第3章 栈和队列

本章主要讨论3种线性结构：栈、队列与优先级队列，以及它们的应用。这3种结构都是顺序存取的表，而且都是限制存取点的表。栈限定只能在表的一端(栈顶)插入与删除，其特点是先进后出。队列和优先级队列限定只能在表的一端(队尾)插入在另一端(队头)删除，不过优先级队列在插入和删除时需要根据数据对象的优先级做适当的调整，令优先级最高的对象调整到队头，其特点是优先级高的先出。而队列不调整，其特点是先进先出。这几种结构在开发各种软件时非常有用。

3.1 复习要点

本章复习的要点如下。

(1) 栈和队列的定义及其特点，包括：

① 栈的定义。注意栈顶进出、栈底不能进出、顺序存取的概念，栈的先进后出的特点。

② 队列的定义。注意队头出、队尾进，顺序存取的概念，队列的先进先出的特点。

③ 注意区分栈、队列、向量(一维数组)。栈和队列是顺序存取的，向量是直接存取的。

④ 以1,2,…,n进栈，计算可能的出栈序列。掌握是否是可能出栈序列的识别方法。

⑤ 以1,2,…,n进队，计算可能的出队序列。由于队列特性，可能出队序列只有一种。

⑥ 进栈、出栈、判空、置空操作的使用。

⑦ 进队、出队、判空、置空操作的使用。

⑧ 栈空、队空在实际使用时不算出错，它标志某种处理的结束。

⑨ 栈满、队满在实际使用时用来判断可能的出错情形。

(2) 栈的存储表示及其基本运算的实现：

① 顺序栈的类结构定义。注意静态数组定义和动态数组定义的不同，以及maxSize的出现位置。

② 顺序栈的栈顶指针实际指示的位置，以及如何用栈顶指针表示顺序栈的栈空、栈满条件。

③ 顺序栈的进栈、出栈运算的实现。注意操作的前置条件和后置条件，以及在参数表中引用参数的作用。

④ 双栈共用一个数组的进栈、退栈、置空栈、判栈空算法及栈满、栈空条件。

⑤ 链式栈的结构定义。注意链式栈的栈顶指针实际指示的位置。

⑥ 链式栈的进栈、出栈运算的实现。注意链式栈的栈空条件。

⑦ 栈满时的扩充算法。

(3) 队列的存储表示及其基本运算的实现：

① 循环队列的类结构定义。注意队头指针和队尾指针进退的方向，以及如何用这两个指针判断队列空和队列满。

② 当牺牲一个单元来区分队列空和队列满时，有两种进队处理方式：一种是先存数据再让队尾指针进一；另一种是先让队尾指针进一再存数据。注意队头指针和队尾指针实际指示的位置。

③ 循环队列的进队、出队的取模操作的实现。

④ 使用 tag 区分队列空和队列满的循环队列的进队列和出队列操作的实现。

⑤ 使用队头指针 front 和队列长度 length 实现循环队列时进队列和出队列的操作。

⑥ 链式队列的类结构定义。注意链式队列的队头指针和队尾指针的位置。

⑦ 链式队列的进队列和出队列操作的实现。注意操作的前置条件和后置条件，以及在参数表中引用参数的作用。

⑧ 双端队列的类结构定义。如何区分输入受限和输出受限。

⑨ 双端队列的进队、出队运算的实现。注意队列空和队列满的条件。

⑩ 优先级队列的存储结构。它的最佳存储表示应是堆(heap)，本章介绍的表示看懂即可。

(4) 栈的应用：

① 栈在递归过程中作为工作栈的使用。

② 在递归算法中根据递归深度计算栈容量的方法。

③ 根据数据的进栈和出栈序列计算栈容量的方法。

④ 在栈式铁路调车线上当进栈序列为 1,2,3,…,n 时，可能的出栈序列及其计数。

⑤ 栈在表达式计算中从中缀表示转后缀表示，以及用后缀表示求值时的使用。

⑥ 栈在括号配对中的应用。

⑦ 栈在数制转换中的应用。

⑧ 双栈共用一个数组的进栈、退栈、置空栈、判栈空算法及栈满、栈空条件。

⑨ 使用两个栈模拟一个队列时的进队列和出队列算法。

(5) 队列的应用：

① 队列在分层处理中的使用，包括二叉树、树、图等层次遍历过程中的使用。

② 队列在对数据循环处理过程中的使用，例如约瑟夫问题、归并排序。

③ 队列在调度算法中的使用。

④ 队列在缓冲区处理中的应用，如输入输出缓冲区、用于并行处理的缓冲区队列。

3.2 难点和重点

本章的知识点有 5 个，包括栈和队列的定义及其特点，栈的存储表示及其基本运算的实现，队列的存储表示及其基本运算的实现，栈的应用和队列的应用。

(1) 栈和队列的定义及其特点：

① 元素 1,2,3,4 依次进栈，可能的出栈序列有多少种？队列呢？

② 当元素以 A,B,C,D,E 顺序进栈，D,B,C,E,A 是可能的出栈顺序吗？

③ 可否用两个栈模拟一个队列？反过来呢？

④ 栈、队列对线性表加了什么限制？

(2) 栈的存储表示及其基本运算的实现：

① 当栈空时顺序栈的栈顶指针 top＝－1，当栈非空时 top 是否指示最后元素加入的位置？

② 顺序栈的进栈、出栈的先决条件是什么？

③ 当一个顺序栈已满，如何才能扩充栈长度，使得客户程序能够继续使用这个栈？

④ 当两个栈共享同一个存储空间 V[m]时，可设栈顶指针数组 t[2]和栈底指针数组 b[2]。如果进栈采用两个栈相向前行的方式，则任一栈的栈满条件是什么？

⑤ 链式栈的栈顶指针是指在链头还是链尾？

⑥ 链式栈只能顺序存取，而顺序栈不但能顺序存取，还能直接存取，这话对吗？

⑦ 理论上链式栈没有栈满问题，但在进栈操作实现时，还要判断一个后置条件，是何条件？

(3) 队列的存储表示及其基本运算的实现：

① 当用牺牲一个单元的方式组织循环队列时，队空和队满的条件是什么？进队、出队的策略是什么？

② 当用队头指针 front 和长度 length 组织循环队列时，队空和队满的条件是什么？进队和出队的策略是什么？(设表长度为 m)

③ 链式队列的队头和队尾在链表的什么地方？

④ 链式队列的队空条件是什么？

⑤ 同时使用多个队列时需采用何种队列结构？如何组织？

⑥ 链式队列的每个结点是否还可是队列？

(4) 栈的应用：

① 在后缀表达式求值过程中用栈存放什么？在中缀表达式求值过程中又用栈存放什么？

② 为判断表达式中的括号是否配对，可采用何种结构辅助进行判断？

③ 在递归算法中采用何种结构来存放递归过程每层的局部变量、返回地址和实参副本？

④ 在回溯法中采用何种结构来记录回退路径？

⑤ 若进栈序列为 1,2,3,4,5,6，出栈序列为 2,4,3,6,5,1，问栈容量至少多大？

⑥ 常用的一种链式栈是基于静态链表的。用一个整数数组 S[n]存放链接指针(游标)，设初始时 top＝－1，表示栈空，则其进栈、出栈、判栈空等操作如何实现？

(5) 队列的应用：

① 在逐层处理一个分层结构的数据时，需采用何种辅助结构来组织数据？

② 为实现输入→处理→输出并行操作需建立多个输入缓冲区队列，这些队列是按数组方式组织的还是按链表方式组织的？

③ 在操作系统中一种进程调度策略是先来先服务，为此使用了何种辅助结构？

④ 在对一个无序单链表进行自然归并排序时，可先把链表截出一段段有序的子链表，再对它们做两路归并排序。为此定义队列来辅助排序，此队列的元素的数据类型是什么？

⑤ 双端队列的作用是什么？有几种双端队列？

⑥ 双端队列的队空条件和队满条件是什么？

⑦ 优先级队列的作用是什么？在插入和删除时如何进行调整？

3.3 教材习题解析

一、思考题

3-1 铁路进行列车调度时，常把站台设计成栈式结构的站台，如图 3-1 所示。试问：

(1) 设有编号为 1,2,3,4,5,6 的 6 辆列车，顺序开入栈式结构的站台，则可能的出栈序列有多少种？

(2) 若进站的 6 辆列车顺序如上所述，那么是否能够得到 435612,325641,154623 和 135426 的出站序列，如果不能，说明为什么不能；如果能，说明如何得到(即写出“进栈”或“出栈”的序列)。

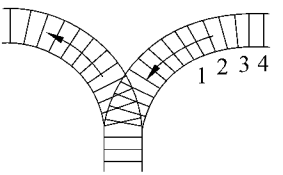


图 3-1

【解答】

(1) 编号为 1,2,3,4,5,6 的 6 辆列车顺序开入栈式结构站台，可能的不同出栈序列有

$$\frac{1}{n+1}C_{2n}^{n}=\frac{1}{6+1}C_{12}^{6}=\frac{1}{7}\frac{12\times11\times10\times9\times8\times7}{6\times5\times4\times3\times2\times1}=132\text{ 种。}$$

(2) 不能得到 435612 和 154623 这样的出栈序列。因为若在 4,3,5,6 之后再将 1,2 出栈，则 1,2 必须一直在栈中，此时 1 先进栈，2 后进栈，2 应压在 1 上面，不可能 1 先于 2 出栈。154623 也是这种情况。出栈序列 325641 和 135426 可以得到，如图 3-2 所示。

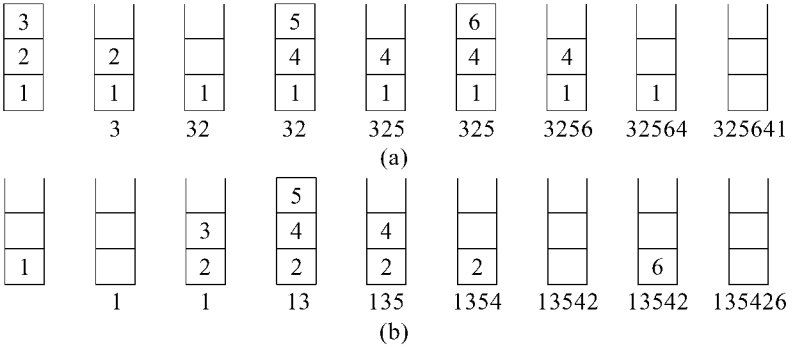


图 3-2

3-2 试证明：若借助栈可由输入序列 1,2,3,…,n 得到一个输出序列 $p_1,p_2,p_3,\cdots,p_n$(它是输入序列的某一种排列)，则在输出序列中不可能出现以下情况，即存在 $i<j<k$，使得 $p_j<p_k<p_i$。(提示：用反证法)

【证明】

(1) 必要性：

按照题意，当 $i<j<k$ 时进栈顺序是 i、j、k，这 3 个元素出栈的相对顺序是 p_i、p_j、p_k。例如，当 $i=1,j=2,k=3$ 时一个合理的出栈序列是 $p_i=2,p_j=3,p_k=1$。如果 $p_j<p_k<p_i$ 成立，意味着出栈顺序为 3、1、2，这恰恰是不可能的。当较大的数首先出栈时，那些较小的数都是降序压在栈内的，如 2、1，这些数不可能如 1、2 那样正序出栈。

(2) 充分性：

如果 $p_j<p_k<p_i$ 成立，表明当 $i<j<k$ 时各元素进栈出栈后的相对顺序为 p_j、p_k、p_i。现做具体分析：

当 $i<j$ 时 $p_j<p_i$，表明 p_j 是在 p_i 进栈后进栈并压在 p_i 上面，且 p_j 在 p_i 出栈前出栈；

当 $j<k$ 时 $p_j<p_k$，表明 p_j 必须在 p_k 入栈之前就出栈，否则 p_j 就被压在 p_k 下面了；

当 $i<k$ 时 $p_k<p_i$，表明 p_i 是先于 p_k 进栈的。

综上所述可知：这与正确的出栈顺序 $p_i<p_j<p_k$ 相矛盾。

3-3 设有一个顺序栈 S，元素 s_1，s_2，s_3，s_4，s_5，s_6 依次进栈，如果 6 个元素的出栈顺序为 s_2，s_3，s_4，s_6，s_5，s_1，则顺序栈的容量至少应为多少？

【解答】

至少为 3。参看如图 3-3 所示的栈的变化情况。

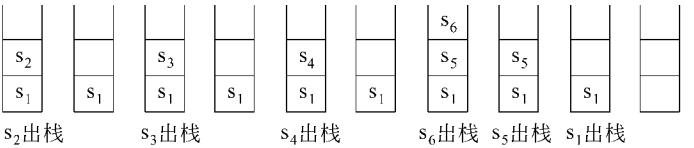


图 3-3

3-4 写出下列中缀表达式的后缀形式：

(1) A×B×C

(2) −A+B−C+D

(3) A×−B+C

(4) (A+B)×D+E/(F+A×D)+C

(5) A&&B‖!(E>F) （注：按 C++ 的优先级）

(6) !(A&&!((B<C)‖(C>D)))‖(C<E)

【解答】

(1) AB×C×

(2) A−B+C−D+

(3) AB−×C+

(4) AB+D×EFAD×+/+C+

(5) AB&&EF>!‖

(6) ABC<CD>‖!&&!CE<‖

3-5 根据课文中给出的优先级，回答以下问题：

(1) 在函数 postfix 中，如果表达式 e 含有 n 个操作符和分界符，问栈中最多可存入多少个元素？

(2) 如果表达式 e 含有 n 个运算符，且括号嵌套的最大深度为 6 层，问栈中最多可存入多少个元素？

【解答】

(1) 函数 postfix 是中缀转后缀的算法，在该程序中，用栈存放操作符。如果表达式 e 含有 n 个操作符和分界符，则栈中最多可存入 n 个操作符。栈中实际存放的操作符个数要

看各操作符的优先级和括号的使用。

(2) 在处理中缀表达式时,假定如课文所列,算术表达式中不同优先级别的操作符有3类:;,×或/或%,+或-。当栈外操作符的优先级高于栈内操作符的优先级时栈外运算符进栈,当栈外操作符的优先级低于栈内操作符的优先级则栈内操作符出栈。因此,在不考虑括号的情形下,需要3个栈单元存储表达式的操作符(包括';'号)就够了。

如果考虑括号:每遇到一个右括号,就需要处理括号内的表达式,结果是括号内的所有操作符连同左括号都出栈。因为表达式的括号嵌套最大深度为6层,假定每重括号在栈内操作符3个(左括号1个;+或-级别操作符1个;*或/或%级别操作符1个),6重括号在栈内占用了6×3=18个栈单元,总共栈可容纳18+3=21个操作符。

3-6 设表达式的中缀表示为a*x-b/x^2,试利用栈将它改为后缀表示ax*bx2^/-。写出转换过程中栈的变化。(其中的"^"表示乘方运算)

【解答】

若设当前扫描到的操作符ch的优先级为icp(ch),该操作符进栈后的优先级为isp(ch),则可规定各个算术操作符的优先级如表3-1所示。

表 3-1

操作符	;	(	^	*,/,%	+,-	)
isp	0	1	7	5	3	8
icp	0	8	6	4	2	1

isp也叫做栈内(in stack priority)优先数,icp也叫做栈外(in coming priority)优先数。当刚扫描到的操作符ch的icp(ch)大于isp(stack)时,则ch进栈;当刚扫描到的操作符ch的icp(ch)小于isp(stack)时,则位于栈顶的操作符退栈并输出。从表3-2中可知,icp("(")最高,但当"("进栈后,isp("(")变得极低。其他操作符进入栈中后优先数都升1,这样可体现在中缀表达式中相同优先级的操作符自左向右计算的要求。操作符优先数相等的情况只出现在括号配对")"或栈底的";"号与输入流最后的";"号配对时。前者将连续退出位于栈顶的操作符,直到遇到"("为止,然后将"("退栈以抵消括号,后者将结束算法。

表 3-2

步序	扫描项	项类型	动　作	栈的变化	输　出
0			';'进栈,读下一符号	;	
1	a	操作数	直接输出,读下一符号	;	a
2	*	操作符	isp(';')<icp('*'),进栈,读下一符号	;*	a
3	x	操作数	直接输出,读下一符号	;*	ax
4	-	操作符	isp('*')>icp('-'),退栈输出	;	ax*
			isp(';')<icp('-'),进栈,读下一符号	;-	ax*
5	b	操作数	直接输出,读下一符号	;-	ax*b
6	/	操作符	isp('-')<icp('/'),进栈,读下一符号	;-/	ax*b

续表

步序	扫描项	项类型	动　作	栈的变化	输　出
7	x	操作数	直接输出，读下一符号	;－/	ax * bx
8	^	操作符	isp('/')<icp('^')，进栈，读下一符号	;－/^	ax * bx
9	2	操作数	直接输出，读下一符号	;－/^	ax * bx2
10	;	操作符	isp('^')>icp(';')，退栈输出	;－/	ax * bx2^
			isp('/')>icp(';')，退栈输出	;－	ax * bx2^/
			isp('－')>icp(';')，退栈输出	;	ax * bx2^/－
			结束		

3-7　试利用运算符优先数法，画出对如下中缀算术表达式求值时运算符栈和运算对象栈的变化。

$$a+b*(c-d)-e\wedge f/g$$

【解答】

设在表达式计算时各运算符的优先规则如上一题所示。因为直接对中缀算术表达式求值时必须使用两个栈，分别对操作符和操作数进行处理，设操作符栈为 OPTR(operator 的缩写)，操作数栈为 OPND(operand 的缩写)。下面给出对中缀表达式求值的一般规则：

(1) 建立并初始化 OPTR 栈和 OPND 栈，然后在 OPTR 栈中压入一个“;”。

(2) 从头扫描中缀表达式，取一字符送入 ch。

(3) 当 ch 不等于“;”时，执行以下工作，否则结束算法。此时在 OPND 栈的栈顶得到运算结果。

① 如果 ch 是操作数，进 OPND 栈，从中缀表达式取下一字符送入 ch。

② 如果 ch 是操作符，比较 ch 的优先级 icp(ch)和 OPTR 栈顶操作符 isp(OPTR)的优先级：

- ✧ 若 icp(ch)>isp(OPTR)，则 ch 进 OPTR 栈，从中缀表达式取下一字符送入 ch。
- ✧ 若 icp(ch)<isp(OPTR)，则从 OPND 栈退出一个操作符作为第 2 操作数 a2，再退出一个操作符作为第 1 操作数 a1，从 OPTR 栈退出一个操作符 θ 形成运算指令(a1)θ(a2)，执行结果进 OPND 栈。
- ✧ 若 icp(ch)==isp(OPTR) 且 ch==“)”，则从 OPTR 栈退出栈顶的“(”，抵消括号，然后从中缀表达式取下一字符送入 ch。

根据以上规则，给出计算 a+b*(c－d)－e^f/g 时两个栈的变化，如表 3-3 所示。

3-8　设有一个双端队列，元素进入该队列的顺序是 1,2,3,4。试分别求出满足下列条件的输出序列。

(1) 能由输入受限的双端队列得到，但不能由输出受限的双端队列得到的输出序列。

(2) 能由输出受限的双端队列得到，但不能由输入受限的双端队列得到的输出序列。

(3) 既不能由输入受限的双端队列得到，又不能由输出受限的双端队列得到的输出序列。

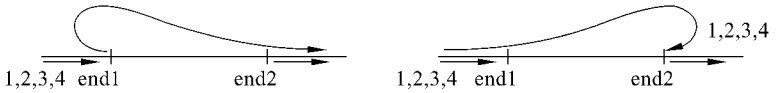
表 3-3

步序	扫描项	项类型	动　作	OPND 栈	OPTR 栈
0			OPTR 栈与 OPND 栈初始化，';'进 OPTR 栈，取第一个符号		;
1	a	操作数	a 进 OPND 栈，取下一符号	a	;
2	+	操作符	icp('+')>isp(';')，进 OPTR 栈，取下一符号	a	;+
3	b	操作数	b 进 OPND 栈，取下一符号	a b	;+
4	*	操作符	icp('*')>isp('+')，进 OPTR 栈，取下一符号	a b	;+*
5	(	操作符	icp('(')>isp('*')，进 OPTR 栈，取下一符号	a b	;+*(
6	c	操作数	c 进 OPND 栈，取下一符号	a b c	;+*(
7	−	操作符	icp('−')>isp('(')，进 OPTR 栈，取下一符号	a b	;+*(−
8	d	操作数	d 进 OPND 栈，取下一符号	a b c d	;+*(−
9	)	操作符	icp(')')<isp('−')，退 OPND 栈'd'，退 OPND 栈'c'，退 OPTR 栈'−'，计算 c−d → s1，结果进 OPND 栈	a b s_1	;+*(
10	同上	同上	icp(')')==isp('(')，退 OPTR 栈'('，对消括号，取下一符号	a b s_1	;+*
11	−	操作符	icp('−')<isp('*')，退 OPND 栈's_1'，退 OPND 栈'b'，退 OPTR 栈'*'，计算 b * s_1 → s_2，结果进 OPND 栈	a s_2	;+
12	同上	同上	icp('−')<isp('+')，退 OPND 栈's_2'，退 OPND 栈'a'，退 OPTR 栈'+'，计算 a * s_2 → s_3，结果进 OPND 栈	s_3	;
13	同上	同上	icp('−')>isp(';')，进 OPTR 栈，取下一符号	s_3	;−
14	e	操作数	e 进 OPND 栈，取下一符号	s_3 e	;−
15	^	操作符	icp('^')>isp('−')，进 OPTR 栈，取下一符号	s_3 e	;−^
16	f	操作数	f 进 OPND 栈，取下一符号	s_3 e f	;−^
17	/	操作符	icp('/')<isp('^')，退 OPND 栈'f'，退 OPND 栈'e'，退 OPTR 栈'^'，计算 e^f → s_4，结果进 OPND 栈	s_3 s_4	;−
18	同上	同上	icp('/')>isp('−')，进 OPTR 栈，取下一符号	s_3 s_4	;−/
19	g	操作数	g 进 OPND 栈，取下一符号	s_3 s_4 g	;−/
20	;	操作符	icp(';')<isp('/')，退 OPND 栈'g'，退 OPND 栈's_4'，退 OPTR 栈'/'，计算 s_4/g → s_5，结果进 OPND 栈	s_3 s_5	;−
21	同上	同上	icp(';')<isp('−')，退 OPND 栈's_5'，退 OPND 栈's_3'，退 OPTR 栈'−'，计算 s_3−s_5 → s_6，结果进 OPND 栈	s_6	;
22	同上	同上	icp(';')==isp(';')，退 OPND 栈's_6'，结束		;

【解答】

双端队列是一种插入和删除工作在两端均可进行的线性表。可以把双端队列看成是底元素连在一起的两个栈。它们与两个栈共享存储空间，不同之处在于两个栈的栈顶指针是往两端延伸的。由于双端队列允许在两端进行插入和删除元素，因此需设立两个指针 end1 和 end2，分别指着双端队列中两端的元素。

允许在一端进行插入和删除，另一端只允许插入的双端队列称为输出受限双端队列；允许在一端进行插入和删除，另一端只允许删除的双端队列称为输入受限双端队列，如图 3-4 所示。



(a) 输入受限的双端队列　　(b) 输出受限的双端队列

图　3-4

先看输入受限的双端队列。假设 end1 端输入 1,2,3,4，那么 end2 端的输出相当于队列的输出：1,2,3,4；而 end1 端的输出相当于栈的输出，n＝4 时仅通过 end1 端有 14 种输出序列，仅通过 end1 端不能得到的输出序列有 4!－14＝10 种，它们是：

1,4,2,3　2,4,1,3　3,4,1,2　3,1,4,2　3,1,2,4　4,1,2,3

4,1,3,2　4,2,1,3　4,2,3,1　4,3,1,2

通过 end1 和 end2 端混合输出，可以输出这 10 种中的 8 种，参看表 3-4。其中，S_L、X_L分别代表 end1 端的进队和出队，X_R代表 end2 端的出队。

表　3-4

输出序列	进队出队顺序	输出序列	进队出队顺序
1, 4, 2, 3	$S_L X_R S_L S_L S_L X_L X_R X_R$ ①①②③④④②③	3, 1, 2, 4	$S_L S_L S_L X_L X_R S_L X_R X_R$ ①②③③①④②④
2, 4, 1, 3	$S_L S_L X_L S_L S_L X_L X_R X_R$ ①②②③④④①③	4, 1, 2, 3	$S_L S_L S_L S_L X_L X_R X_R X_R$ ①②③④④①②③
3, 4, 1, 2	$S_L S_L S_L X_L S_L X_L X_R X_R$ ①②③③④④①②	4, 1, 3, 2	$S_L S_L S_L S_L X_L X_R X_L X_R$ ①②③④④①③②
3, 1, 4, 2	$S_L S_L S_L X_L X_R S_L X_L X_R$ ①②③③①④④②	4, 3, 1, 2	$S_L S_L S_L S_L X_L X_L X_R X_R$ ①②③④④③①②

还有两种是不可能通过输入受限的双端队列输出的，即 4,2,1,3 和 4,2,3,1。

再看输出受限的双端队列。假设 end1 端和 end2 端都能输入，仅 end2 端可以输出。如果都从 end2 端输入，从 end2 端输出，就是一个栈了，当输入序列为 1,2,3,4，输出序列可以有 14 种。对于其他 10 种不能输出的序列，通过交替从 end1 和 end2 端输入，还可以输出其中 8 种。设 S_L代表 end1 端的输入，S_R、X_R代表 end2 端的输入和输出，则有表 3-5。

表　3-5

输出序列	进队出队顺序	输出序列	进队出队顺序
1, 4, 2, 3	$S_L X_R S_L S_L S_R X_R X_R X_R$ ①①②③④④②③	3, 1, 2, 4	$S_L S_L S_R X_R X_R X_R S_L X_R$ ①②③③①②④④
2, 4, 1, 3	$S_L S_R X_R S_L S_R X_R X_R X_R$ ①②②③④④①③	4, 1, 2, 3	$S_L S_L S_L S_R X_R X_R X_R X_R$ ①②③④④①②③
3, 4, 1, 2	$S_L S_L S_R X_R S_R X_R X_R X_R$ ①②③③④④①②	4, 2, 1, 3	$S_L S_R S_L S_R X_R X_R X_R X_R$ ①②③④④②①③
3, 1, 4, 2	$S_L S_L S_R X_R X_R S_R X_R X_R$ ①②③③①④④②	4, 3, 1, 2	$S_L S_L S_R S_R X_R X_R X_R X_R$ ①②③④④③①②

通过输出受限的双端队列不能输出的两种序列是 4,1,3,2 和 4,2,3,1。

由此综合,可得:

(1) 能由输入受限双端队列得到,但不能由输出受限双端队列得到的输出序列是 4,1,3,2。

(2) 能由输出受限双端队列得到,但不能由输入受限双端队列得到的输出序列是 4,2,1,3。

(3) 既不能由输入受限的双端队列得到,又不能由输出受限双端队列得到的输出序列是 4,2,3,1。

二、练习题

3-9 改写顺序栈的进栈成员函数 Push (x),要求当栈满时执行一个 stackFull()操作进行栈满处理。其功能是:动态创建一个比原来的栈数组大二倍的新数组,代替原来的栈数组,原来栈数组中的元素占据新数组的前 maxSize 位置。

【解答】

```
template<class T>
void Seqstack<T>::Push (T item){
    if (isFull ()) stackFull ();                          //栈满,做溢出处理
    elements [++top]=item;                                //进栈
};
template<class T>
void Seqstack<T>::stackFull (){
    T * temp=new T[3 * maxSize];                          //创建体积大二倍的数组
    for (int i=0; i<=top; i++) temp[i]=elements[i];       //传送原数组的数据
    delete [] elements;                                   //删去原数组
    maxSize * =3;                                         //数组最大体积增大二倍
    elements=temp;                                        //新数组成为栈的数组空间
};
```

3-10 借助栈实现单链表上的逆置运算。

【解答】

算法的思路是:首先遍历单链表,逐个结点删除并把被删结点存入栈中,再从栈中取出存放的结点把它们依次链入单链表中。通过栈把结点的次序颠倒过来。

```
#include "LinkedStack.h"
template<class T, E>
void Reverse(List<T, E>& L){
    LinkedStack<LinkNode<T, E> * >S;  LinkNode<T, E> * p, * r;
    p=L.First()->link;
    if (p!=NULL){                                         //检测原链表
        r=p->link;  L.First()->link=r;                    //结点 p 从原链表摘下
        S.Push(p);                                        //进栈
        p=r;
    }
```