第四章　数　　组

【教学目的】

掌握一维数组的定义、初始化和引用。
掌握二维数组的定义、初始化和引用。
掌握字符数组的定义、初始化和引用。

【教学内容】

如何定义一维、二维数组，正确理解数组的存储形式，在什么样的实际问题中该使用数组结构。

正确引用数组元素，理解下标所代表的含义，掌握数组和循环的结合使用。

字符数组的初始化、输入与输出，字符数组的应用。

【教学重点和难点】

定义数组，正确理解数组名代表数组的首地址，通过数组名（首地址）结合下标，可以任意访问数组中的任一元素。

字符串处理函数的使用，特别是函数参数的正确使用。

【问题的提出】

在介绍数组的使用之前，首先看一个例子。

【例】将从键盘输入的 20 个数逆序输出。

根据前面所学的方法，可得程序如下：

```
#include "stdio.h"
main ( )
  {float a1,a2,a3,a4,a5,……,a19,a20;          /* 定义 20 个变量 */
printf ("请输入 20 个实数： ");
   scanf ("%f",&a1);             /* 连续 20 个输入函数语句 */
scanf ("%f",&a2);
……
scanf ("%f",&a20);
printf ("反序输出为： ",);
   printf ("%f ",a20);           /*连续 20 个输出函数语句 */
printf ("%f ",a19);
……
printf("%f ",a1);
  }
```

分析：这样的程序是无法接受的，因为基本数据类型，它们通常用于解决一些简单的问题，输入和输出的数据也是少量的。若输入的数据增加到 2000 个，在一个程序中要保存 2000 个实型数据需要定义 2000 个变量，若在各类计算机语言中仅有简单变量，是没有办法解决复杂问题的。

试问：要存放大量类型相同的数据，该如何处理？程序如何写？如在数学问题中有一个10 行 10 列的矩阵，该怎样存储？有 50 个字符串又该如何处理？

方法：为了解决复杂问题，C 语言中提供了构造类型的数据，本章中将介绍用数组解决这些问题的方法。

【教学要点】

1. 一维数组的定义、初始化和引用

【例 4.1】由键盘输入 20 个数，逆序输出。

分析：在程序设计中，数组是十分有用的数据类型，是一组具有相同类型的变量，用一个数组名标识，其中每个变量(称为数组元素)通过该变量在数组中的相对位置(称为下标)来引用。要引用每个数组元素需要使用循环结构控制元素的下标，本程序采用数组和循环相结合，不仅书写简洁，而且通用性强。若输入的数据个数不是 20，而是 2000，所做的工作只是把符号常量 N 的值改为 2000 而已。

源程序如下：

```
#define N 20          /* 定义符号常量 N 为 20 */
#include "stdio.h"
main ( )
{ int a[N], k;              /*定义 a 数组，长度为 20；数组要先定义后引用*/
printf(“input %d data:”,N);
for ( k=0; k<N;k++)
scanf ("%d",&a[k]);
/*数组元素的下标从 0 开始，最后一个元素的下标为 N-1，通过 for()循环实现依次的变化*/
for ( k=N-1; k>=0; k-- )   /*   数组最后一个元素的下标是 N-1     */
printf ("%d ",a[k]);
}
```

【例 4.2】求 Fibonacci 数列前 15 个数。这个数列有如下特点：第 1，2 两个数为 0，1。从第 3 个数开始，该数是其前面两个连续数之和，即 a[i]=a[i-1]+a[i-2]。

分析：15 个数用一个一维数组 a 来处理，该数组最少可以存储 15 个数。已知 a[0]=0, a[1]=1；后 13 个数用一重 for()循环求出。

源程序如下：

```
main()
{   int a[15]={0,1};        /*定义 a 数组同时赋初始值，使 a[0]=0, a[1]=1 */
    int i;
clrscr();
    for(i=2;i<15;i++)          /*求出第 3 个数到第 15 个数的值*/
        a[i]=a[i-1]+a[i-2];
    printf("\n result is:     ");
    for(i=0;i<15;i++) /*输出第 1 个数到第 15 个数的值，每个数占 6 个字符的宽度*/
        printf("%6d",a[i]);
    getch( );       /*  暂停函数，便于观察结果  */
}
```

说明：

1. 在本例中需注意数组赋初始值的问题。动态数组可以部分初始化，如本例中的 int

a[15]={0,1}，其余 13 个数在 TC 系统中将赋初值 0，但对于在定义后进行初始化的动态数组不遵循该原则；如果不对动态数组赋初值，数组中存放的将是内存单元中的随机值。因此在使用动态数组之前，必须要赋初值，否则会得到不可想象的结果。如果对全部数组元素赋初值时，可以不指定数组的长度，系统将根据大括号中数字的个数自动确定数组的长度。如：int a[ ]= {10，20，30，40，50};系统自动确定数组的长度为 5。

2. 本例中的两个 for()循环是顺序结构的循环。第一个循环求出第 3 个数到第 15 个数的值，存放到对应的数组元素中，第二个循环是输出第 1 个数到第 15 个数的值，不存在包含关系。若想只用一个循环实现，可用以下程序段：

```
printf("%6d%6d",a[0],a[1]);        /*先输出第 1、2 个数*/
for(i=2;i<15;i++)          /*依次求出第 3 个数到第 15 个数的值,并立即输出*/
  { a[i]=a[i-1]+a[i-2];
    printf("%6d",a[i]);
    }
```

【例 4.3】编写程序，从键盘输入 N 个数，用选择排序法按从小到大的顺序重新排列后输出，并输出每趟排序的结果。

分析：选择排序的基本思想是：第一趟排序是在无序的数{ r_1,r_2,r_3……r_n}中选出最小的元素，将它与 r_1 交换；第二趟排序是在无序的数{ r_2,r_3……r_n}中选出最小的元素，将它与 r_2 交换；……，第 i 趟排序是在无序的数{ r_i,r_{i+1}, ……r_n}中选出最小的元素，将它与 r_i 交换；直到第 n-1 趟排序后，整个数据元素就递增有序。

程序实现：

1. 从键盘输入 N 个数：需定义 1 个一维数组，用一重 for()循环实现。
2. 选择排序需用两重 for()循环实现，外循环控制比较趟数，内循环进行每趟比较, 比较时，采用 if（）语句实现。找出每趟的最小值,用一个变量记录下该趟最小值元素的下标，然后将最小值元素与该组第一个元素进行交换。
3. 输出排序后的数据：用一重 for( )循环实现。

源程序如下：

```
#include<stdio.h>
#define N 8
main( )
{ int a[N];          /*  定义整型数组 a ,最多可存储 8 个整数    */
  int i,j,t,k;
  printf("输入%d 个待排序的数：",N);
  for(i=0;i<N;i++)            /*  输入第 1 个数到第 N 个数的值  */
        scanf("%d",&a[i]);
  for(i=0;i<N-1;i++)         /*  外循环：控制比较趟数  */
  {   k=i;     /*  用变量 k 记下待选择组的首元素的下标，作为最小元素的初始下标    */
  for(j=i+1;j<N;j++)         /*  内循环：进行每趟比较次数  */
        if(a[k]>a[j])   k=j;         /*  比较时，用 k 记录下该趟最小值元素的下标  */
    if(k!=i)   /*  一趟比较完成后，将该趟最小值元素 a[k]与该趟的首元素 a[i]交换  */
        {t=a[k]; a[k]=a[i]; a[i]=t;}     /*    元素交换      */
   printf("\n 第%d 趟的排序结果为：",i+1);
   for(j=0;j<N;j++)
       printf("%5d ",a[j]);
}
   getch();
```

```
 }
```
运行结果:

输入 8 个待排序的数：80　92　76　63　83　89　72　69
第 1 趟的排序结果为：63　92　76　80　83　89　72　69
第 2 趟的排序结果为：63　69　76　80　83　89　72　92
第 3 趟的排序结果为：63　69　72　80　83　89　76　92
第 4 趟的排序结果为：63　69　72　76　83　89　80　92
第 5 趟的排序结果为：63　69　72　76　80　89　83　92
第 6 趟的排序结果为：63　69　72　76　80　83　89　92
第 7 趟的排序结果为：63　69　72　76　80　83　89　92

说明：本例中输出了每一趟的排序结果，若只需输出最终结果，将实现输出的 for()循环放到实现排序的两重 for()循环之后，不存在包含关系。在编写程序时一定要理清楚每一个循环应该要实现怎样的功能，包含哪些语句。

2. 二维数组的定义、初始化和引用

【例 4.4】矩阵的转置。有如下矩阵 A，经过转置后结果放在矩阵 B 并输出。

$$A=\begin{vmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{vmatrix}$$

分析：当数组元素具有两个下标时，该数组称为二维数组。矩阵可以看作具有行和列的数据结构，用二维数组实现。矩阵的转置问题就转换为一个二维数组行和列元素的互换，存到另一个二维数组中去。

源程序如下：

```
main(     )
{ int a[3][2]={1,2,3,4,5,6};
  int b[2][3],i,j;
  clrscr( );
  printf("array    a:\n");
  for(i=0;i<=2;i++)                  /*    控制行      */
    { for(j=0;j<=1;j++)              /*    控制列      */
        { printf("%5d",a[i][j]);     /*     输出转置前的数组元素    */
          b[j][i]=a[i][j];           /*    转置赋值    */
        }
       printf("\n");         /*      一行输出后换行      */
     }
  printf("array b:\n");
  for(i=0;i<=1;i++)          /*    按行输出转置后的数组 B 的元素    */
     { for(j=0;j<=2;j++)
       printf("%5d",b[i][j]);
  printf("\n");        /*      一行输出后换行      */

  }
}
```

运行结果:

```
array a:
1    2
3    4
5    6
array b:
1    3    5
2    4    6
```

说明：

1. 定义了 int 型数组 a，编译程序将为 a 数组在内存中开辟 3*2=6 个连续的存储单元，用来存放 a 数组的 6 个数组元素。存储方式为按行顺序存放。数组名 a 表示 a 数组的首地址，是地址常量。
2. 在 C 语言中，二维数组 a 的每一行都可以看作一维数组，用 a[i]表示第 i 行构成的一维数组的数组名。二维数组 a 有三个数组元素 a[0]、a[1]、a[2]，而 a[0]、a[1]、a[2]均是包含 2 个元素的一维数组，即二维数组中的一个特殊元素表示一个一维数组名，是该行的首地址，同样是常量。

【例 4.5】计算两个矩阵 A、B 的乘积。

分析：两个矩阵的乘积仍然是矩阵。若 A 矩阵有 m 行 w 列，B 矩阵有 w 行 n 列,则它们的乘积 C 矩阵有 m 行 n 列。矩阵乘法遵循以下规则：

$$C_{ij} = A_{i0} * B_{0j} + A_{i1} * B_{1j} + \ldots + A_{ik} * B_{kj} \quad (i=0,1,\ldots\ldots,m-1;j=0,1,\ldots\ldots,n-1)$$

假设 A、B、C 矩阵用 3 个 2 维数组表示：a 数组有 3 行 2 列，b 数组有 2 行 3 列，则 c 数组有 3 行 3 列。如：

```
c[0][0]= a[0][0]*b[0][0]+a[0][1]*b[1][0];
c[1][0]= a[1][0]*b[0][1]+a[1][1]*b[1][1];
```

从以上算法可以看出，需要 3 重循环（i、j、k）才能计算 C 矩阵的各元素。

源程序如下：

```
#define M 3
#define W 2
#define N 3
main( )
{ int i,j,k,t,a[M][W],b[W][N],c[M][N];
  printf("请输入 A 矩阵元素（%d 行%d 列）:\n"，M,W);
  for(i=0;i<M;i++)       /*   控制行     */
    for(j=0;j<W;j++)     /*   控制列     */
       scanf("%d",&a[i][j]);
  printf("请输入 B 矩阵元素（%d 行%d 列）:\n"，W,N)；
  for(i=0;i<W;i++)
    for(j=0;j<N;j++)
       scanf("%d",&b[i][j]);
  for(i=0;i<M;i++)                  /*  三重循环计算矩阵 C*/
    for(j=0;j<N;j++)
       { for(k=t=0;k<W;k++)
              t+ = a[i][k]*b[k][j];
         c[i][j]=t;
       }
  for(i=0;i<M;i++)              /*  两重循环输出矩阵 C*/
    {for(j=0;j<N;j++)
```

```
            printf("%5d",c[i][j]);
          printf("\n");
        }
    }
```

【例 4.6】在一个 3 行 4 列的矩阵中查找指定数据，并输出该数据及其在矩阵中位置。

分析：首先用二维数组 a 表示矩阵，然后输入任意一个整数，并在 a 矩阵中搜索该数(用两重循环实现);该数可能不止一个,若在 i 行 j 列处找到该数，则输出该数及其位置（下标 i 和 j）；否则输出找不到的提示信息。

源程序如下：

```
#include "stdio.h"
main( )
{ int i,j,k, flag=0;
  int a[][4]={{1,2,3,4},{-5,7,12,6},{13,4,19,-8}};
  /*定义数组时对全部元素赋初值，可省略第 1 维的长度,第 2 维的长度不能省略*/
  printf ("输入要查找的整数:");
  scanf ("%d",&k);
  for (i=0;i<3;i++)
  { for(j=0;j<4;j++)
      if(a[i][j]==k)
      { printf("%d 为第%d 行第%d 列的元素。\n",k,i,j);
        flag =1;              /*   标志量 flag =1，找到 k   */
      }
    }
  if(flag==0) printf("%d not found!\n",k);     /*   flag =0，未找到 k   */
  }
```

3. 字符数组的定义、引用和字符串处理函数的使用

【例 4.7】不使用字符串比较函数 strcmp，实现两个字符串 s1、s2 的比较。

分析：在 C 语言中，一个一维字符数组存放一个字符串；串的比较是两个字符数组对应下标的元素按 ASCII 码值进行比较，若元素 s1[i]>s2[i],则串 s1>s2；若 s1[i]<s2[i],则 s1>s2, 若相等继续比较下一对字符，直到出现'\0', 仅当两个字符串完全一样时，才能确定它们相等。

源程序如下：

```
#include "stdio.h"
main( )
{ char s1[20],s2[20];
  int i=0,k;
  printf("输入两个长度小于 20 的字符串 s1,s2:");
  gets(s1);          /*用 gets()函数整体输入字符串，可以包含空格符，以回车键结束   */
  gets(s2);
  while(s1[i]&&s2[i])       /*   比较每一对字符，直到遇到'\0'退出循环     */
   {if(s1[i]!=s2[i])          /*   若某一对字符不同，分出大小，强制退出循环     */
      break;
    else
      i++;             /*     若相等继续比较下一对字符     */
    }
```

```
    k=s1[i]-s2[i];
    if(k==0)
        printf("%s=%s\n",s1,s2);
     else if(k>0)
        printf("%s>%s\n",s1,s2);
    else
        printf("%s<%s\n",s1,s2);
}
```

运行程序：

输入两个长度小于 20 的字符串 s1,s2：abcdef123
abcsgy

输出：abcdef123< abcsgy

说明：

1. 在本例中需注意字符串的输入问题。用 gets()函数整体输入字符串，可以包含空格符，以回车键结束；函数括号中的参数常用数组名表示存放字符串的首地址。也可用 scanf("%s%s",s1,s2); 整体输入字符串。但两者的区别是 scanf()函数不能输入带空格的字符串，空格符与回车键同样表示结束,一次可输入多个串；gets()函数一次只能输入一个串，无需格式说明符。
2. 本例中还需注意字符串的输出问题。如 printf("%s>%s\n",s1,s2); 用 printf()函数输出要使用格式说明符，一次可输出多个串；也可用 puts()函数，但一次只能输出一个串，无需格式说明符，如 puts(s1);。

【例 4.8】判断 s1 字符串中是否包含 s2 字符串。

分析：从 s1 字符串的第一个字符开始，依次与 s2 字符串的各字符比较，若均相同，则 s1 包含 s2。否则再从 s1 的下一个字符（第 2 个字符）开始，依次与 s2 字符串的各字符比较，……。设 k1，k2 分别表示 s1 串和 s2 串的长度，则最后一次应从 s1 的第 k1-k2+1 个字符开始（即 s1[k1-k2]），依次与 s2 字符串的各字符比较，若存在不同字符，则 s1 肯定不包含 s2。采用标志变量 flag =1,s2 包含在 s1 中，flag =0, s2 不包含在 s1 中 。

源程序如下：

```
#include "string.h"
main(    )
{ char s1[30],s2[30];
   int i,j,k,k1,k2,flag=0;
   gets(s1); gets(s2);
   k1=strlen(s1); k2=strlen(s2);        /*     求两个串的长度       */
   for (i=0;i<k1-k2+1&&!flag;i++)
      { for (j=0,k=i; s1[k]==s2[j];k++,j++)       /*      存在不同字符时退出循环       */
          if (s2[j+1]=='\0')
          { flag=1;break; }              /*      s2 包含在 s1 中时退出循环       */
      }
   if (flag ==1 ) printf ("%s is in %s\n",s2,s1);
   else printf ("%s is not in %s\n",s2,s1);
}
```

运行程序：

输入：I am a boy
am

输出：am is in I am a boy

说明：在本例中字符串长度的求得采用了 strlen( )函数，其参数常用数组名表示存放字符串的首地址，求出从该首地址开始直到遇到'\0'结束标志的字符个数。

【例 4.9】5 个学生按姓名从小到大排序。

分析：二维字符数组的每一行可以看作一维字符数组，即二维字符数组的每一行可以存放一个字符串，因此可以利用二维字符数组存放多个字符串。建立二维字符数组 s[5][20],存放输入的 5 个姓名（字符串）。再建立一个一维数组 a[5],存放各字符串在 s 数组中的行号。排序时，需要进行字符串交换位置时，并不交换它们在 s 数组中位置，而是交换 a 数组中相应的行号。最后输出排序结果时，按照 a 数组元素的顺序，分别输出相应行号的字符串。排序前，a 数组中存放 s 数组各行字符串的行号，排序后，根据各行字符串的大小，从小到大地重排它们在 a 数组中的行号。

源程序如下：

```
#include "stdio.h"
#include "string.h"
main( )
{ char s[5][20],a[5];
   int i,j,k,t;
   clrscr();
   for(i=0;i<5;i++)        /* 输入各字符串，s[i]表示第 i 行的首地址 */
   { printf("input name s[%d]:",i);
      scanf("%s",s[i]);    /*用 scanf ()函数整体输入字符串，用空格符或回车键表示结束*/
      a[i]=i;              /* a[i]记录各字符串在 s 数组中的行号 i   */
      }
   for(i=0;i<4;i++)       /* 选择法从小到大排序 */
   { k=i;
      for(j=i+1;j<5;j++)
         if(strcmp(s[a[k]],s[a[j]])>0) k=j;      /*比较，记录当前最小串的行号 */
      if(k!=i)
         {   t=a[k]; a[k]=a[i]; a[i]=t; }   /*最小串的行号 a[k]与 a[i]交换位置*/
      }
   printf("result is:\n");
   for(i=0;i<5;i++)         /* 按排序后的行号 a[i]输出各字符串 */
      puts(s[a[i]]);
   }
```

说明：在本例中字符串的比较采用了 strcmp()函数，其参数常是两个数组名，表示要比较的两个字符串从首地址开始，逐个进行字符的比较，直到遇到不同的字符或遇到'\0'为止。

【例 4.10】输入一串字符，统计其中有多少个单词。

分析：定义一个一维字符数组存放这串字符，从第一个元素到最后一个元素，依次用 if()语句判断（当前字符==空格），是（相等）：未出现新单词，使标志变量 word=0,记数变量 num 不累加；否（不相等）：判断前一字符为空格(word==0),则新单词出现, word=1,num 加 1；否则前一字符为非空格(word==1),未出现新单词,num 不变。

源程序如下：

```
#include <stdio.h>
main( )
{ char s[81];
   int i,num=0,word=0;
   printf("input a string: ");
```

```
    gets(s);
    for(i=0;s[i]!='\0';i++)   /* 串从第一个元素依次判断，直到遇到'\0'为止   */
        if(s[i]==' ')  word=0;
        else if(word==0)
          {    word=1;   num++;    }
    printf("There are %d words in the line.\n",num);
}
```

运行结果:

input a string: There is a book.
There are 4 words in the line.

【小结】

1、数组是同类型数据的集合。数组元素的数据类型可以是整型、实型、字符型、以及后面将介绍的指针型、结构型等。通过下标的变化可以引用任意一个数组元素。需要注意的是，不要进行下标越界的引用，那样会带来意外的副作用，比如会隐含地修改其它变量的值。

2、数组在数据处理和数值计算中有十分重要的作用，数组与循环结合，使很多问题的算法得以简单地表述，高效地实现。常用的是一维数组和二维数组。

3、C 语言中没有字符串变量。如何存储字符串？用一维字符数组存放一个字符串，各数组元素依次存放字符串的各字符。数组名代表该数组的首地址，这为引用整个字符串提供了极大的方便。处理多个字符串时（如字符串排序等），常用二维字符数组存放它们（每行存放一个字符串）。

4、需要注意的是，若定义的一维字符数组用来存放 n 个字符的字符串，则定义时数组元素的个数至少为 n+1，多出的一个数组元素存放字符串结束符'\0'。否则,字符串没有结束标志，处理字符串时可能会出现错误。

5、C 语言提供相应的库函数对字符串做操作（如字符串的连接、复制、比较、求长度等）。在 string.h 文件中对这些函数进行了定义，用户只要在程序的开头加上命令行 #include "string.h"，就可以调用它们完成相应的操作，但在 TC 系统可以省略，默认可以处理。

6、本章还介绍了一些有用的基本算法，如冒泡法排序和选择法排序、在数组中求最大值、最小值、查找、插入和删除数组元素的方法、矩阵的乘法、矩阵的转置等算法。