

《数据结构课程实践》

**实验报告**

**实验名称：队列**

**姓 名：杨天诏**

**学 号：20081831**

**专 业：通信工程**

**实验时间：10月31日**

**杭州电子科技大学**

**通信工程学院**

**（实验报告要求：填写三四五一共三部分内容，实验报告文件命名方式：*数据结构课程实践2021\_实验报告5-姓名-学号*，完成后上传电子版到校园网的网络教学平台，无需打印。）**

1. **实验目的**

掌握环形队列和链队的基本算法设计。

1. **实验内容**

1.（实验指导书63页实验3.3）实现环形队列基本运算的算法，其中MaxSize改为5，连续入队abcde共5个元素再进行出队操作，并输出显示入队出队过程；

2.（实验指导书66页实验3.4）实现链队基本运算的算法；

3. （附加题）利用两个顺序栈（MaxSize设为5）S1和S2，模拟一个队列，并在main()函数对队列连续执行以下操作：入队a b c, 出队a, 入队d e f g, 出队b c d

1. **算法设计**
2. 针对实验内容1给定的5个元素的入队出队操作，如果环形队列发生溢出，如何修改算法增加队列容量使得刚好不会溢出（假定MaxSize不变）？

修改后代码如下：

#pragma once

//顺序队列（非环形队列）基本运算算法

#include <stdio.h>

#include <malloc.h>

#define MaxSize 5

typedef char ElemType;

typedef struct

{

ElemType data[MaxSize];

int front, rear; //队头和队尾指针

int flag;

} SqQueue;

void InitQueue(SqQueue\*& q)

{

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

q->front = q->rear = 0;

q->flag = 0;

}

void DestroyQueue(SqQueue\*& q) //销毁队列

{

free(q);

}

bool QueueEmpty(SqQueue\* q) //判断队列是否为空

{

return((q->front == q->rear) && (q->data[q->flag] == NULL));

}

bool enQueue(SqQueue\*& q, ElemType e) //进队

{

if ((q->rear+1) == q->flag) //队满上溢出

return false; //返回假

q->rear= (q->rear + 1) % MaxSize; //队尾增1

q->data[q->rear] = e; //rear位置插入元素e

return true; //返回真

}

bool deQueue(SqQueue\*& q, ElemType& e) //出队

{

if ((q->front == q->rear)&&(q->data[q->flag]==NULL)) //队空下溢出

return false;

q->front= (q->front + 1) % MaxSize;

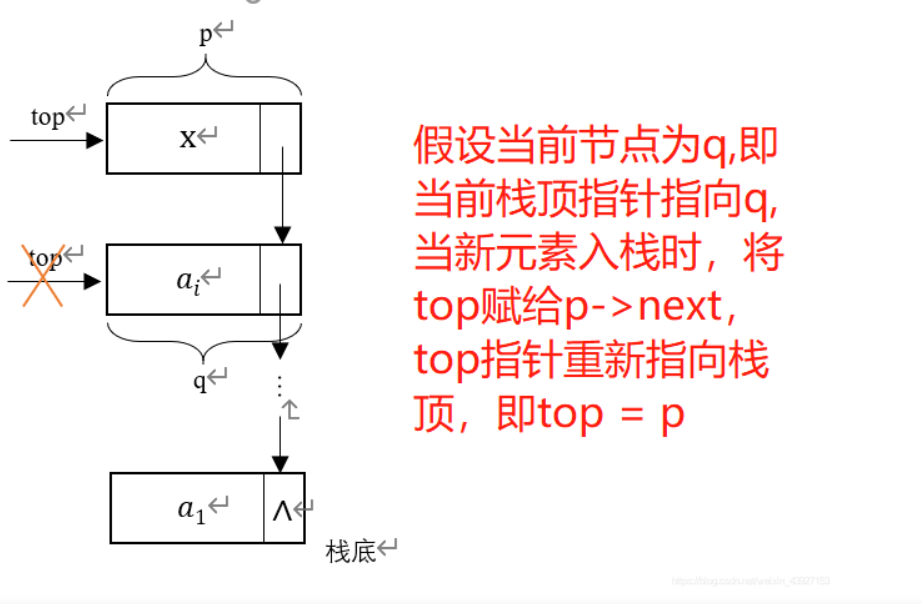
e = q->data[q->front];

q->data[q->front] = NULL;

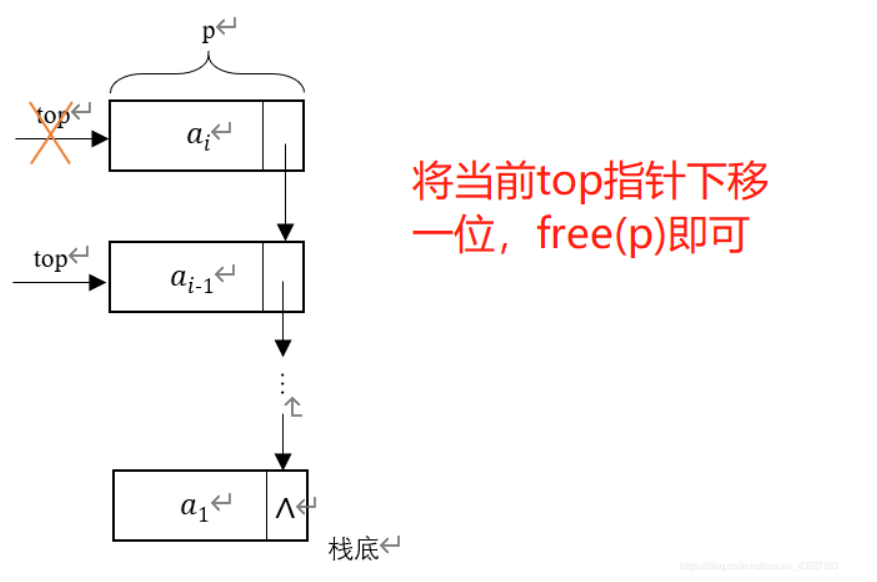
return true;

}

1. 对于实验中链队的进栈和出栈操作，分别画图表示其过程步骤。

进栈：

出栈：



1. 对于附加题：(1)简要说明算法思想；(2)给出完整的算法代码；(3)你设计的模拟队列的容量是多少？有没有算法可以让容量达到10？（简要说明即可）
2. **enQueue:当执行入队操作时,先判断s1是否满,如果s1满了,需要将s1元素全部移动到s2中,如果此时s2中已经有元素存在,此时不能移动,因为如果移动到s2,则队列出队顺序会混乱! 此时抛出队满即可,如果s2为空,将元素移动到s2,在入栈s1,即可完成入队。deQueue:出队时,判断s2是否为空,如果为空先将s1中元素移动到s2中,在将s2中元素出栈即可,若不为空直接将s2栈顶元素出栈**

bool enQueue(SqStack\*& S1, SqStack\*& S2, ElemType e)

{

if (!StackFULL(S1))

{

Push(S1, e);

return true;

}

if (StackFULL(S1) && StackFULL(S2))

{

printf("队列满，不可入队\n");

return false;

}

char x;

if (StackFULL(S1) && StackEmpty(S2))

{

while (!StackEmpty(S1)) //判断S1栈是否为空,不空则将S2栈元素全都依次入S1栈,最后继续入新元素

{

Pop(S1, x);

Push(S2, x);

}

Push(S1, e);

return true;

}

}

//出队算法

void deQueue(SqStack\*& S1, SqStack\*& S2, ElemType& x)

{

if (!StackEmpty(S2))

{

Pop(S2, x);

}

else if (!StackEmpty(S1))

{

while (!StackEmpty(S1))

{

Pop(S1, x);

Push(S2, x);

}

Pop(S2, x);

char e;

while (!StackEmpty(S2))

{

Pop(S2, e);

Push(S1, e);

}

}

else

{

printf("队列为空，不可出队\n");

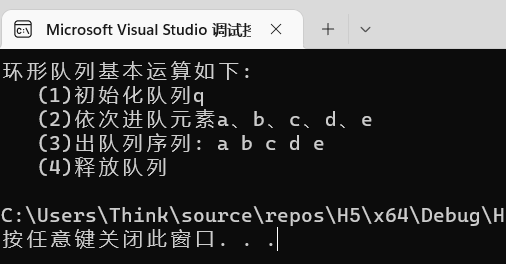
}

}

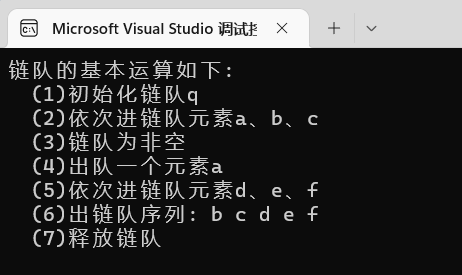
（3）模拟队列容量为5；将两个顺序栈的容量为10即可

1. **运行结果与分析（给出运行结果截图，如有必要，请加以简要说明）**

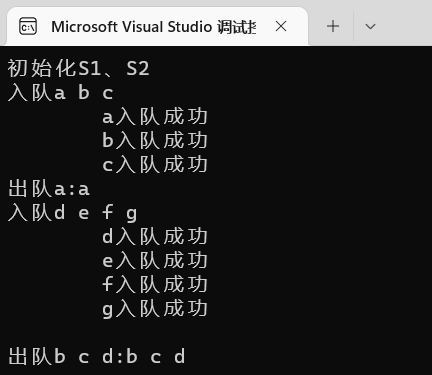
**（1）**

****

**（2）**

****

**（3）**

****

1. **实验小结（记录实验过程中遇到的主要问题和心得）**

**通过本次实验，对环形队列和链队的算法设计有了基本的了解。**