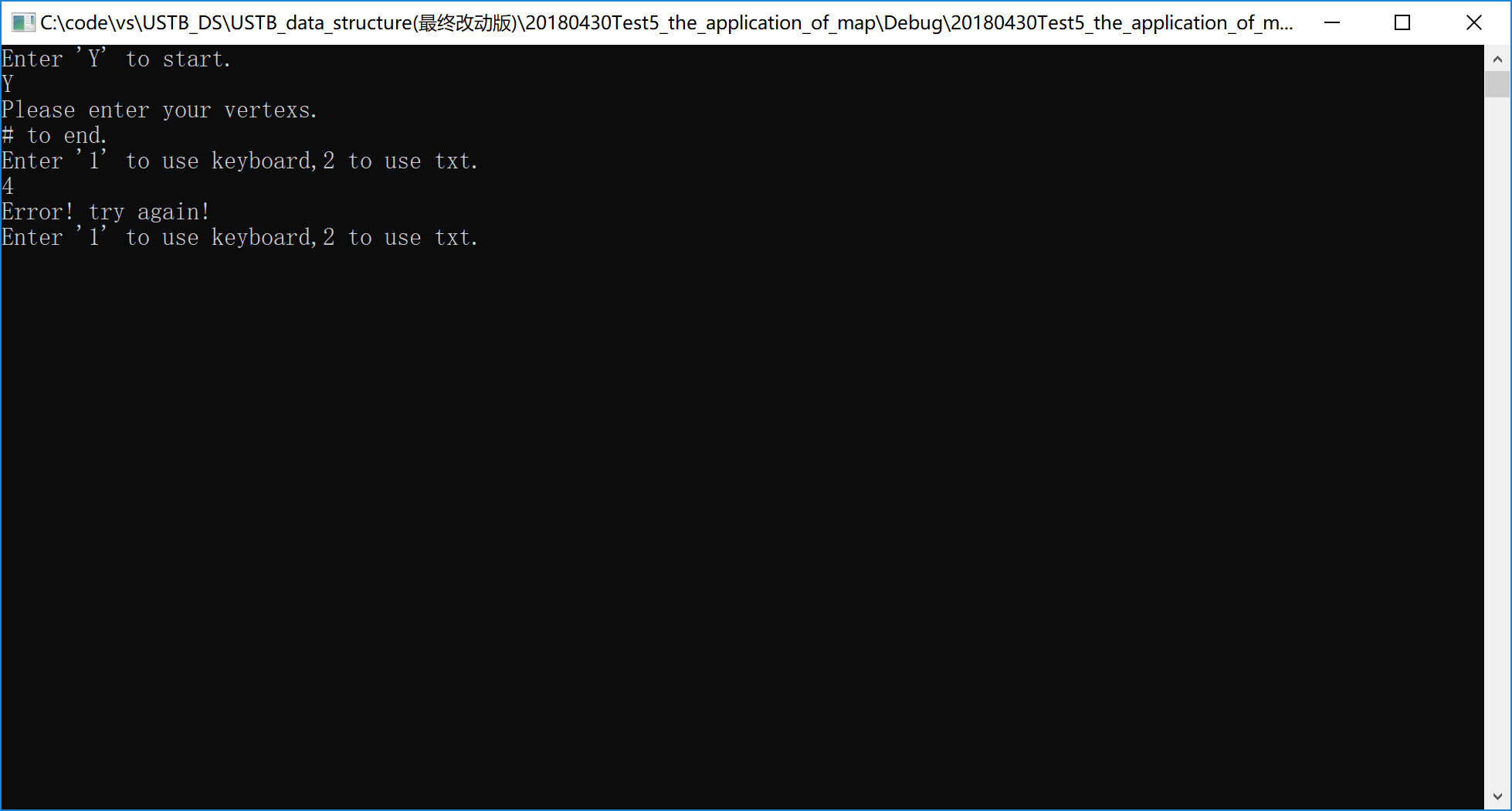




测试重试：



测试错误输入：



**算法的有效性分析：**

时间空间复杂度：时间复杂度为O(n)，空间复杂度合适

有效性：程序易懂，算法正确，用邻接表和逆邻接表构造十字链表，并运用链式堆栈，使用的数据结构合理，txt和键盘输入都实现了。

不足：程序不够精简