栈和队列 第 14 题

问题描述:

用带头结点的循环链表来表示队列，实现入队、出队、置空队操作。

1：函数结构设计

（1）入队

函数名	EnQueue
函数正常输入	Queue 型的队列指针， ElemType 型(宏定义为 char 型)的要入队的数据
函数正常输出	成功返回 TRUE=1，失败返回 FALSE=-1

（2）出队

函数名	DeQueue
函数正常输入	Queue 型的队列指针
函数正常输出	ElemType 型(宏定义为 char 型)的出队数据 成功返回 TRUE=1，失败返回 FALSE=-1

（3）置空

函数名	QueueClear
函数正常输入	Queue 型的队列指针
函数正常输出	成功返回 TRUE=1，失败返回 FALSE=-1

2：测试样例设计

（1）入队

	输入	预测结果
一般正常情况	队列指针，要入队的数据	正常入队，返回 TRUE=1
异常情况	队列指针为空	返回 FALSE=-1

（2）出队

	输入	预测结果
一般正常情况	队列指针	正常出队，返回要出队的数据，以及 TRUE=1
异常情况	队列指针为空， 队列没有初始化	返回 FALSE=-1
边界情况	队列为空	返回 FALSE=-1

（3）置空队

	输入	预测结果
一般正常情况	队列指针	正常置空，返回 TRUE=1
异常情况	队列指针为空	返回 FALSE=-1

3：伪代码描述

（1）入队

如果输入的队列指针为空，返回 FALSE=-1
产生要入队的新节点
入队的新节点为队尾元素，其 next 指针指向头结点
原先的队尾元素 next 指针指向新节点，队列的尾指针指向新节点
返回 TRUE=1

（2）出队

如果输入的队列指针为空，队列没有初始化，或队列为空，返回 FALSE=-1
队头节点数据通过指针赋值给 ElemType *dat
链表头结点 next 指针指向新的队头元素
如果只有一个元素，出队后队列为空，尾指针指向链表头结点
删除出队节点，回收内存
返回 TRUE=1

（3）置空队

如果输入的队列指针为空，返回 FALSE=-1
循环调用出队函数 DeQueue，直到返回 FALSE，则队列为空
返回 TRUE=1

4：程序描述

```
// 栈和队列 14 题

#include "stdafx.h"
#include <assert.h>
#include <iostream>
using namespace std;

#define TRUE            1
#define FALSE           -1

typedef char ElemType;
```

```
typedef struct LinkList
{
	ElemType dat;
	LinkList *next;
} LinkList;

typedef struct Queue
{
	LinkList *list;
	LinkList *rear;
} Queue;

// 创建空队列
Queue *QueueCreate(void)
{
	Queue *q = new Queue;
	assert(q != NULL);
	LinkList *head = new LinkList;
	assert(head != NULL);
	head->dat = '#';
	head->next = head;
	q->list = head;
	q->rear = head;
	return q;
}

// 入队
int EnQueue(Queue *q, ElemType dat)
{
	if(q == NULL) {
		return FALSE;
	}
	LinkList *New = new LinkList;
	assert(New != NULL);
	New->dat = dat;
	New->next = q->list;		// 入队的新节点为队尾元素，其 next 指针指向头结点
	q->rear->next = New;		// 原先的队尾元素 next 指针指向新节点
	q->rear = New;			// 队列的尾指针指向新节点
	return TRUE;
}
```

```
// 出队
int DeQueue(Queue *q, ElemType *dat)
{
	if((q == NULL) || (q->list == NULL) || (q->list == q->rear)) {
		return FALSE;			// 队列指针为空 / 队列链表为空 / 队列为空(链表中只有头结点)，则返回 FALSE
	}
	LinkList *lst = q->list->next;	// 队头元素
	*dat = lst->dat;
	q->list->next = lst->next;	// 链表头结点 next 指针指向新的队头元素
	if(q->rear == lst) {			// 队头和队尾相同，说明只有一个元素
		q->rear = q->list;		// 出队后，队列为空，尾指针指向链表头结点
	}
	delete lst;				// 回收内存
	return TRUE;
}

// 清空队列
int QueueClear(Queue *q)
{
	if(q == NULL) {
		return FALSE;
	}
	ElemType dat;
	while(DeQueue(q, &dat) == TRUE);
	return TRUE;
}

int main(void)
{
	Queue *q = QueueCreate();

	// 入队函数测试
	cout << "入队函数测试" << endl;
	cout << EnQueue(NULL, ' ') << endl;	// 异常情况

	for(int i = 0; i < 3; ++i) {		// 正常情况：'0','1','2'依次入队
		cout << EnQueue(q, i + '0') << endl;
	}

	// 出队函数测试
	ElemType e;
	cout << "出队函数测试" << endl;
	cout << DeQueue(NULL, &e) << endl;	// 异常情况
```

```
    for(int i = 0; i < 4; ++i) {
        if(DeQueue(q, &e) == TRUE) {
            cout << e << " ";              // 正常情况：'0','1','2'依次出队，输出出队元素
        } else {
            cout << "NULL ";               // 边界情况：队列为空，输出"NULL"
        }
    }
    cout << endl;

    // 置空队函数测试
    cout << "置空队函数测试" << endl;
    cout << QueueClear(NULL) << endl;      // 异常情况

    for(int i = 0; i < 3; ++i) {
        EnQueue(q, i + '0');
    }
    cout << QueueClear(q) << endl;         // 正常情况：置空队
    cout << DeQueue(q, &e) << endl;        // 此时队列为空，出队返回 FALSE=-1

    system("pause");
    return 0;
}
```

5：结果展示

```
入队函数测试
-1                  // 异常情况
1                   // 正常情况：'0','1','2'依次入队

1
1
出队函数测试
-1                  // 异常情况
0 1 2 NULL          // 正常情况：'0','1','2'依次入队，边界情况：队列为空，输出 NULL
置空队函数测试
-1                  // 异常情况
1                   // 正常情况：置空队
-1                  // 此时队列为空，出队返回 FALSE=-1
```