第3章 chapter 3

栈和队列

栈和队列在程序设计中应用十分广泛。栈和队列的逻辑结构和线性表相同，但它们是一种特殊的线性表。其特殊性在于运算受到了限制，插入和删除仅在表的一端或两端进行。因此栈和队列又称为操作受限的线性表。栈按“后进先出”的规则进行操作，队按“先进先出”的规则进行操作。本章主要学习栈和队列的逻辑结构、存储结构及运算操作的实现，以及栈和队列在软件设计中的应用。

3.1 知识点串讲

3.1.1 知识结构图

本章主要知识点的关系结构如图 3.1 所示。

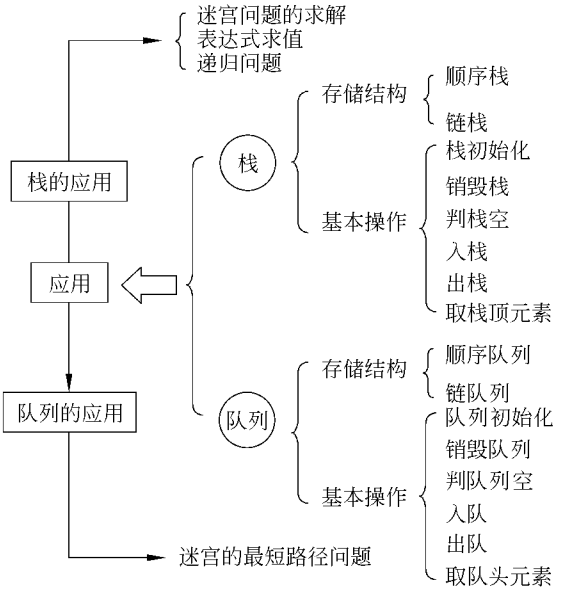

图 3.1 栈和队列的知识结构图

3.1.2 相关术语

(1) 栈顶、栈底、队尾、队头。
(2) 入栈、出栈。
(3) 入队、出队。
(4) 顺序栈、链栈。
(5) 循环队列、链式队列。

3.1.3 栈和队列的存储结构

栈和队列的存储与一般的线性表的实现类似,也有两种存储方式:顺序存储和链式存储。

1. 栈的顺序存储——顺序栈

顺序栈类似于顺序表,要分配一块连续的存储空间存放栈中的元素。这样需要用一个足够长度的一维数组来实现。栈底位置可以固定设置在数组的任一端(一般在下标为0的一端),而栈顶是随着插入和删除操作而变化,用一个 top 变量指明当前栈顶的位置。通常将 data 和 top 封装在一个结构体中,顺序栈的数据结构类型描述如下:

```
#define MAXSIZE 100                    /* 栈的最大容量 */
typedef struct{
    DataType data[MAXSIZE];            /* 栈的存储空间 */
    int top;
}SeqStack, * PSeqStack;
```

要点:
(1) 静态分配存储空间。
(2) 插入与删除操作仅在栈顶处执行。
(3) 注意顺序栈的溢出现象。
(4) “后进先出”规则。

2. 栈的链式存储——链栈

链栈结点结构与单链表的结构相同,即结点结构为:

```
typedef struct node{
    DataType data;
    struct node * next;
}StackNode, * PStackNode;
```

因为栈的主要运算是在栈顶进行插入、删除操作,显然在链表的头部作为栈顶的处理最方便,而且没有必要同单链表那样为了运算方便附加一个头结点。为了方便操作和强调栈顶是栈的一个属性,链栈的数据结构描述如下:

```
typedef struct {
            PStackNode top;
    }LinkStack, * PLinkStack;
```

要点：

(1) 链栈动态分配存储空间，无栈满问题，空间可扩充。

(2) 插入与删除仅在栈顶处执行。

(3) 链栈的栈顶在链头。

(4) “后进先出”规则。

3. 队列的顺序存储——顺序队列

顺序队列和顺序栈一样，要分配一块连续的存储空间来存放队列里的元素。但由于队列的队头和队尾都是活动的，因此需要用两个变量指针来指示队头和队尾位置，这一点和顺序栈不同。这里约定队头指向实际队头元素所在的位置的前一位置，队尾指向实际队尾元素所在的位置。

顺序队的类型定义如下：

```
#define MAXSIZE 100                    /* 队列的最大容量 */
typedef struct{
    DataType data[MAXSIZE];            /* 队列的存储空间 */
    int front, rear;                   /* 队头、队尾指针 */
}SeqQueue, * PSeqQueue;
```

要点：

(1) 静态分配存储空间。

(2) 入队操作是在队尾进行，出队操作是在队头进行。

(3) 注意顺序栈的溢出和“假溢出”现象。可以用循环队列解决“假溢出”问题。

(4) 注意队列空和满的条件。

(5) “先进先出”规则。

4. 队列的链式存储——链队列

链队列就是用一个单链表来表示队列。为了操作上的方便，需要设置一个头指针和尾指针分别指向队头和队尾元素。

链队列的描述如下：

```
typedef struct node{
    DataType data;
    struct node * next;
} Qnode, * PQNode;                     /* 链队列结点的类型 */
typedef struct {
    PQNnode front,rear;
}LinkQueue, * PLinkQueue;              /* 将头尾指针封装在一起的链队列 */
```

要点：

(1) 动态分配存储空间。

(2) 队头在链头，队尾在链尾。

(3) 链队列在进队时无队列满问题，但有队列空问题。注意队列空的条件。

(4) “先进先出”规则。

3.2 典型例题详解

一、选择题

1. 栈与一般线性表的区别在于_____。

A. 数据元素的类型不同　　B. 运算是否受限制

C. 数据元素的个数不同　　D. 逻辑数据不同

分析：该题目主要考查栈的定义。栈属于特殊的线性表，特殊性在于其删除和插入操作只能够在栈顶进行，是一种运算受到限制的线性表。答案为 B。

2. 一个顺序栈一旦被声明，其占用空间的大小_____。

A. 已固定　　B. 可以改变　　C. 不能固定　　D. 动态变化

分析：该题目主要考查顺序栈存储结构的特点。顺序栈用数组实现，因此一旦顺序栈被声明，则其空间大小固定。答案应选择 A。

3. 设有一顺序栈 S，元素 s1、s2、s3、s4、s5、s6 依次进栈，如果 6 个元素出栈的顺序是 s2、s4、s3、s6、s5、s1，则栈的容量至少应该是_____。

A. 3　　B. 4　　C. 5　　D. 6

分析：该题目主要考查顺序栈的存储空间定义。当 s2 出栈时，则栈的容量为 2(s1，s2 在栈中)；当 s4 出栈时，则栈的容量为 3(s1，s3，s4 在栈中)；当 s6 出栈时，则栈的容量为 3(s1，s5，s6 在栈中)；因此栈的最小容量应为 3。答案应选择 A。

4. 从一个栈顶指针为 top 的链栈中删除一个结点时，用 x 保存被删除的结点，执行_____。

A. x=top->data;top=top->next;

B. top=top->next;x=top;

C. x=top->data;

D. x=top;top=top->next;

分析：该题目主要考查链栈的具体操作。首先将指针变量 x 保存被删除结点，然后调整栈顶指针(top=top->next)。选项 A 中，x=top->data 操作目的是将栈顶结点的元素值赋给 x，故无法满足题目要求。选项 B 中，首先进行栈顶指针 top 调整，则 x 保存的不是当前删除的结点，而是栈调整后的栈顶元素。因此答案应选择 D。

5. 在一个栈顶指针为 top 的链栈中，将一个 s 指针所指的结点入栈，执行_____。

A. top->next=s;

B. s->next=top->next; top->next=s;

C. s->next=top; top=s;

D. s->next=top; top=top->next;

分析：该题目主要考查链栈的具体操作。根据链栈入栈操作定义，即可得到答案。答案为C。从选择题4、5可以看出，链栈的入栈和出栈操作实质上是对没有头结点的单链表进行插入与删除操作。

6. 链栈和顺序栈相比，有一个比较明显的优点，即_____。

A. 插入操作更加方便　　B. 通常不会出现栈满的情况

C. 不会出现栈空的情况　　D. 删除操作更加方便

分析：该题目主要考查链栈和顺序栈的特点。栈的两种存储方式各有特点，即顺序栈所需存储空间较少，但栈的大小受数组的限制；链栈由于每个结点都包含数据域和指针域，则所需空间相对较大，但栈的大小不受限制(受内存容量的限制)。所以答案为B。对顺序栈而言，其插入操作(入栈)不需要大量地移动元素，故插入操作(入栈)则不是链栈的优点，因此A选项的说法有问题。

7. 用单链表表示的链式队列的队头在链表的_____位置。

A. 链头　　B. 链尾　　C. 链中　　D. 任意位置

分析：该题目主要考查链式队列的存储结构，如图3.2所示。队列的队头是对队列元素进行删除的一端，链队列的队头在链表的链头位置(不考虑不包含数据元素的头结点)。答案为A。

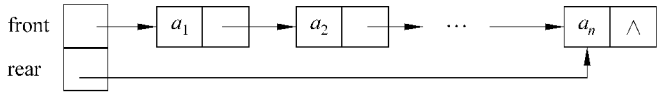

图3.2　链式队列

8. 在解决计算机主机与打印机之间速度不匹配问题时，通常设置一个打印数据缓冲区，主机将要输出的数据依次写入该缓冲区，而打印机则从该缓冲区中取出数据打印。该缓冲区应该是一个_____结构。

A. 堆栈　　B. 队列　　C. 数组　　D. 线性表

分析：该题目主要考查队列的性质和应用。缓冲区中的数据应该是先到达的先打印，所以使用具有FIFO性质的队列来实现。答案为B。

9. 做入栈运算时，应先判断栈是否为__(1)__，做出栈运算时，应先判断栈是否为__(2)__。当顺序栈中元素为n个，做入栈运算时发生上溢，则说明该栈的最大容量为__(3)__。为了增加内存空间的利用率和减少发生上溢的可能性，由两个顺序栈共享一片连续的内存空间时，应将两栈的__(4)__分别设在这片内存空间的两端，这样，只有当__(5)__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. 两个栈均不为空，且一个栈的栈顶到达另一个栈的栈底

分析：该题目主要考查栈的性质、结构和操作。答案为：(1)B；(2)A；(3)B；(4)D；(5)C。

10. 若已知一个栈的入栈序列是 1，2，3，…，30，其输出序列是 $p1$，$p2$，$p3$，…，pn，若 $p1=30$，则 $p10$ 为_____。

A. 11　　B. 20　　C. 30　　D. 21

分析：该题目主要考查栈的性质、结构和操作。已知数据的入栈序列是 1，2，3，…，30，出栈序列的第 1 个元素是 30。因此可以确定，所有元素是按入栈序列顺序全部入栈之后才开始出栈的。也就是说，出栈序列与入栈序列刚好相反，可求得出栈序列的第 10 个元素为 21，即 D 答案正确。

11. 循环队列 A[m]存放其元素，用 front 和 rear 分别表示队头及队尾，则循环队列满的条件是_____。

A. (Q->rear+1)%m==Q->front

B. Q->rear==Q->front+1

C. Q->rear+1==Q->front

D. Q->rear==Q->front

分析：该题目主要考查循环队列的存储结构以及队满的判断条件。循环队列的引入是为了解决队列存在的“假溢出”问题，但在循环队列中会出现队满和队空的判断条件相同而导致无法判断，通常采用少用一个元素空间的策略来解决该问题(其他策略请参考配套教材)。使队尾指针 Q->rear 无法赶上 Q->front，即队尾指针加 1 就会从后面赶上队头指针，这种情况下队满的条件是：(Q->rear+1)%m==Q->front。答案是 A。

12. 设数组 data[m]作为循环队列 sq 的存储空间，front 为队头指针，rear 为队尾指针，则执行出队操作后其头指针 front 值为_____。

A. front=front+1　　B. front=(front+1)%(m-1)

C. front=(front-1)%m　　D. front=(front+1)%m

分析：该题目主要考查循环队列的存储结构和出队操作。队列的头指针指向队首元素的实际位置，因此出队操作后，头指针需向上移动一个元素的位置。根据第 11 题的解答可知，循环队列的容量为 m，所以头指针 front 加 1 以后，需对 m 取余，使之自动实现循环，即当 front 取到最大下标($m-1$ 处)以后，自动循环回来取 0 值。所以答案是 D。

13. 由两个栈共享一个向量空间的好处是_____ 。

A. 减少存取时间，降低下溢发生的几率

B. 节省存储空间，降低上溢发生的几率

C. 减少存取时间，降低上溢发生的几率

D. 节省存储空间，降低下溢发生的几率

分析：该题目主要考查对顺序栈存储结构的理解。两个栈无论是共享向量空间还是

单独分配空间,对它们的操作所需的时间没有影响。两个栈共享向量空间,主要是为了节省存储空间,降低上溢的发生几率,因为当一个栈中的元素较少时,另一个栈可用空间可以超过向量空间的一半。答案应选择 B。

14. 若用单链表来表示队列,则应该选用_____。

A. 带尾指针的非循环链表　　B. 带尾指针的循环链表

C. 带头指针的非循环链表　　D. 带头指针的循环链表

分析:本题主要考查读者对循环单链表存储结构和队列定义的理解。设尾指针为 rear,则通过 rear 可以访问队尾,通过 rear->next 可以访问队头,因此带尾指针的循环链表较适合。答案应选择 B。

二、判断题

1. 消除递归不一定需要使用栈,此说法是否正确。

答案:正确。

分析:该题目主要考查栈在递归中的应用。对于尾递归可以将其转化成递推,不需要栈。所以这种说法是正确的。

尾递归是指一个递归函数的递归调用语句是递归函数的最后一句可执行语句。

例如,下面是一个输出数组元素的递归函数。

```
void RPrintArray(int list[],int n)
{
    if(n>=0)
    {
        printf("%d",list[n]);
        RPrintArray(list,--n);
    }
}
```

此程序是尾递归程序,消除尾递归很简单,只需首先计算新的 n 值,$n=n-1$,然后程序转到函数的开始处执行就行了,可以使用 while 语句来实现。

相应非递归函数如下:

```
void PrintArray(int list[],int n)
{
 while (n>=0)
     {
         printf("%d",list[n]);
         --n;
     }
 }
```

2. 栈和队列都是限制存取点的线性结构。

答案:正确。

分析：该题目主要考查栈和队列的定义。它们都只能在一端或两端存取数据的线性结构。见选择题第1题解答。

3. 入栈操作、出栈操作算法的时间复杂性均为 $O(n)$。

答案：错误。

分析：该题目主要考查栈的操作算法。入栈与出栈操作都在确定的栈顶位置进行，算法的时间复杂性均为 $O(1)$。所以这种说法是错误的。

4. 队列逻辑上是一个下端和上端既能增加又能减少的线性表。

答案：错误。

分析：该题目主要考查队列的逻辑结构和操作。队列是上端只能进行入队操作(即增加操作)，下端只能进行出队操作(即减少操作)。

5. 用I代表入栈操作，O代表出栈操作，栈的初始状态和栈的最终状态都为空，则使用栈的入栈和出栈可能出现的操纵序列为IOIOIOOI。

答案：错误。

分析：该题目主要考查栈的操作和“溢出”问题。在栈的操作中，保证出栈前提是栈中有元素，否则会造成栈的下溢，IOIOIOOI会出现下溢。

6. 任何一个递归过程都可以转换为非递归过程。

答案：正确。

分析：该题目主要考查栈在递归过程非递归化中的应用。任何一个递归过程都可以按照一定的步骤机械地转换为非递归过程。由于栈的后进先出特性吻合递归算法的执行过程，因而可以用非递归算法替代递归算法。递归过程转换为非递归过程的实质就是将递归中隐含的栈机制转化为由用户直接控制的栈，利用堆栈保存参数。

三、填空题

1. 已知一个栈s的输入序列为abcd，判断下面两个序列能否通过栈的Push_Stack和Pop_Stack操作输出：

(1) dbca　　(1)

(2) cbda　　(2)

答案：(1)不能；(2)能。

分析：该题目主要考查栈的操作。(1)不能。因为弹出d时，a、b、c均已依次压入栈，下一个弹出的元素只能是c而不是b。(2)能。Push_Stack (s,a)，Push_Stack (s,b)，Push_Stack (s,c)，Pop_Stack (s)，Pop_Stack (s)，Push_Stack (s,d)，Pop_Stack (s)，Pop_Stack (s)。

2. 对循环队列Q(设循环队列空间大小为MAXSIZE)，如何修改队头指针？ (1) ；如何修改队尾指针？ (2)

答案：(1)front=(front+1)% MAXSIZE；(2)rear=(rear+1)% MAXSIZE。

分析：该题目主要考查循环队列的操作。具体解答请参见选择题第11和12题。

3. 设长度为 n 的链队列用单循环链表表示，若只设头指针，则入队列出队列操作的时间分别为 (1) 和 (2) ；若只设尾指针，则入队列出队列操作的时间分别为

(3) 和 (4) 。

答案：(1) $O(n)$；(2) $O(n)$；(3) $O(1)$；(4) $O(1)$。

分析：该题目主要考查链队列和单循环链表的综合知识。当只设头指针时(见图 3.3(a))，入队相当于在 a_n 结点之后执行结点插入操作。根据链表的插入操作特点可知，插入操作前必须知道结点 a_1 和 a_n 的地址，获取结点 a_n 的地址的时间复杂度为$O(n)$；出队则相当于删除结点 a_1 的操作，因此必须获取结点 a_n 的地址，同样其时间复杂度为 $O(n)$。若设置尾指针(见图 3.3(b))，入队相当于在结点 a_1 之前和 a_n 之后执行结点插入操作，通过 rear 和 rear－＞next 可以获得结点 a_n 和 a_1 的地址，则其时间复杂度为 $O(1)$；出队则相当于删除结点 a_1 的操作，同样其时间复杂度为 $O(1)$。

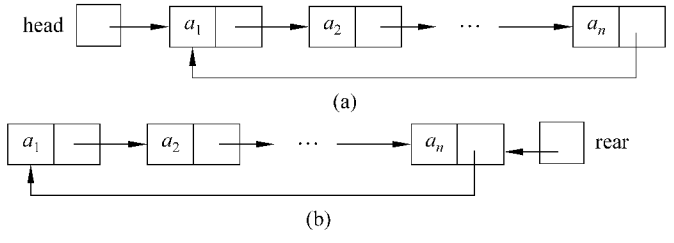

图 3.3　单循环链表表示的链队列示意图

4. 对于循环向量中的循环队列，写出求队列中元素个数的公式_____。

答案：(rear－front＋MAXSIZE)％MAXSIZE，其中 MAXSIZE 表示队列的存储空间。

分析：该题目主要考查循环队列的存储结构特点。

5. 向顺序栈插入新元素分为三步：第一步，进行 (1) 判断，判断条件是 (2) ；第二步是修改 (3) ；第三步把新元素赋给 (4) 。同样从顺序堆栈删除元素分为三步；第一步，进行 (5) 判断，判断条件是 (6) ；第二步是把 (7) 值返回；第三步 (8) 。

答案：(1)栈是否满；(2)s－＞top＝ MAXSIZE－1；(3)栈顶指针(top＋＋)；(4)栈顶对应的数组元素；(5)栈是否空；(6)s－＞top＝－1；(7)栈顶元素；(8)修改栈顶指针(top－－)。

分析：该题目是考虑栈的运算规则及其入、出栈的实现步骤。入栈时一般考虑判断栈满否，条件是是否超出最大空间。如果没有超出应该修改栈顶指针，然后将元素压入堆栈。出栈时，应首先考虑堆栈是否空。如果不空，先保留栈顶元素，然后修改栈顶指针。

6. 在将中缀表达式转换成后缀表达式和计算后缀表达式的算法中，都需要使用栈。对于前者，进入栈的元素为表达式中的 (1) ，而对于后者，进入栈的元素为 (2) 。中缀表达式$(a+b)/c-(f-d/e)$所对应的后缀表达式是 (3) 。

答案：(1)运算符；(2)操作数；(3)$ab+c/fde/--$。

分析：该题目主要考查栈的应用。中缀表达式就是将运算符写于参与运算的操作数的中间，操作数依原序排列。后缀表达式就是将运算符列于参与运算的操作数之后，操

作数的排列依原序。因此计算后缀表达式值的过程为：从左向右扫描后缀表达式，遇到操作数就进栈，遇到运算符就从栈中弹出两个操作数，执行该运算符所规定的运算，并将所得结果进栈。如此下去，直到表达式结束。所以对于计算后缀表达式，进栈的元素为操作数。

7. 假设以 S 和 X 分别表示入栈和出栈操作，则对输入序列 a、b、c、d、e 进行一系列栈操作 SSXSXSSXXX 之后，得到的输出序列为_____。

答案：bceda。

分析：该题目主要考查入栈和出栈操作。入栈和出栈操作只能在栈顶位置进行。根据操作序列，首先 a、b 进栈，然后 b 出栈；接着 c 进栈、c 出栈；d、e 相继进栈，栈顶元素为 e，最后 e、d、a 相继出栈。这样，得到出栈序列为 bceda。

四、应用题

1. 将整数 1、2、3、4 依次入栈或出栈，请回答下述问题：

(1) 当入、出栈次序为 Push_Stack(S,1)、Pop_Stack(S)、Push_Stack(S,2)、Push_Stack(S,x3)、Pop_Stack(S)、Push_Stack(S,4)、Pop_Stack(S)，出栈的数字序列是什么？

(2) 能否得到出栈序列 1423 和 1432？并说明为什么不能得到或者如何得到。

(3) 设有 n 个数据元素的序列顺序进栈，试给出可能输出序列的个数和不可能输出序列的个数。当 $n=4$(1、2、3、4)有 24 种排列，哪些序列是可以通过相应的入出栈操作得到的。

分析与解答：该题目主要考查栈的性质、结构和操作。

(1) 出栈序列为 1、3、4。

(2) 序列 1、4、2、3 不可能得到。因为 4 和 2 之间隔了 3，当 4 出栈后，栈顶元素是 3，而 2 在 3 的下面。序列 1、4、3、2 可以得到 Push_Stack(S,1)、Pop_Stack(S)、Push_Stack(S,2)、Push_Stack(S,3)、Push_Stack(S,4)、Pop_Stack(S)、Pop_Stack(S)、Pop_Stack(S)。

(3) 设有 n 个数据元素的序列(假设这个序列为 1,2,3,4,5,…,n)顺序进栈，那么输出序列个数 $f(n)$可以递推求出：为讨论方便，设 $n=0$，$f(0)=1$，当：

$n=1$ 时，显然 $f(1)=1$。

$n=2$ 时，容易得知 $f(2)=2$。

$n=3$ 时，1 最后出栈的序列有 $f(2)$种，2 最后出栈的序列有 $f(1)\times f(1)$种，3 最后出栈的序列有 $f(2)$种，所以 $f(3)=2\times f(2)+f(1)\times f(1)=5$。

$n=4$ 时，1 最后出栈的序列有 $f(3)$种，2 最后出栈的序列有 $f(1)\times f(2)$种，3 最后出栈的序列有 $f(2)\times f(1)$种，4 最后出栈的序列有 $f(3)$种，所以 $f(4)=2\times f(3)+ f(1)\times f(2)+f(2)\times f(1)=14$。

可以看出 $i(i=1,2,3,\cdots,n)$最后出栈的序列有 $f(i-1)\times f(n-i)$ 。

所以 $f(n)=f(0)\times f(n-1)+ f(1)\times f(n-2)+f(2)\times f(n-3)+f(3)\times f(n-4)+\cdots+f(n-1)\times f(0)$，用数学方法可得到：