# 第 3 章栈和队列作业

**一、填空题**

1. **向量(线性表)、栈和队列都是 线性结构，可以在向量的 任意 位置插入和删除元素； 对于栈只能在 栈顶插入和删除元素；对于队列只能在 队尾 插入和 队头 删除元素。**
2. **栈是一种特殊的线性表，允许插入和删除运算的一端称为 栈顶 。不允许插入和删除运算的一端称为 栈底 。**
3. 栈 **是被限定为只能在表的一端进行插入运算，在表的另一端进行删除运算的线性表。**
4. **在具有 n 个单元的循环队列中，队满时共有 n-1个元素。**
5. **向栈中压入元素的操作是先 进栈 ，后 压栈 。**
6. **从循环队列中删除一个元素时，其操作是 先 出队 ，后 删除 。**
7. **带表头结点的空循环双向链表的长度等于 0 。**

# 二、判断正误（判断下列概念的正确性，并作出简要的说明。）

（ T）1. 线性表的每个结点只能是一个简单类型，而链表的每个结点可以是一个复杂类型。

（ F）2. 在表结构中最常用的是线性表，栈和队列不太常用。

（ T）3. 栈是一种对所有插入、删除操作限于在表的一端进行的线性表，是一种后进先出型结构。

（ F）4. 对于不同的使用者，一个表结构既可以是栈，也可以是队列，也可以是线性表。

（ T）5. 栈和链表是两种不同的数据结构。

（ F）6. 栈和队列是一种非线性数据结构。

（ T）7. 栈和队列的存储方式既可是顺序方式，也可是链接方式。

（ T）8. 两个栈共享一片连续内存空间时，为提高内存利用率，减少溢出机会，应把两个栈的栈底分别设在这片内存空间的两端。

（ T）9. 队是一种插入与删除操作分别在表的两端进行的线性表，是一种先进后出型结构。

（ F）10. 一个栈的输入序列是 12345，则栈的输出序列不可能是 12345。

# 三、单项选择题

（ B）1. 栈中元素的进出原则是

Ａ．先进先出 Ｂ．后进先出 Ｃ．栈空则进 Ｄ．栈满则出

（ C）2. 若已知一个栈的入栈序列是 1，2，3，…，n，其输出序列为 p1，p2，p3，…， pn，若 p1=n，则 pi 为

Ａ．i Ｂ．n=i Ｃ．n-i+1 Ｄ．不确定

（ B）3. 判定一个栈 ST（最多元素为 m0）为空的条件是

Ａ ． ST->top<>0 Ｂ ． ST->top=0 Ｃ ． ST->top<>m0

Ｄ．ST->top=m0

（ B）4. 判定一个队列 QU（最多元素为 m0）为满队列的条件是Ａ．QU->rear － QU->front = = m0

Ｂ．QU->rear － QU->front －1= = m0

Ｃ．QU->front = = QU->rear

Ｄ．QU->front = = QU->rear+1

四、简答题

1、简述队列和堆栈这两种数据类型的相同点和差异处

两者本质上都是线性表,都有结点和指向下一个元素的指针,都可以进行删除插入操作

队列先进先出,而栈先进后出

2、写出以下程序段的输出结果（队列中的元素类型 QElemType 为 char） void main()

{

Queue Q; InitQueue(Q); char x= 'e'；y= 'c';

EnQueue(Q，'h');

EnQueue(Q，'r'); EnQueue(Q，y); DeQueue(Q，x); EnQueue(Q， x); DeQueue(Q， x); EnQueue(Q， 'a'); While(!QueueEmpty(Q))

{

DeQueue(Q，y); Printf(y);

}

Prinntf(x);

}

X=r

Cha

输出

char

3、简述以下算法的功能（栈和队列的元素类型均为 int） void algo3(Queue &Q)

{

Stack S; int d;

InitStack(S); while(!QueueEmpty(Q))

{

DeQueue(Q,d);

Push(S,d);

}

while(!StackEmpty(S))

{

Pop(S, d);

EnQueue(Q, d);

}

}

^

将队列 Q 中的元素逆序，并将它们重新排列在队列中,通过辅助栈来实现

4、按照四则运算加、减、乘、除和幂运算（↑）优先关系的惯例，画出将下列算术表达式 变为后缀表达式栈及输出串的过程：

A－B＊Ｃ／Ｄ＋Ｅ↑Ｆ

printString -> o

Stack ->s

执行过程(如果没有变化就不列出来了)

1. O:A S:null
2. S:-
3. O:AB
4. S:-\*
5. O:ABC
6. O:ABC\* S:-/
7. O:ABC\*D
8. O:ABC\*D/ S-+
9. O:ABC\*D/E
10. S-+^
11. O:ABC\*D/EF
12. O:ABC\*D/EF^+ - S:null