

《数据结构课程实践》

**实验报告**

**实验名称：栈与队列的应用**

**姓 名：杨天诏**

**学 号：20081831**

**专 业：通信工程**

**实验时间：11月14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2022\_实验报告7-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**

掌握栈与环形队列应用的的算法设计。

1. **实验内容**

1.（实验指导书75页实验3.7）学习对栈元素排序的算法；

2. 给定一个的长度为MaxSize的环形队列，其中包含有m（m<MaxSize）个字符。请设计一个算法对队列中的元素进行排序，即从队头元素开始，元素的大小依次递增，最大的元素位于队尾。

**\*要求：不使用额外的数组、队列或栈结构，可以使用临时变量，即算法的空间复杂度为O(1).**

**验证的字符队列：c f b f f e a b**

**提示：学习第一题的思路，将队列划分成前后两部分，无序区在前，有序区在后。每次从无序区取出一个队头元素，插入到有序区。有序区初始长度为0，每次插入一个新元素长度加一。需要设置一个计数器，动态地表示有序区长度。（可以参考教材10.2.1:直接插入排序）**

**注意：由于队列的数据封装特征，只允许队尾插入数据，队首输出数据。**

1. **算法设计**
2. 请描述实验题2的算法步骤（文字描述或者伪代码）；

将前k个元素进行排序，第k+1个元素出队作为下一次排序的比较值，之后将未参与排序的元素排到最后面，不断循环，直到排完所有元素。

出队元素e;

while (k<队列长度-1){

k++; .

for(i=0;i<k;i++){

出队元素el;

比较el，e的大小:

若e1大于e

入队元素e

e=el;

else

enQueue(st, e1);

}

入队元素e，出队元素e

逆转队列顺序;

1. 给出实验题2的完整代码，其中环形队列基本操作的部分可以调用上次实验的代码；

3. 说明所给算法的时间和空间复杂度。

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）**

**题目一：**

#include"x.h"

void StackSort(SqStack\*& st)

{

SqStack\* tmpst; //临时栈

ElemType e, e1;

InitStack(tmpst);//初始化临时栈

while (!StackEmpty(st))// st栈不空循环

{

Pop(st, e);//出栈元素e

printf(" st:出栈%c=> ", e);

while (!StackEmpty(tmpst)) // tmpst栈不空循环

{

GetTop(tmpst, e1);//取栈顶元素e1

printf("tmpst:取栈顶元素%c", e1);

if (e1 > e)

{

printf("因%c>%c ", e1, e);

printf("tmpst:退栈%c ", e1);

Pop(tmpst, e1);

printf("s:进栈%c ", e1);

Push(st, e1);

}

else

{

printf("因%c<%c,退出循环", e1, e);

break;

}

}

Push(tmpst, e);

printf("tmpst:进栈%c\n", e);

}

while (!StackEmpty(tmpst))

{

Pop(tmpst, e);

Push(st, e);

}

DestroyStack(tmpst);//销毁临时栈

}

int main()

{

ElemType e;

SqStack \*s;

InitStack(s);

printf("(1)依次进栈元素1, 3, 4, 2\n");

Push(s, '1');

Push(s, '3');

Push(s, '4');

Push(s, '2');

printf("(2)栈s排序过程:\n");

StackSort(s);

printf("(3)栈s排序完毕\n");

printf("(4)s的出栈序列:");

while (!StackEmpty(s))

{

Pop(s, e);

printf("%c ", e);

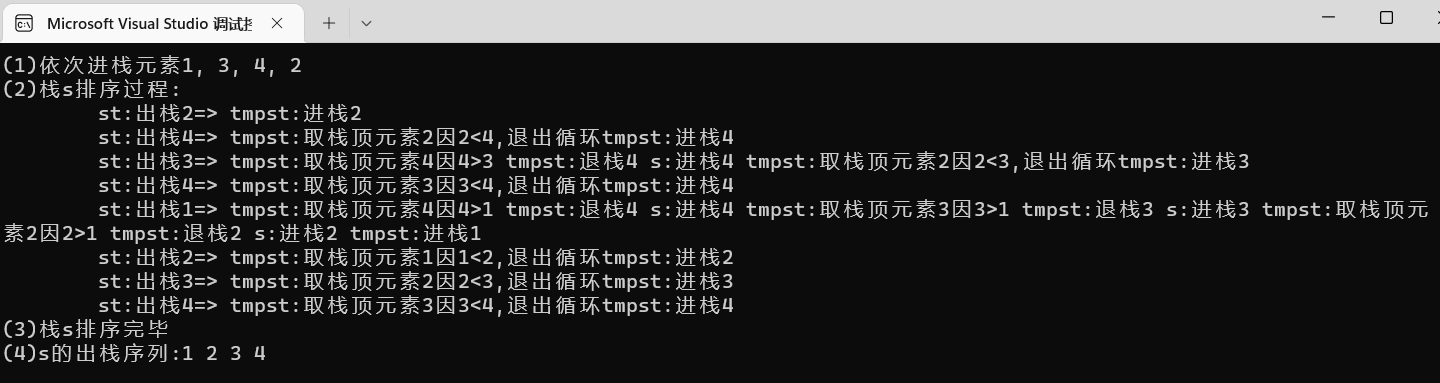
}

printf("\n");

DestroyStack(s);

return 0;

}

****

**通过辅助栈tmpst，将栈st中的元素进行排序。让st栈顶元素出栈，放入栈tmpst中,为e1，然后与st栈顶元素e对比，如果e>e1，则将e入栈tmpst，如果e1>e，则将栈tmpst栈顶元素e1放入栈st，不断比较，直到tmpst栈顶元素小于e；如此循环，将栈st的所有元素都按从小到大的顺序放入栈tmpst中，最终将tmpst元素依次存入st，即可得到元素从大到小排序的栈st，此时栈st出栈顺序便为从小到大。**

**题目二：**

#include"Y.h"

void QueueSort(SqQueue\*& st) {

int k = 0, count;

ElemType e, e1;

count = (st->rear - st->front + MaxSize) % MaxSize;

deQueue(st, e);

while (k < count - 1) {

k++;

for (int i = 0; i < k; i++) {

deQueue(st, e1);

if (e1 > e) {

enQueue(st, e);

e = e1;

}

else

enQueue(st, e1);

}

enQueue(st, e); deQueue(st, e);

for (int i = 0; i < count - k - 2; i++) {

deQueue(st, e1);

enQueue(st, e1);

}

}

enQueue(st, e);

for (int i = 0; i < count - 1; i++) {

deQueue(st, e1);

enQueue(st, e1);

}

}

int main()

{

SqQueue\* q;

ElemType e;

InitQueue(q);

if (!enQueue(q, 'c')) printf("\t提示:队满,不能进队\n");

if (!enQueue(q, 'f')) printf("\t提示:队满,不能进队\n");

if (!enQueue(q, 'b')) printf("\t提示:队满,不能进队\n");

if (!enQueue(q, 'f')) printf("\t提示:队满,不能进队\n");

if (!enQueue(q, 'f')) printf("\t提示:队满,不能进队\n");

if (!enQueue(q, 'e')) printf("\t提示:队满,不能进队\n");

if (!enQueue(q, 'a')) printf("\t提示:队满,不能进队\n");

if (!enQueue(q, 'b')) printf("\t提示:队满,不能进队\n");

QueueSort(q);

while (!QueueEmpty(q))

{

deQueue(q, e);

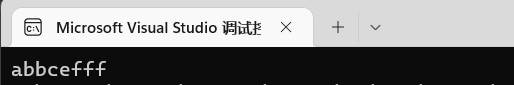
printf("%c", e);

}

DestroyQueue(q);

return 0;

}



时间复杂度：O(n2);空间复杂度：O(n)

1. **实验小结（记录实验过程中遇到的主要问题和心得）**

掌握栈与环形队列应用的的算法设计，栈的排序可以通过冒泡排序实现，环形队列的排序通过对元素的遍历和最小元素查找来完成。