**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 5月21 日

**实验名称：《数据结构》实验3队列的应用**

**实验目的：理解队列先进先出的特点，熟悉入队、出队等队列基本操作，练习队列的存储结构及应用。**

**实验内容：**

将从键盘输入的一系列字符存储到链式队列中，当输入的字符为’0’时，执行出队操作并将出队元素打印到屏幕上；当输入的字符为’@’时，队列中剩余所有元素依次出队并打印到屏幕上；当输入其他字符时，字符入队。

要求：可根据用户需求多次重复该过程；自行补充所需的队列基本操作。

**问题分析与算法思路：**

本程序运用的数据结构是链式的队列，用到的算法为普通的队列操作，循环操作，起到进队出队等基本操作。

其中对‘0’和‘@’的操作为出队一个元素并打印和出队所有元素并打印。

本程序较为基础，时间复杂度为O(n),空间复杂度合适。

**算法描述：**

typedef struct node

{

char str;

struct node \*next;

}Qnode,\*Qlink;

typedef struct

{

Qnode \*front, \*rear;

}linkqueue;

此为链式队列的结构体定义。

void CreateQueue(linkqueue \*q)

{

申请一个头节点，q指向它(front,rear)

}

此为创建队列的函数，注意指针的指向。

void EnQueue(linkqueue \*q, char \*str\_new)

{

申请一个节点，插到队列头

}

此为进队的函数，

void DeQueue(linkqueue \*q)

{

Qlink mid;

if (队列空)

ERROR

else

{

删除尾部节点

}

}

此为出队的函数，提前判断了队列是否为空

**程序实现：**

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

\* Copyright(c) :Ellis

\* All rights reserved.

\*

\* 文件名称:USTB2018\_data\_structure\_test3

\* 简要描述:链式队列的基本操作

\* 将从键盘输入的一系列字符存储到链式队列中，当输入的字符为’0’时，执行出队操作并将出队元素打印

\* 到屏幕上；当输入的字符为’@’时，队列中剩余所有元素依次出队并打印到屏幕上；当输入其他字符时，字符入队

\*

\* 创建日期:20180418\_10:32

\* 作者:Ellis

\* 说明:完成任务，此任务相对简单

\*

\* 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: 20180418

\* 作者 : Ellis

\* 说明 : 无

\*

\* 修改日期:

\* 作者 :

\* 说明 :

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

#define \_CRT\_SECURE\_NO\_WARNINGS //解决不安全问题

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

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

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

\* 名称 : 结构体定义区

\* 创建日期: 20180418

\* 作者 : Ellis

\* 说明 : 链式的队列

\*

\* 修改日期:

\* 作者 :

\* 说明 :

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

typedef struct node

{

char str;

struct node \*next;

}Qnode,\*Qlink;

typedef struct

{

Qnode \*front, \*rear;

}linkqueue;

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

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

\* 名称 : 函数定义区

\* 创建日期: 20180418

\* 作者 : Ellis

\* 说明 : 定义各种函数

\*

\* 修改日期:

\* 作者 :

\* 说明 :

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

void CreateQueue(linkqueue \*q); //创建队列

bool isEmpty(linkqueue \*q); //队列判空

void EnQueue(linkqueue \*q, char \*str\_new); //入队

void DeQueue(linkqueue \*q); //出队

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

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

Function Name:void main()

Author :Ellis

Date :20180418

Description :主函数

Inputs :

Outputs :

Notes :

Revision :

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

void main()

{

//Qlink q = (Qlink)malloc(sizeof(Qnode));

linkqueue \*q = (linkqueue\*)malloc(sizeof(linkqueue));

//linkqueue \*q = ;

//q = (linkqueue \*)malloc(sizeof(linkqueue));

//q->front = NULL;

//q->rear = NULL;

CreateQueue(q);

printf("Warning: Only character is supported!\n");

while (1)

{

while (1)

{

printf("Please enter your number:\t");

char mid[100];

gets(mid);

printf("%s\n", mid);

if (strcmp(mid,"0") == 0) //0时候出一个队列

{

if (!isEmpty(q))

{

DeQueue(q);

}

else

{

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

}

}

else if (strcmp(mid, "@") == 0) //@时候出完所有队列

{

if (!isEmpty(q))

{

while (1)

{

if (

//!q->front->next

isEmpty(q))

{

break;

}

DeQueue(q);

}

break;

}

else

{

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

}

}

//否则进队

else

{

EnQueue(q, mid);

}

//Qlink new\_q = (Qlink)malloc(sizeof(Qnode));

}

printf("Continue or not?\n");

printf("Print arbitrary character to continue. \t Print 'No' to exit.\n");

char recall[5];

gets(recall);

if (strcmp(recall, "No") == 0 || strcmp(recall, "no") == 0)

{

//exit(1);

break;

}

else

{

;

}

}

system("pause");

}

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

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

Function Name:void CreateQueue(linkqueue \*q)

Author :Ellis

Date :20180418

Description :

Inputs :

Outputs :

Notes :

Revision :

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

void CreateQueue(linkqueue \*q)

{

q->front = (Qlink)malloc(sizeof(Qnode));

q->front->next = NULL;

q->rear = q->front;

}

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

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

Function Name:bool isEmpty(linkqueue \*q)

Author :Ellis

Date :20180418

Description :

Inputs :

Outputs :

Notes :

Revision :

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

bool isEmpty(linkqueue \*q)

{

if (q->front == q->rear)

{

return true;

}

else

{

return false;

}

}

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

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

Function Name:void EnQueue(linkqueue \*q, char \*str\_new)

Author :Ellis

Date :20180418

Description :

Inputs :

Outputs :

Notes :

Revision :

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

void EnQueue(linkqueue \*q, char \*str\_new)

{

Qlink new\_q = (Qlink)malloc(sizeof(Qnode));

new\_q->str = \*str\_new;

new\_q->next = NULL;

q->rear->next = new\_q;

q->rear = new\_q;

}

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

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

Function Name:void DeQueue(linkqueue \*q)

Author :Ellis

Date :20180418

Description :

Inputs :

Outputs :

Notes :

Revision :

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

void DeQueue(linkqueue \*q)

{

Qlink mid;

if (isEmpty(q))

{

return;

}

else

{

mid = q->front;

q->front = mid->next;

printf("%c has out.\n", mid->next->str);

free(mid);

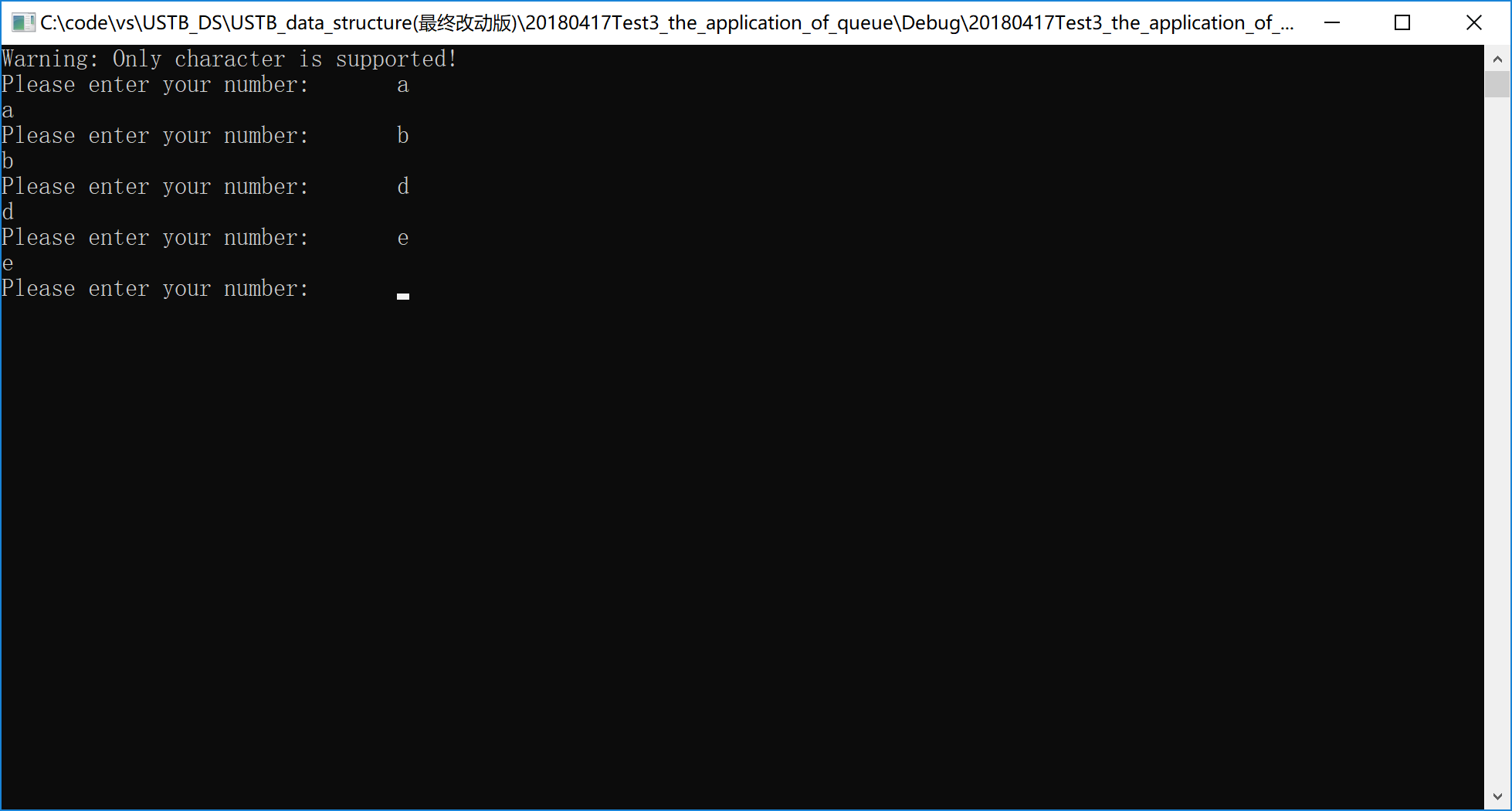
}

}

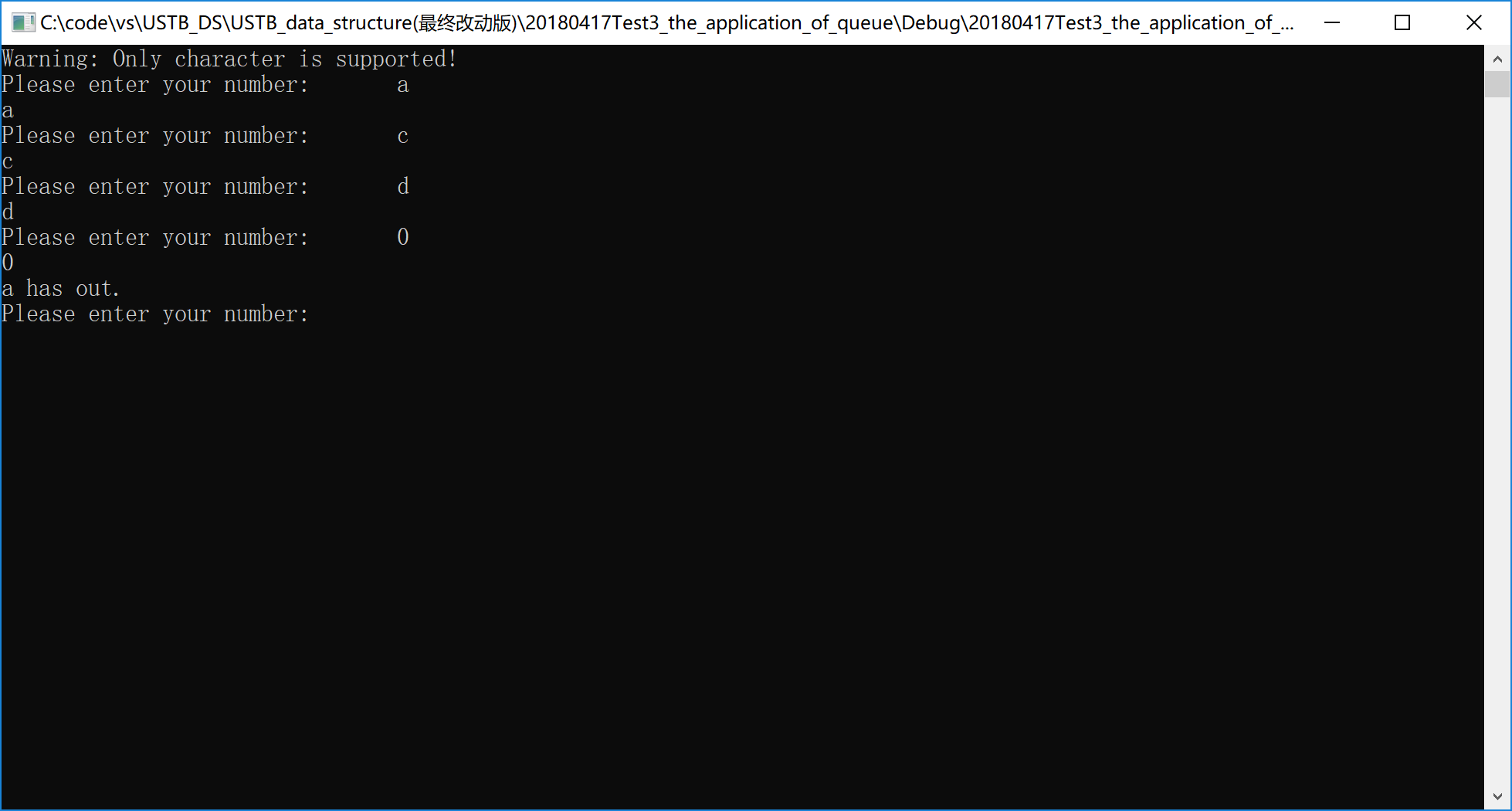
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**测试：**

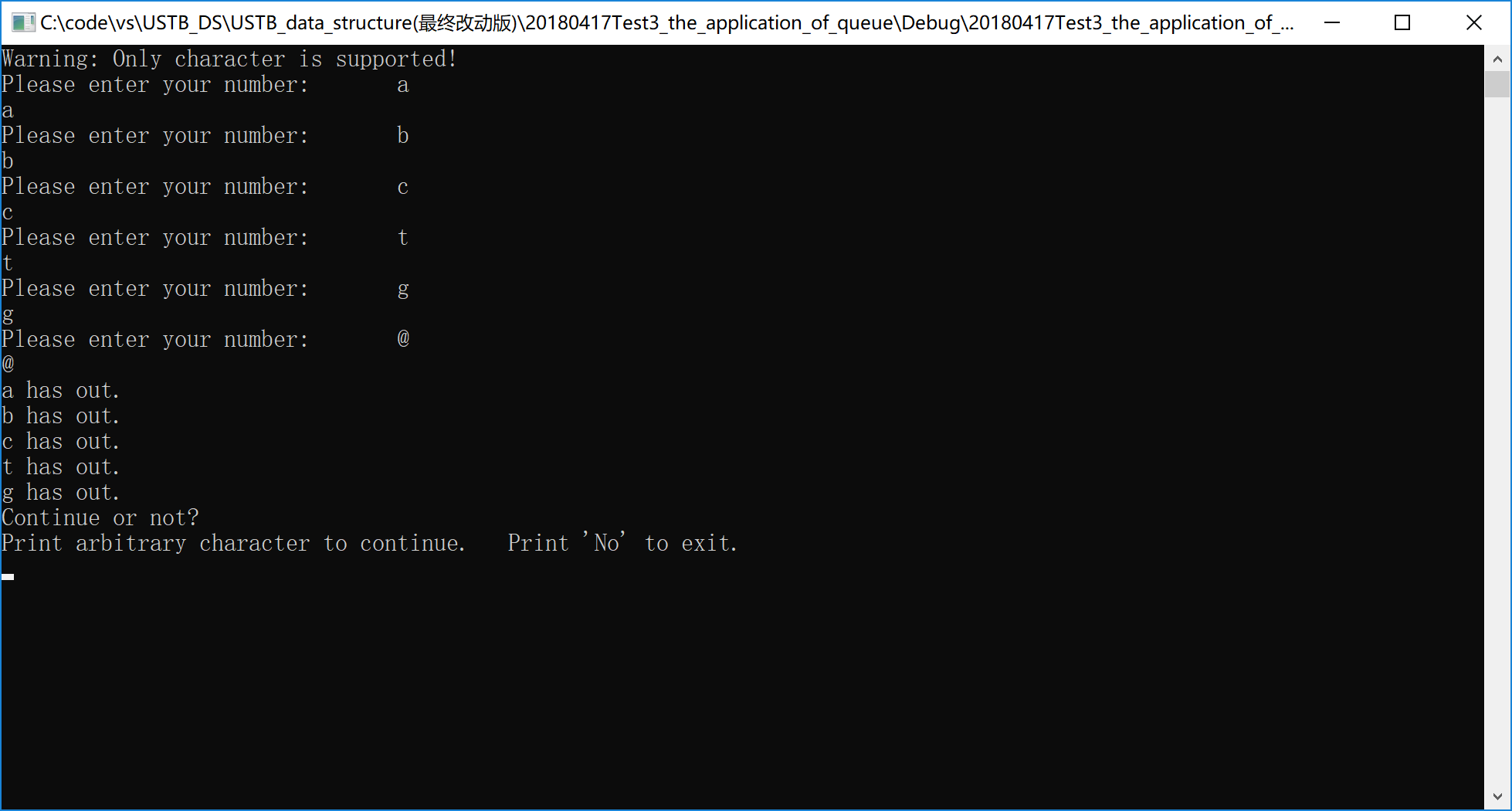
测试入队：



测试’0’出队



测试’@’清空队：



测试重试：



**算法的有效性分析：**

时间复杂度和空间复杂度：算法时间复杂度为O(n)，空间复杂度合适。

有效性：算法正确且简单，测试完整且成功

不足：代码不够简洁