## 第三章 限定性线性表—栈和队列

**一、**判断题

1. 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。（ √ ）

解析：“双端栈”，书P81页

1. 对同一输入序列（元素互不相同）进行两组不同的合法的入栈和出栈组合操作，所得的输出序列一定相同。（ × ）

解析：栈的性质是后进先出，不同的入栈组合得到的输出序列不一定相同。

1. 若一个栈的输入序列为1，2，3，4，5，6，则可得到输出序列1,5,4,6,2,3。（ × ）

解析：栈的性质是后进先出，所以3应该在2前输出

1. 队列是一种插入与删除操作分别在表的两端进行的线性表，是一种先进先出型结构。（ √ ）
2. 栈和队列的存储方式，既可以是顺序方式，又可以是链式方式。（ √ ）
3. 在栈空的情况下，不能作出栈操作，否则产生下溢，在栈满的情况下，不能作入栈操作，否则产生上溢。（ √ ）
4. 将十进制数转换为二进制数是栈的典型应用之一。（ √ ）

解析：栈的性质是后进先出，十进制转二进制是从下往上取值，所以是正确的。

1. 在带头结点的链队列上做出队操作时，不会改变front指针的值，但可能会改变rear指针的值。（√ ）

解析：front指针始终指向头结点，所以做出队操作不会改变front指针的值，rear指针指向最后一个元素，如果该队列只有一个元素，则会改变rear指针的值。

1. 循环队列中，若尾指针rear大于头指针front，其元素个数为rear-front。（ **√** ）
2. 只要还有可用空间，链栈和链队就不会出现栈满和队满的情况。（ **√** ）

二、选择题

1. 栈操作的原则是（ B ）。

A. 先进先出 B. 后进先出 C. 后进后出 D. 不分顺序

1. 在作进栈运算时,应先判别栈是否( ①B ),在作退栈运算时应先判别栈是否( ②A )。当栈中元素为n个,作进栈运算时发生上溢,则说明该栈的最大容量为( ③B )。为了增加内存空间的利用率和减少溢出的可能性,由两个栈共享一片连续的内存空间时,应将两栈的 ( ④ D )分别设在这片内存空间的两端,这样,当( ⑤C )时，才产生上溢。

①, ②: A. 空 B. 满 C. 上溢 D. 下溢

③: A. n-1 B. n C. n+1 D. n/2

④: A. 长度 B. 深度 C. 栈顶 D. 栈底

⑤: A. 两个栈的栈顶同时到达栈空间的中心点.

B. 其中一个栈的栈顶到达栈空间的中心点.

C. 两个栈的栈顶在栈空间的某一位置相遇.

D. 两个栈均不空,且一个栈的栈顶到达另一个栈的栈底.

1. 一个栈的输入序列为123…n，若输出序列的第一个元素是n，输出第i（1<=i<=n）个元素是（ B ）。
2. 不确定 B. n-i+1 C. i D. n-i

解析：栈的性质是后进先出，n是最后一个入栈的，即为栈顶，因为输出的第一个元素是n，所以其余的元素仍然在堆栈中，所以说第i个元素是n-(i-1)。

1. 若一个栈的输入序列为1,2,3,…,n，输出序列的第一个元素是i，则第j个输出元素是（ D ）。

A. i-j-1 B. i-j C. j-i+1 D. 不确定的

1. 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pN是n，则pi是( D )。

A. i B. n-i C. n-i+1 D. 不确定

1. 设abcdef以所给的次序进栈，若在进栈操作时，允许退栈操作,则下面得不到的序列为（ D ）。
2. fedcba B. bcafed C. dcefba D. cabdef

解析：由于栈的性质为后进先出，所以D中ab是不可以实现的

1. 输入序列为ABC，将序列变为CBA时，经过的栈操作为（ B ）

A. push,pop,push,pop,push,pop B. push,push,push,pop,pop,pop

C. push,push,pop,pop,push,pop D. push,pop,push,push,pop,pop

1. 若一个栈以向量V[1..n]存储，初始栈顶指针top为n+1，则下面x进栈的正确操作是( C )。

A．top=top+1; V[top]=x B. V[top]=x; top=top+1

C. top=top-1; V[top]=x D. V[top]=x; top=top-1

解析：进栈是先移动栈顶指针，再存入元素。

1. 若栈采用顺序存储方式存储，现两栈共享空间V[1..m]，top[i]代表第i个栈( i =1,2)栈顶，栈1的底在v[1]，栈2的底在V[m]，则栈满的条件是（ B ）。

A. |top[2]-top[1]|=0 B. top[1]+1=top[2]

C. top[1]+top[2]=m D. top[1]=top[2]

解析：判断栈是否满看两个栈顶是否相邻。

1. 栈在（ D ）中有应用。

A. 递归调用 B. 子程序调用

C. 表达式求值 D. A，Ｂ，Ｃ

1. 一个递归算法必须包括（ B ）。

A. 递归部分 B. 终止条件和递归部分

C. 迭代部分 D. 终止条件和迭代部分

1. 设计判别表达式中左，右括号是否匹配的算法，应采用（ D ）数据结构最佳。

A．线性表的顺序存储结构 B. 队列

C. 线性表的链式存储结构 D. 栈

解析：使用栈解决的方法：把表达式依次压入栈，当压入的是右括号时，就退栈直到退出一个左括号，若最终栈空，则表示配对完成。

1. 用链接方式存储的队列，在进行删除运算时（ D ）。

A. 仅修改头指针 B. 仅修改尾指针

C. 头、尾指针都要修改 D. 头、尾指针可能都要修改

解析：队列的性质是先进先出，所以一般情况下仅修改队头指针，但只有一个元素时，出队后队空，此时还需要修改队尾指针。

1. 假设以数组A[m]存放循环队列的元素,其头尾指针分别为front和rear，则当前队列中的元素个数为（ A ）。

A．(rear-front+m)%m B．rear-front+1

C．(front-rear+m)%m D．(rear-front)%m

解析：如果rear>=front，元素个数为rear-front,如果rear<front，元素的个数为m-(front-rear)。

1. 循环队列存储在数组A[0..m]中，则入队时的操作为（ D ）。

A. rear=rear+1 B. rear=(rear+1) mod (m-1)

C. rear=(rear+1)mod m D. rear=(rear+1)mod(m+1)

解析：进队操作时，队尾指针的变化时rear=(rear+1) mod MAXSIZE,详解P100。

1. 若用一个大小为6的数组来实现循环队列，且当前rear和front的值分别为0和3，当从队列中删除一个元素，再加入两个元素后，rear和front的值分别为多少？( B )
2. 1和 5 B. 2和4 C. 4和2 D. 5和1

解析：队列的性质是先进先出，删除一个元素，front就会往下移动一次，则front值为2，加入两个元素，rear便会往下移动两次，是循环链表，所以rear值为4。

1. 最大容量为n的循环队列，队尾指针是rear，队头指针是front，则队空的条件是（ B ）。

A. (rear+1) MOD n=front B. rear=front

C．rear+1=front D. (rear-l) MOD n=front

解析：最大容量为n的循环队列，队满的条件是(rear+1)%n==front,队空的条件是rear==front。

1. 设有一个顺序栈S，元素A、B、C、D、E、F依次进栈，如果6个元素出栈的顺序是B、D、C、F、E、A，则栈的容量至少应是（ A ）

A、3 B、4 C、5 D、 6

出栈元素 栈内元素

B A

D C A

C A

F E A

F A

A NULL

所以栈内元素最多时是ACD,AEF。

1. 从一个栈顶指针为top的链栈（无头结点）中删除一个结点时，用x保存被删除的结点，应执行下列（ D ）命令。

A、x=top;top=top->next; B、top=top->next;x=top->data;

C、x=top->data; D、x=top->data;top=top->next;

1. 在一个栈顶指针为HS的链栈（无头结点）中，将S指针所指的结点入栈，应执行下列（ C ）命令。

A、HS->next=S； B、S->next=HS->next；HS->next=S；

C、S->next=HS；HS=S D、S->next=HS；HS=HS->next；

1. 4个元素按A、B、C、D顺序进S栈，执行两次Pop（S，x）运算后，栈顶元素的值是（ B ）

A、A B、B C、C D、D

1. 循环队列占用的空间（ A ）。

A．必须连续 B．不必连续 C．不能连续 D．可以不连续

1. 若进队的序列为A、B、C、D，则出队的序列只能是（ C ）。

A．B、C、D、A B．A、C、B、D

C．A、B、C、D D．C、B、D、A

三、填空题

1. (顺序)栈是限定仅在表尾进行插入或删除操作的线性表。
2. 一个栈的输入序列是：1，2，3则不可能的栈输出序列是3,1,2。
3. 设有一个空栈，栈顶指针为1000H（十六进制），现有输入序列为1，2，3，4，5，经过PUSH, PUSH, POP,PUSH,POP,PUSH,PUSH之后，输出的序列是23，而栈顶指针值是100CH（十六进制）。设栈为顺序栈，每个元素占4个字节。

解析：栈内还有1，4，5三个元素，每个元素占4个字节，即1000H+4\*3=100CH

1. 当两个栈共享一存储区时，栈利用一维数组stack(1,n)表示，两栈顶指针为top[1]与top[2]，则当栈1空时，top[1]为0，栈2空时 ，top[2]为n+1，栈满时为top[1]+1=top[2]。
2. 多个栈（多于2个）共存时，最好用链式存储结构作为存储结构。
3. 用S表示入栈操作，X表示出栈操作，若元素入栈的顺序为1234，为了得到1342出栈顺序，相应的S和X的操作串为SXSSXSXX。
4. 循环队列的引入，目的是为了克服假溢出时大量移动数据元素。
5. 队列是限制插入只能在表的一端，而删除在表的另一端进行的线性表，其特点是先进先出。
6. 已知链队列的头尾指针分别是f和r，则将值x入队的操作序列是s=(LinkedList)malloc(sizeof(LNode))；s->data=x; s->next=r->next； r->next=s；r=s；。

10.区分循环队列的满与空，通常的两种方法是牺牲一个存储单元和设标记。

11.设循环队列用数组A[1..M]表示，队首、队尾指针分别是FRONT和TAIL，判定队满的条件为（TAIL+1）% M==FRONT。

12.表达式求值是栈应用的一个典型例子。

13.在有n个元素的栈中，进栈和出栈操作的时间复杂度都是O（1）。

14.向一个栈顶指针为top的链栈（无头结点）插入一个新结点\*p时，应执行p->next=top; top=p操作。

15.设长度为n的链队列用单循环链表表示，若只设头指针，则入队操作的时间复杂度为O（n）。

16.设长度为n的链队列用单循环链表表示，若只设尾指针，则出队操作的时间复杂度为O（1）。

17.在一个带头结点链队列Q中，若队首指针为front，队尾指针为rear，则判断该队列只有一个结点的条件为Q->rear==Q->front->next。

四、简答题

1. 有5 个元素，其入栈次序为：A，B，C，D，E，在各种可能的出栈次序中，以元素C，D最先出栈（即C第一个且D第二个出栈）的次序有哪几个？

解析：三个：CDEBA，CDBEA，CDBAE

1. 如果输入序列为1 2 3 4 5 6,试问能否通过栈结构得到以下两个序列:4 3 5 6 1 2和1 3 5 4 2 6;请说明为什么不能或如何才能得到。

解析：输入序列为123456，不能得出435612，其理由是，输出序列最后两元素是12，前面4个元素（4356）得到后，栈中元素剩12，且2在栈顶，不可能栈底元素1在栈顶元素2之前出栈。得到135426的过程如下：1入栈并出栈，得到部分输出序列1；然后2和3入栈，3出栈，部分输出序列变为：13；接着4和5入栈，5，4和2依次出栈，部分输出序列变为13542；最后6入栈并退栈，得最终结果135426。

1. 画出对算术表达式A-B\*C/D-E↑F求值时操作数栈和运算符栈的变化过程。

解析：设操作数栈是opnd，操作符栈是optr，对算术表达式A-B\*C/D-E↑F求值，过程如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 步骤 | opnd栈 | optr栈 | 输入字符 | 主要操作 |
| 初始 |  | # | A-B\*C/D-E↑F# | PUSH(OPTR,’#’) |
| 1 | A | # | A-B\*C/D-E↑F# | PUSH(OPND,A) |
| 2 | A | # - | -B\*C/D-E↑F# | PUSH(OPTR,’-’) |
| 3 | AB | # - | B\*C/D-E↑F# | PUSH(OPND,B) |
| 4 | AB | # - \* | \*C/D-E↑F# | PUSH(OPTR,’\*’) |
| 5 | ABC | # - \* | C/D-E↑F# | PUSH(OPND,C) |
| 6 | AT(T=B\*C) | # - / | /D-E↑F# | PUSH(OPND,POP(OPND)\*POP(OPND))  PUSH(OPTR,’/’) |
| 7 | ATD | # - / | D-E↑F# | PUSH(OPND,D) |
| 8 | AT(T=T/D)  T(T=A-T) | # -  # - | -E↑F# | x=POP(OPND);y=POP(OPND)  PUSH(OPND,y/x);  x=POP(OPND);y=POP(OPND);  PUSH(OPND,y-x)  PUSH(OPTR,’-’) |
| 9 | TE | # - | E↑F# | PUSH(OPND,E) |
| 10 | TE | # -↑ | ↑F# | PUSH(OPTR, ‘↑’) |
| 11 | TEF | # -↑ | F# | PUSH(OPND,F) |
| 12 | TE  TS(S=E↑F)  R(R=T-S) | #-# | # | X=POP(OPND) Y=POP(OPND)  POP(OPTR) PUSH(OPND,y↑x)  x=POP(OPND) y=POP(OPND)  POP(OPTR) PUSH(OPND,y-x) |