7 数组

在程序设计中，为了处理方便，把具有相同类型的若干变量按有序的形式组织起来。这些按序排列的同类数据元素的集合称为数组。

7.1 一维数组的定义和引用

7.1.1 一维数组的定义方式

在C语言中使用数组必须先进行定义。

一维数组的定义方式为：

类型说明符 数组名 [常量表达式]；

其中：

类型说明符是任一种基本数据类型或构造数据类型。

数组名是用户定义的数组标识符。

方括号中的常量表达式表示数据元素的个数，也称为数组的长度。

例如：

```
int a[10];              说明整型数组 a，有 10 个元素。
float b[10],c[20];      说明实型数组 b，有 10 个元素，实型数组 c，有 20 个元素。
char ch[20];            说明字符数组 ch，有 20 个元素。
```

对于数组类型说明应注意以下几点：

1) 数组的类型实际上是指数组元素的取值类型。对于同一个数组，其所有元素的数据类型都是相同的。
2) 数组名的书写规则应符合标识符的书写规定。
3) 数组名不能与其它变量名相同。

例如：

```
main()
{
  int a;
  float a[10];
  ……
}
```

是错误的。

4) 方括号中常量表达式表示数组元素的个数，如 a[5]表示数组 a 有 5 个元素。但是其下标从 0 开始计算。因此 5 个元素分别为 a[0],a[1],a[2],a[3],a[4]。
5) 允许在同一个类型说明中，说明多个数组和多个变量。

例如：

```
int a,b,c,d,k1[10],k2[20];
```

7.1.2 一维数组元素的引用

数组元素是组成数组的基本单元。数组元素也是一种变量， 其标识方法为数组名后跟一个下标。下标表示了元素在数组中的顺序号。

数组元素的一般形式为：

数组名[下标]

其中下标只能为整型常量或整型表达式。如为小数时，C 编译将自动取整。

例如：

```
a[5]
a[i+j]
a[i++]
```

都是合法的数组元素。

在 C 语言中只能逐个地使用下标变量，而不能一次引用整个数组。

例如，输出有 10 个元素的数组必须使用循环语句逐个输出各下标变量：

```
for(i=0; i<10; i++)
   printf("%d",a[i]);
```

而不能用一个语句输出整个数组。

下面的写法是错误的：

```
printf("%d",a);
```

【例 7.1】

```
main()
{
  int i,a[10];
  for(i=0;i<=9;i++)
    {
      a[i]=i;
    }
 for(i=9;i>=0;i--)
   {
      printf("%d ",a[i]);
   }
}
```



【例 7.2】

```
main()
{
  int i,a[10];
  for(i=0;i<10;)
      a[i++]=i;
  for(i=9;i>=0;i--)
      printf("%d",a[i]);
 }
```

7.1.1 一维数组的初始化

给数组赋值的方法除了用赋值语句对数组元素逐个赋值外， 还可采用初始化赋值和动态赋值的方法。

数组初始化赋值是指在数组定义时给数组元素赋予初值。数组初始化是在编译阶段进行的。这样将减少运行时间，提高效率。

初始化赋值的一般形式为：

类型说明符 数组名[常量表达式]={值，值……值}；

其中在{ }中的各数据值即为各元素的初值，各值之间用逗号间隔。

例如：

```
int a[10]={ 0,1,2,3,4,5,6,7,8,9 };
```

相当于 a[0]=0;a[1]=1...a[9]=9;

C 语言对数组的初始化赋值还有以下几点规定：

1) 可以只给部分元素赋初值。

当{ }中值的个数少于元素个数时，只 给前面部分元素赋值。

例如：

```
int a[10]={0,1,2,3,4};
```

表示只给 a[0]～a[4]5 个元素赋值，而后 5 个元素自动赋 0 值。

2) 只能给元素逐个赋值，不能给数组整体赋值。

例如给十个元素全部赋 1 值，只能写为：

```
int a[10]={1,1,1,1,1,1,1,1,1,1};
```

而不能写为：

```
int a[10]=1;
```

3) 如给全部元素赋值，则在数组说明中， 可以不给出数组元素的个数。

例如：

```
int a[5]={1,2,3,4,5};
```

可写为：

```
int a[]={1,2,3,4,5};
```

7.1.2 一维数组程序举例

可以在程序执行过程中，对数组作动态赋值。这时可用循环语句配合 scanf 函数逐个对数组元素赋值。

【例 7.4】

```
main()
{
  int i,max,a[10];
  printf("input 10 numbers:\n");
  for(i=0;i<10;i++)
      scanf("%d",&a[i]);
```

```
  max=a[0];
  for(i=1;i<10;i++)
      if(a[i]>max) max=a[i];
  printf("maxmum=%d\n",max);
}
```



本例程序中第一个 for 语句逐个输入 10 个数到数组 a 中。 然后把 a[0]送入 max 中。在第二个 for 语句中，从 a[1]到 a[9]逐个与 max 中的内容比较，若比 max 的值大，则把该下标变量送入 max 中，因此 max 总是在已比较过的下标变量中为最大者。比较结束，输出 max 的值。

7.2 二维数组的定义和引用

7.2.1 二维数组的定义

前面介绍的数组只有一个下标，称为一维数组，其数组元素也称为单下标变量。

在实际问题中有很多量是二维的或多维的，因此 C 语言允许构造多维数组。多维数组元素有多个下标，以标识它在数组中的位置，所以也称为多下标变量。本小节只介绍二维数组，多维数组可由二维数组类推而得到。

二维数组定义的一般形式是：

类型说明符 数组名[常量表达式 1][常量表达式 2]

其中常量表达式 1 表示第一维下标的长度，常量表达式 2 表示第二维下标的长度。

例如：

```
int a[3][4];
```

说明了一个三行四列的数组，数组名为 a，其下标变量的类型为整型。该数组的下标变量共有 3×4 个，即：

a[0][0],a[0][1],a[0][2],a[0][3]
a[1][0],a[1][1],a[1][2],a[1][3]
a[2][0],a[2][1],a[2][2],a[2][3]

7.2.2 二维数组元素的引用

二维数组的元素也称为双下标变量，其表示的形式为：

数组名[下标][下标]

其中下标应为整型常量或整型表达式。

例如：

```
a[3][4]
```

表示 a 数组三行四列的元素。

下标变量和数组说明在形式中有些相似，但这两者具有完全不同的含义。数组说明的方括号中给出的是某一维的长度，即可取下标的最大值；而数组元素中的下标是该元素在数组

中的位置标识。前者只能是常量，后者可以是常量，变量或表达式。

【例 7.6】一个学习小组有 5 个人，每个人有三门课的考试成绩。求全组分科的平均成绩和各科总平均成绩。

	张	王	李	赵	周
Math	80	61	59	85	76
C	75	65	63	87	77
Foxpro	92	71	70	90	85

可设一个二维数组 a[5][3]存放五个人三门课的成绩。再设一个一维数组 v[3]存放所求得各分科平均成绩，设变量 average 为全组各科总平均成绩。编程如下：

```
main()
{
  int i,j,s=0,average,v[3],a[5][3];
  printf("input score\n");
  for(i=0;i<3;i++)
  {
      for(j=0;j<5;j++)
      { scanf("%d",&a[j][i]);
        s=s+a[j][i];
      }

      v[i]=s/5;
      s=0;
   }
  average =(v[0]+v[1]+v[2])/3;
  printf("math:%d\nc languag:%d\ndbase:%d\n",v[0],v[1],v[2]);
  printf("total:%d\n", average );
}
```



程序中首先用了一个双重循环。在内循环中依次读入某一门课程的各个学生的成绩，并把这些成绩累加起来，退出内循环后再把该累加成绩除以 5 送入 v[i]之中，这就是该门课程的平均成绩。外循环共循环三次，分别求出三门课各自的平均成绩并存放在 v 数组之中。退出外循环之后，把 v[0],v[1],v[2]相加除以 3 即得到各科总平均成绩。最后按题意输出各个成绩。

7.2.3 二维数组的初始化

二维数组初始化也是在类型说明时给各下标变量赋以初值。二维数组可按行分段赋值，也可按行连续赋值。

例如对数组 a[5][3]：

1) 按行分段赋值可写为:

```
int a[5][3]={ {80,75,92},{61,65,71},{59,63,70},{85,87,90},{76,77,85} };
```

2) 按行连续赋值可写为:

```
        int a[5][3]={ 80,75,92,61,65,71,59,63,70,85,87,90,76,77,85};
```

这两种赋初值的结果是完全相同的。

【例 7.7】

```
main()
{
  int i,j,s=0, average,v[3];
  int a[5][3]={{80,75,92},{61,65,71},{59,63,70},{85,87,90},{76,77,85}};
  for(i=0;i<3;i++)
      { for(j=0;j<5;j++)
            {
            s=s+a[j][i];
            }
        v[i]=s/5;
        s=0;
      }
  average=(v[0]+v[1]+v[2])/3;
  printf("math:%d\nc languag:%d\ndFoxpro:%d\n",v[0],v[1],v[2]);
  printf("total:%d\n", average);
 }
```



对于二维数组初始化赋值还有以下说明：

1) 可以只对部分元素赋初值，未赋初值的元素自动取 0 值。

例如：

```
int a[3][3]={{1},{2},{3}};
```

是对每一行的第一列元素赋值，未赋值的元素取 0 值。 赋值后各元素的值为：

```
1 0 0
2 0 0
3 0 0
```

```
int a [3][3]={{1,1},{1,1,2},{3}};
```

赋值后的元素值为:

```
1 1 0
1 1 2
3 0 0
```

2) 如对全部元素赋初值，则第一维的长度可以不给出。

例如：

```
int a[3][3]={1,2,3,4,5,6,7,8,9};
```

可以写为：

```
int a[][3]={1,2,3,4,5,6,7,8,9};
```

7.3 字符数组

用来存放字符量的数组称为字符数组。

7.3.1 字符数组的定义

形式与前面介绍的数值数组相同。

例如：

```
char c[10];
```

由于字符型和整型通用，也可以定义为 int c[10]但这时每个数组元素占 2 个字节的内存单元。

字符数组也可以是二维或多维数组。

例如：

```
char c[5][10];
```

即为二维字符数组。

7.3.2 字符数组的初始化

字符数组也允许在定义时作初始化赋值。

例如：

```
char c[10]={'c', ' ', 'p', 'r', 'o', 'g', 'r', 'a','m'};
```

赋值后各元素的值为：

```
数组 C    c[0]的值为'c'
          c[1]的值为' '
          c[2]的值为'p'
          c[3]的值为'r'
          c[4]的值为'0'
          c[5]的值为'g'
          c[6]的值为'r'
          c[7]的值为'a'
          c[8]的值为'm'
```

其中 c[9]未赋值，系统自动赋予 空 值。

当对全体元素赋初值时也可以省去长度说明。

例如：

```
char c[]={`c`,` `,`p`,`r`,`o`,`g`,`r`,`a`,`m`};
```

这时 C 数组的长度自动定为 9。

7.3.3 字符数组的引用

【例 7.8】

```
main()
{
  int i,j;
  char a[][5]={{'B','A','S','I','C',},{'d','B','A','S','E'}};
  for(i=0;i<=1;i++)
    {
      for(j=0;j<=4;j++)
          printf("%c",a[i][j]);
      printf("\n");
    }
}
```



本例的二维字符数组由于在初始化时全部元素都赋以初值，因此一维下标的长度可以不加以说明。

7.3.4 字符串和字符串结束标志

在 C 语言中没有专门的字符串变量，通常用一个字符数组来存放一个字符串。前面介绍字符串常量时，已说明字符串总是以'\0'作为串的结束符。因此当把一个字符串存入一个数组时，也把结束符'\0'存入数组，并以此作为该字符串是否结束的标志。有了'\0'标志后，就不必再用字符数组的长度来判断字符串的长度了。

C 语言允许用字符串的方式对数组作初始化赋值。

例如：

```
char c[]={'c', ' ','p','r','o','g','r','a','m'};
```

可写为：

```
char c[]={"C program"};
```

或去掉{}写为：

```
char c[]="C program";
```

用字符串方式赋值比用字符逐个赋值要多占一个字节，用于存放字符串结束标志'\0'。上面的数组 c 在内存中的实际存放情况为：

C		p	r	o	g	r	a	m	\0

'\0'是由 C 编译系统自动加上的。由于采用了'\0'标志，所以在用字符串赋初值时一般无须指定数组的长度， 而由系统自行处理。

7.3.5 字符数组的输入输出

在采用字符串方式后，字符数组的输入输出将变得简单方便。

除了上述用字符串赋初值的办法外，还可用 printf 函数和 scanf 函数一次性输出输入

一个字符数组中的字符串，而不必使用循环语句逐个地输入输出每个字符。

【例 7.9】

```
main()
{
  char c[]="BASIC\ndBASE";
  printf("%s\n",c);
}
```



注意在本例的 printf 函数中，使用的格式字符串为"%s"，表示输出的是一个字符串。而在输出表列中给出数组名则可。不能写为：

```
printf("%s",c[]);
```

【例 7.10】

```
main()
{
  char st[15];
  printf("input string:\n");
  scanf("%s",st);
  printf("%s\n",st);
}
```



本例中由于定义数组长度为 15，因此输入的字符串长度必须小于 15，以留出一个字节用于存放字符串结束标志`\0`。应该说明的是，对一个字符数组，如果不作初始化赋值，则必须说明数组长度。还应该特别注意的是，当用 scanf 函数输入字符串时，字符串中不能含有空格，否则将以空格作为串的结束符。

例如当输入的字符串中含有空格时，运行情况为：

```
input string:
this is a book
```

输出为：

```
this
```

从输出结果可以看出空格以后的字符都未能输出。为了避免这种情况，可多设几个字符数组分段存放含空格的串。

程序可改写如下：

【例 7.11】

```
main()
{
  char st1[6],st2[6],st3[6],st4[6];
  printf("input string:\n");
  scanf("%s%s%s%s",st1,st2,st3,st4);
  printf("%s %s %s %s\n",st1,st2,st3,st4);
}
```



7.3.6 字符串处理函数

C 语言提供了丰富的字符串处理函数， 大致可分为字符串的输入、输出、合并、修改、比较、转换、复制、搜索几类。 使用这些函数可大大减轻编程的负担。用于输入输出的字符串函数，在使用前应包含头文件"stdio.h"，使用其它字符串函数则应包含头文件"string.h"。

下面介绍几个最常用的字符串函数。

1. 字符串输出函数 puts

格式： puts（字符数组名）

功能：把字符数组中的字符串输出到显示器。 即在屏幕上显示该字符串。

【例 7.12】

```
#include"stdio.h"
main()
{
  char c[]="BASIC\ndBASE";
  puts(c);
}
```



从程序中可以看出 puts 函数中可以使用转义字符，因此输出结果成为两行。puts 函数完全可以由 printf 函数取代。当需要按一定格式输出时，通常使用 printf 函数。

2. 字符串输入函数 gets

格式： gets （字符数组名）

功能：从标准输入设备键盘上输入一个字符串。

本函数得到一个函数值，即为该字符数组的首地址。

【例 7.13】

```
#include"stdio.h"
main()
{
  char st[15];
  printf("input string:\n");
  gets(st);
  puts(st);
}
```



可以看出当输入的字符串中含有空格时，输出仍为全部字符串。说明 gets 函数并不以空格作为字符串输入结束的标志，而只以回车作为输入结束。这是与 scanf 函数不同的。

3. 字符串连接函数 strcat

格式： strcat（字符数组名 1，字符数组名 2）

功能：把字符数组 2 中的字符串连接到字符数组 1 中字符串的后面，并删去字符串 1 后的串标志"\0"。本函数返回值是字符数组 1 的首地址。

【例 7.14】

```
#include"string.h"
main()
{
  static char st1[30]="My name is ";
  int st2[10];
  printf("input your name:\n");
  gets(st2);
  strcat(st1,st2);
  puts(st1);
}
```



本程序把初始化赋值的字符数组与动态赋值的字符串连接起来。要注意的是，字符数组 1 应定义足够的长度，否则不能全部装入被连接的字符串。

4. 字符串拷贝函数 strcpy

格式： strcpy（字符数组名 1，字符数组名 2）

功能：把字符数组 2 中的字符串拷贝到字符数组 1 中。串结束标志"\0"也一同拷贝。字符数名 2，也可以是一个字符串常量。这时相当于把一个字符串赋予一个字符数组。

【例 7.15】

```
#include"string.h"
main()
{
  char st1[15],st2[]="C Language";
  strcpy(st1,st2);
  puts(st1);printf("\n");
}
```



本函数要求字符数组 1 应有足够的长度，否则不能全部装入所拷贝的字符串。

5. 字符串比较函数 strcmp

格式： strcmp(字符数组名 1，字符数组名 2)

功能：按照 ASCII 码顺序比较两个数组中的字符串，并由函数返回值返回比较结果。

字符串 1＝字符串 2，返回值＝0；

字符串 2〉字符串 2，返回值〉0；

字符串 1〈字符串 2，返回值〈0。

本函数也可用于比较两个字符串常量，或比较数组和字符串常量。

【例 7.16】

```
#include"string.h"
main()
{ int k;
  static char st1[15],st2[]="C Language";
  printf("input a string:\n");
  gets(st1);
```

```
  k=strcmp(st1,st2);
  if(k==0) printf("st1=st2\n");
  if(k>0) printf("st1>st2\n");
  if(k<0) printf("st1<st2\n");
}
```



本程序中把输入的字符串和数组 st2 中的串比较，比较结果返回到 k 中，根据 k 值再输出结果提示串。当输入为 dbase 时，由 ASCII 码可知“dBASE”大于“C Language”故 k〉0，输出结果“st1>st2”。

6. 测字符串长度函数 strlen

格式： strlen(字符数组名)

功能：测字符串的实际长度(不含字符串结束标志‘\0’) 并作为函数返回值。

【例 7.17】

```
#include"string.h"
main()
{ int k;
  static char st[]="C language";
  k=strlen(st);
  printf("The lenth of the string is %d\n",k);
}
```



7.4 本章小结

1.数组是程序设计中最常用的数据结构。数组可分为数值数组(整数组，实数组)，字符数组以及后面将要介绍的指针数组，结构数组等。

2.数组可以是一维的，二维的或多维的。

3.数组类型说明由类型说明符、数组名、数组长度(数组元素个数)三部分组成。数组元素又称为下标变量。 数组的类型是指下标变量取值的类型。

4.对数组的赋值可以用数组初始化赋值，输入函数动态赋值和赋值语句赋值三种方法实现。对数值数组不能用赋值语句整体赋值、输入或输出，而必须用循环语句逐个对数组元素进行操作。