第 4 章　串、数组和广义表

1．选择题

（1）串是一种特殊的线性表，其特殊性体现在（　）。

A．可以顺序存储　　B．数据元素是一个字符

C．可以链式存储　　D．数据元素可以是多个字符若

答案：B

（2）串下面关于串的的叙述中，（　）是不正确的？

A．串是字符的有限序列　　B．空串是由空格构成的串

C．模式匹配是串的一种重要运算　D．串既可以采用顺序存储，也可以采用链式存储

答案：B

解释：空格常常是串的字符集合中的一个元素，有一个或多个空格组成的串成为空格串，零个字符的串成为空串，其长度为零。

（3）串“ababaaababaa”的 next 数组为（　）。

A．012345678999　B．012121111212　C．011234223456　D．0123012322345

答案：C

（4）串“ababaabab”的 nextval 为（　）。

A．010104101　　B．010102101　　C．010100011　　D．010101011

答案：A

（5）串的长度是指（　）。

A．串中所含不同字母的个数　　B．串中所含字符的个数

C．串中所含不同字符的个数　　D．串中所含非空格字符的个数

答案：B

解释：串中字符的数目称为串的长度。

（6）假设以行序为主序存储二维数组 A=array[1..100,1..100]，设每个数据元素占 2 个存储单元，基地址为 10，则 LOC[5,5]=（　）。

A．808　　B．818　　C．1010　　D．1020

答案：B

解释：以行序为主，则 LOC[5,5]=[（5-1）*100+（5-1）]*2+10=818。

（7）设有数组 A[i,j]，数组的每个元素长度为 3 字节，i 的值为 1 到 8，j 的值为 1 到 10，数组从内存首地址 BA 开始顺序存放，当用以列为主存放时，元素 A[5,8]的存储首地址为（　）。

A．BA+141　　B．BA+180　　C．BA+222　　D．BA+225

答案：B

解释：以列序为主，则 LOC[5,8]=[（8-1）*8+（5-1）]*3+BA=BA+180。

（8）设有一个 10 阶的对称矩阵 A，采用压缩存储方式，以行序为主存储，a_{11} 为第一元素，其存储地址为 1，每个元素占一个地址空间，则 a_{85} 的地址为（　）。

A．13　　B．32　　C．33　　D．40

答案：C

（9）若对 n 阶对称矩阵 A 以行序为主序方式将其下三角形的元素(包括主对角线上所有元素)依次存放于一维数组 B[1..(n(n+1))/2]中，则在 B 中确定 a_{ij}（i<j）的位置 k 的关系为（　）。

A．i*(i-1)/2+j　　B．j*(j-1)/2+i　　C．i*(i+1)/2+j　　D．j*(j+1)/2+i

答案：B

（10）二维数组 A 的每个元素是由 10 个字符组成的串，其行下标 i=0, 1, …, 8, 列下标 j=1, 2, …, 10。若 A 按行先存储，元素 A[8, 5]的起始地址与当 A 按列先存储时的元素（　　）的起始地址相同。设每个字符占一个字节。

A．A[8,5]　　B．A[3,10]　　C. A[5,8]　　D．A[0,9]

答案：B

解释：设数组从内存首地址 M 开始顺序存放，若数组按行先存储，元素 A[8, 5]的起始地址为：M+[（8-0）*10+（5-1）]*1=M+84；若数组按列先存储，易计算出元素 A[3, 10]的起始地址为：M+[（10-1）*9+（3-0）]*1=M+84。故选 B。

（11）设二维数组 A[1.. m，1.. n]（即 m 行 n 列）按行存储在数组 B[1.. m*n]中，则二维数组元素 A[i,j]在一维数组 B 中的下标为（　）。

A．(i-1)*n+j　　B．(i-1)*n+j-1　　C．i*(j-1)　　D．j*m+i-1

答案：A

解释：特殊值法。取 i=j=1，易知 A[1,1]的的下标为 1，四个选项中仅有 A 选项能确定的值为 1，故选 A。

（12）数组 A[0..4,-1..-3,5..7]中含有元素的个数（　）。

A．55　　B．45　　C．36　　D．16

答案：B

解释：共有 5*3*3=45 个元素。

（13）广义表 A=(a,b,(c,d),(e,(f,g)))，则 Head(Tail(Head(Tail(Tail(A)))))的值为（　）。

A．(g)　　B．(d)　　C．c　　D．d

答案：D

解释：Tail(A)=(b,(c,d),(e,(f,g)))；Tail(Tail(A))=((c,d),(e,(f,g)))； Head(Tail(Tail(A)))= (c,d)；Tail(Head(Tail(Tail(A))))=(d)；Head(Tail(Head(Tail(Tail(A)))))=d。

（14）广义表((a,b,c,d))的表头是（　），表尾是（　）。

A．a　　B．( )　　C．(a,b,c,d)　　D．(b,c,d)

答案：C、B

解释：表头为非空广义表的第一个元素，可以是一个单原子，也可以是一个子表，((a,b,c,d))的表头为一个子表(a,b,c,d)；表尾为除去表头之外，由其余元素构成的表，表为一定是个广义表，((a,b,c,d))的表尾为空表( )。

（15）设广义表 L=((a,b,c))，则 L 的长度和深度分别为（　）。

A．1 和 1　　B．1 和 3　　C．1 和 2　　D．2 和 3

答案：C

解释：广义表的深度是指广义表中展开后所含括号的层数，广义表的长度是指广义表中所含元素的个数。根据定义易知 L 的长度为 1，深度为 2。

2．应用题

（1）已知模式串 t=‘abcaabbabcab’写出用 KMP 法求得的每个字符对应的 next 和 nextval 函数值。

答案：

模式串 t 的 next 和 nextval 值如下：

j	1	2	3	4	5	6	7	8	9	10	11	12
t 串	a	b	c	a	a	b	b	a	b	c	a	b
next[j]	0	1	1	1	2	2	3	1	2	3	4	5
nextval[j]	0	1	1	0	2	1	3	0	1	1	0	5

（2）设目标为 t=“abcaabbabcabaacbacba”,模式为 p=“abcabaa”

① 计算模式 p 的 naxtval 函数值；

② 不写出算法,只画出利用 KMP 算法进行模式匹配时每一趟的匹配过程。

答案：

① p 的 nextval 函数值为 0110132。（p 的 next 函数值为 0111232）。

② 利用 KMP(改进的 nextval)算法，每趟匹配过程如下：

第一趟匹配： abcaabbabcabaacbacba
abcab(i=5,j=5)

第二趟匹配： abcaabbabcabaacbacba
abc(i=7,j=3)

第三趟匹配： abcaabbabcabaacbacba
a(i=7,j=1)

第四趟匹配： abcaabbabcabaac bacba
（成功） abcabaa(i=15,j=8)

（3）数组 A 中，每个元素 A[i,j]的长度均为 32 个二进位,行下标从-1 到 9，列下标从 1 到 11，从首地址 S 开始连续存放主存储器中，主存储器字长为 16 位。求：

① 存放该数组所需多少单元？

② 存放数组第 4 列所有元素至少需多少单元？

③ 数组按行存放时，元素 A[7,4]的起始地址是多少？

④ 数组按列存放时，元素 A[4,7]的起始地址是多少？

答案：

每个元素 32 个二进制位，主存字长 16 位，故每个元素占 2 个字长，行下标可平移至 1 到 11。

（1）242 （2）22 （3）s+182 （4）s+142

(4)请将香蕉 banana 用工具 H( )—Head( )，T( )—Tail( )从 L 中取出。

L=(apple,(orange,(strawberry,(banana)),peach),pear)

答案：H（H（T（H（T（H（T（L））））））

3．算法设计题

（1）写一个算法统计在输入字符串中各个不同字符出现的频度并将结果存入文件（字符串中的合法字符为 A-Z 这 26 个字母和 0-9 这 10 个数字）。

[题目分析] 由于字母共 26 个，加上数字符号 10 个共 36 个，所以设一长 36 的整型数组，前 10 个分量存放数字字符出现的次数，余下存放字母出现的次数。从字符串中读出数字字符时，字符的 ASCII 代码值减去数字字符 ‘0’ 的 ASCII 代码值，得出其数值(0..9)，字母的 ASCII 代码值减去字符‘A’的 ASCII 代码值加上 10，存入其数组的对应下标分量中。遇其它符号不作处理，直至输入字符串结束。

[算法描述]

```
void Count（）
//统计输入字符串中数字字符和字母字符的个数。
 {int i，num[36];
  char ch;
```

```
for（i＝0；i<36；i++）num[i]＝0；// 初始化
while（(ch＝getchar（)) != ‘#’）  // ‘#’表示输入字符串结束。
    if（‘0’ <=ch<= ‘9’）{i=ch－48;num[i]++;}          // 数字字符
    else  if（‘A’ <=ch<= ‘Z’）{i=ch-65+10;num[i]++;} // 字母字符
for（i=0；i<10；i++）  // 输出数字字符的个数
  cout<<“数字”<<i<<  “的个数=”<<num[i]<<endl;
for（i＝10；i<36；i++）// 求出字母字符的个数
  cout<<“字母字符”<<i+55<<  “的个数=”<<num[i]<<endl;
}
```

（2）写一个递归算法来实现字符串逆序存储，要求不另设串存储空间。

[题目分析]实现字符串的逆置并不难，但本题“要求不另设串存储空间”来实现字符串逆序存储，即第一个输入的字符最后存储，最后输入的字符先存储，使用递归可容易做到。

[算法描述]

```
void InvertStore(char A[])
//字符串逆序存储的递归算法。
{char ch;
 static int i = 0;//需要使用静态变量
 cin>>ch;
 if (ch!= '.')    //规定'.'是字符串输入结束标志
    {InvertStore(A);
     A[i++] = ch;//字符串逆序存储
    }
 A[i] = '\0';  //字符串结尾标记
 }
```

（3）编写算法，实现下面函数的功能。函数 void insert(char*s,char*t,int pos)将字符串 t 插入到字符串 s 中，插入位置为 pos。假设分配给字符串 s 的空间足够让字符串 t 插入。（说明：不得使用任何库函数）

[题目分析]本题是字符串的插入问题，要求在字符串 s 的 pos 位置，插入字符串 t。首先应查找字符串 s 的 pos 位置，将第 pos 个字符到字符串 s 尾的子串向后移动字符串 t 的长度，然后将字符串 t 复制到字符串 s 的第 pos 位置后。

对插入位置 pos 要验证其合法性，小于 1 或大于串 s 的长度均为非法，因题目假设给字符串 s 的空间足够大，故对插入不必判溢出。

[算法描述]

```
void  insert(char *s,char *t,int pos)
//将字符串 t 插入字符串 s 的第 pos 个位置。
{int i=1,x=0;  char *p=s,*q=t;  //p，q 分别为字符串 s 和 t 的工作指针
 if(pos<1) {cout<<“pos 参数位置非法”<<endl;exit(0);}
 while(*p!=' \0' &&i<pos) {p++;i++;} //查 pos 位置
 //若 pos 小于串 s 长度，则查到 pos 位置时，i=pos。
 if(*p == '/0') { cout<<pos<<"位置大于字符串 s 的长度";exit(0);}
 else      //查找字符串的尾
   while(*p!='/0') {p++; i++;}  //查到尾时，i 为字符‘\0’的下标，p 也指向‘\0’。
```

```
while(*q!= '\0') {q++; x++; }    //查找字符串 t 的长度 x，循环结束时 q 指向'\0'。
for(j=i;j>=pos ;j--){*(p+x)=*p; p--;}//串 s 的 pos 后的子串右移，空出串 t 的位置。
q--;  //指针 q 回退到串 t 的最后一个字符
for(j=1;j<=x;j++) *p--=*q--;  //将 t 串插入到 s 的 pos 位置上
```

[算法讨论] 串 s 的结束标记('\0')也后移了，而串 t 的结尾标记不应插入到 s 中。

（4）已知字符串 S1 中存放一段英文，写出算法 format(s1,s2,s3,n),将其按给定的长度 n 格式化成两端对齐的字符串 S2，其多余的字符送 S3。

[题目分析]本题要求字符串 s1 拆分成字符串 s2 和字符串 s3，要求字符串 s2“按给定长度 n 格式化成两端对齐的字符串”，即长度为 n 且首尾字符不得为空格字符。算法从左到右扫描字符串 s1，找到第一个非空格字符，计数到 n，第 n 个拷入字符串 s2 的字符不得为空格，然后将余下字符复制到字符串 s3 中。

[算法描述]

```
void format (char *s1,*s2,*s3)
//将字符串 s1 拆分成字符串 s2 和字符串 s3，要求字符串 s2 是长 n 且两端对齐
{char *p=s1, *q=s2;
 int i=0;
    while(*p!= '\0' && *p== ' ') p++;//滤掉 s1 左端空格
    if(*p== '\0') {cout<<"字符串 s1 为空串或空格串"<<endl;exit(0);  }
    while( *p!='\0' && i<n){*q=*p; q++; p++; i++;}
    //字符串 s1 向字符串 s2 中复制
    if(*p =='\0'){cout<<"字符串 s1 没有"<<n<<"个有效字符"<<endl; exit(0);}
    if(*(--q)==' ' ) //若最后一个字符为空格，则需向后找到第一个非空格字符
     {p-- ;           //p 指针也后退
      while(*p==' '&&*p!='\0') p++;//往后查找一个非空格字符作串 s2 的尾字符
      if(*p=='\0')
        {cout<<"s1 串没有"<<n<<"个两端对齐的字符串"<<endl; exit(0);}
      *q=*p;          //字符串 s2 最后一个非空字符
      *(++q)='\0';    //置 s2 字符串结束标记
      }
    *q=s3;p++;       //将 s1 串其余部分送字符串 s3。
    while (*p!= '\0') {*q=*p; q++; p++;}
    *q='\0';         //置串 s3 结束标记
 }
```

（5）设二维数组 a[1..m, 1..n] 含有 m*n 个整数。

① 写一个算法判断 a 中所有元素是否互不相同?输出相关信息(yes/no)；

② 试分析算法的时间复杂度。

①

[题目分析]判断二维数组中元素是否互不相同，只有逐个比较,找到一对相等的元素，就可结论为不是互不相同。如何达到每个元素同其它元素比较一次且只一次？在当前行，每个元素要同本行后面的元素比较一次（下面第一个循环控制变量 p 的 for 循环），然后同第 i+1 行及以后各行元素比较一次，这就是循环控制变量 k 和 p 的二层 for 循环。

[算法描述]

```
int JudgEqual(ing a[m][n],int m,n)
 //判断二维数组中所有元素是否互不相同，如是，返回 1；否则，返回 0。
{for(i=0;i<m;i++)
  for(j=0;j<n-1;j++)
   {for(p=j+1;p<n;p++) //和同行其它元素比较
     if(a[i][j]==a[i][p]) {cout<<“no”; return(0); }
     //只要有一个相同的，就结论不是互不相同
     for(k=i+1;k<m;k++)  //和第 i+1 行及以后元素比较
      for(p=0;p<n;p++)
       if(a[i][j]==a[k][p]) { cout<<“no”; return(0); }
   }// for(j=0;j<n-1;j++)
cout<<“yes”; return(1);   //元素互不相同
}//算法 JudgEqual 结束
```

②二维数组中的每一个元素同其它元素都比较一次，数组中共 m*n 个元素，第 1 个元素同其它 m*n-1 个元素比较，第 2 个元素同其它 m*n-2 个元素比较，……，第 m*n-1 个元素同最后一个元素(m*n)比较一次,所以在元素互不相等时总的比较次数为 (m*n-1)+(m*n-2)+…+2+1=（m*n）(m*n-1)/2。在有相同元素时,可能第一次比较就相同,也可能最后一次比较时相同,设在(m*n-1)个位置上均可能相同,这时的平均比较次数约为（m*n）(m*n-1)/4，总的时间复杂度是 O(n^4)。

(6)设任意 n 个整数存放于数组 A(1:n)中，试编写算法，将所有正数排在所有负数前面（要求算法复杂度为 O(n)）。

[题目分析]本题属于排序问题，只是排出正负，不排出大小。可在数组首尾设两个指针 i 和 j，i 自小至大搜索到负数停止，j 自大至小搜索到正数停止。然后 i 和 j 所指数据交换，继续以上过程，直到 i=j 为止。

[算法描述]

```
void Arrange(int A[],int n)
//n 个整数存于数组 A 中，本算法将数组中所有正数排在所有负数的前面
{int i=0,j=n-1,x;  //用类 C 编写，数组下标从 0 开始
 while(i<j)
  {while(i<j && A[i]>0)  i++;
   while(i<j && A[j]<0)  j--;
   if(i<j) {x=A[i]; A[i++]=A[j]; A[j--]=x; }//交换 A[i] 与 A[j]
  }// while(i<j)
 }//算法 Arrange 结束.
```

[算法讨论]对数组中元素各比较一次，比较次数为 n。最佳情况(已排好,正数在前,负数在后)不发生交换，最差情况(负数均在正数前面)发生 n/2 次交换。用类 c 编写，数组界偶是 0..n-1。空间复杂度为 O(1).