**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 任何 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 插入元素和 队首 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. 队列 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 移动栈顶指针 ，后 存入元素 。**

**6. 从循环队列中删除一个元素时，其操作是先 移动队首元素 ，后 取出元素 。**

**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。）**

**（ × ）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。**

**（ × ）2. 在表结构中最常用的是线性表，栈和队列不太常用。**

**（ √ ）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。**

**（ √ ）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。**

**（ × ）5. 栈和链表是两种不同的数据结构。**

**（ × ）6. 栈和队列是一种非线性数据结构。**

**（ √ ）7. 栈和队列的存储方式既可是顺序方式，也可是链接方式。**

**（ × ）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。**

**（ × ）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。**

**（ × ）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**

**（ A ）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**

**答：**

**相同点：两者都是线性结构，都是逻辑结构的概念。**

**都可以用顺序存储或链式存储。**

**堆栈和队列是两种特殊的线性表，即对插入和删除运算加以限制的受限的线性表。**

**不同点：运算规则不同，堆栈只允许在一段进行插入与删除运算，是后进先出表；而队列只允许在一端进行插入，另一端进行删除运算，是先进先出表。**

**用途不同，堆栈多用于函数调用、保护进程等，而队列多用于多到作业处理、指令寄存及其他运算等等。**

**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);**

**}**

**答：输出为“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);**

**}**

**}**

**答：功能是利用堆栈做辅助，将队列中的数据元素进行逆置。**

**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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **OPTR栈** | **OPND栈** | **输入字符** | **主要操作** |
| **1** | **#** |  | **A – B \* C / D + E ↑ F** | **PUSH(OPND,’A’)** |
| **2** | **#** | **A** | **– B \* C / D + E ↑ F** | **PUSH(OPTR,’-’)** |
| **3** | **# -** | **A** | **B \* C / D + E ↑ F** | **PUSH(OPND,’B’)** |
| **4** | **# -** | **AB** | **\* C / D + E ↑ F** | **PUSH(OPTR,’\*’)** |
| **5** | **# - \*** | **AB** | **C / D + E ↑ F** | **PUSH(OPND,’C’)** |
| **6** | **# - \*** | **ABC** | **/ D + E ↑ F** | **X = B \* C** |
| **7** | **# -** | **AX** | **/ D + E ↑ F** | **PUSH(OPTR,’/’)** |
| **8** | **# -** | **AX** | **D + E ↑ F** | **PUSH(OPND,’D’)** |
| **9** | **# - /** | **AXD** | **+ E ↑ F** | **Y = X / D** |
| **10** | **# - /** | **AY** | **+ E ↑ F** | **Z = A - Y** |
| **11** | **#** | **Y** | **+ E ↑ F** | **PUSH(OPTR,’+’)** |
| **12** | **# +** | **Z** | **E ↑ F** | **PUSH(OPND,’E’)** |
| **13** | **# +** | **ZE** | **↑ F** | **PUSH(OPTR,’↑’)** |
| **14** | **# + ↑** | **ZE** | **F** | **PUSH(OPND,’F’)** |
| **15** | **# + ↑** | **ZEF** | **#** | **O = E ↑ F** |
| **16** | **# +** | **ZO** | **#** | **N = Z + O** |
| **17** | **#** | **N** | **#** | **RETURN (N)** |