第5章　数　　组

大家知道，程序设计中使用到的变量都属于诸如 int、char、float、double 型中的简单数据类型，对于处理简单问题还不觉得有障碍。当遇到同类、大量数据时，就必须引入数组这一新的数据类型，处理相同特征的批量数据比较便利。

所谓数组，就是相同数据类型的元素按一定顺序排列的集合。根据数据的复杂程序，还可以分为一维数组、二维数组、字符数组等。

5.1　一 维 数 组

多个变量以组(Group)为单位被统一处理，相当于一个变量集合，称之为数组。数组名的命名方法如同变量，采用字母、数字、下划线的组合。

5.1.1　一维数组的定义

一维数组的格式：

```
数据类型　数组名[长度]=初始值；　(长度可省略)
int  data[5]={10, 20, 30, 40, 50};
```

说明：

数组名后[ ]中的有一个长度项，说明此数组是一维数组。

data[0]，data[1]，…，data[4]是数组元素。

0，1，…，4 是数组元素的下标。即 data 数组中各元素的值分别为：

```
data[0]=10;
data[1]=20;
data[2]=30;
data[3]=40;
data[4]=50;
```

注：数组元素的下标是从 0 开始的，故不存在 data[5]。

长度值：正整数或常量，以及其表达式，如下列设计也是可以的。

```
#define  SUM  5
int  data[SUM]={10, 20, 30, 40, 50};
```

问题 1　正确定义数组

以下是有关数组的定义，请选出定义正确的标号。

(1) int data={1, 2, 3};

(2) int data=(1, 2, 3);

(3) int data[];

(4) int data[]={1, 2, 3};

(5) int data[3]={1, 2, 3};

(6) int data[3]={1.2, 2.0, 3};

(7) int data[5]={1, 2, 3};

答案

(4) (5) (7)

解说

(1)属于语法错误,缺少[]。长度值可以默认,如果加上[],而不给出长度值,也是可以的。

如果在定义数组的同时,给数组进行了赋值,此时的长度值可省略。其长度默认是所赋值的个数,如(4)。反之,若不赋值,数组长度是不可以省略的,如(3)属非法定义。

所赋的值一定要与数组的数据类型匹配,(6)是不合法的。

数组长度大于赋值个数时,依据数组赋值规律,未赋值元素值为 0(因为数组是 int 型)。

注意

数组的命名、定义同普通变量,且在内存中开辟了连续的空间。

5.1.2 一维数组的应用

问题 2 数组元素值的输出

将 19、8、12、16、11 这 5 个数存放于一数组中,并验证存放正确与否。

答案

```
#include "stdio.h"

main(){
    int i;
    int data[]={19, 8, 12, 16, 11};
    for(i=0; i<5; i++) {
        printf("第%d 个值: %d\n", i+1, data[i]);
    }
    return 0;
}
```

程序运行结果:

```
第1个值: 19
第2个值: 8
第3个值: 12
第4个值: 16
第5个值: 11
```

解说

5 个数据,故数组 data 的默认长度为 5,且从 data[0]开始顺序存储,如图 5-1 所示。

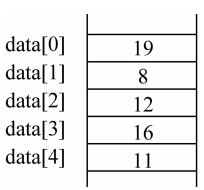


图 5-1 数组元素在内存中的存储示意图

拓展 1:若逆序输出数组元素值,程序将如何修改?

拓展 2:将数组中各元素循环右移三个位置,该如何补充

代码？

分析：首先想到的是数组单元依次赋值，但位置与数值的个数相同，单纯赋值势必导致现有数据被覆盖，故有必要在依次赋值前临时保存首个目标数组单元中的值，待依次赋值完毕后再将临时保留数据存放入指定位置。代码如下。

```
int temp,t;                          //temp：临时变量，t：计数器
for(t=1;t<=3;t++){                   //按照题设要求，进行三轮移动，每轮动作相同
    temp=data[4];                    //暂时保存首个"移动目标数组单元"中的数据
    for(i=4;i>0;i--){                //下标从大到小，可避免数据覆盖
        data[i]=data[i-1];           //依次赋值，实现右移
    }
    data[0]=temp;                    //最左端空出的数组单元存放曾保留的临时数据
}
```

注意

程序设计 for 循环时，往往将循环变量 i 的初始值设为 0，利于循环体中编程需求。若将 19、8、12、16、11 这 5 个数逆序输出，则只需修改 for 语句便可，如下。

```
for(i=4; i>=0; i--)
```

问题 3　数组应用：求解 Fibonacci 数列

Fibonacci 数列公式：

已知：$a_1=a_2=1$，$a_n=a_{n-1}+a_{n-2}$

即：此数列为 1,1,2,3,5,8,13,…

答案

```
#include<stdio.h>

main(){
    int i;
    int fib[20]={1,1};                  //第 1,2 个数组元素赋值为 1,其余默认为 0

    for(i=2; i<20; i++)                 //从第 3 个数开始，可以采用计算公式
        fib[i]=fib[i-2]+fib[i-1];       //for 循环体中仅有一条语句，故{ }可省略

    for(i=0; i<20; i++){
        if(i%5==0) printf("\n");        //控制换行，确保每行输出 5 个数
        printf("%10d", fib[i]);         //输出数的域宽为 10，区域右侧显示
    }
    return 0;
}
```

程序运行结果：

```
         1         1         2         3         5
         8        13        21        34        55
        89       144       233       377       610
       987      1597      2584      4181      6765请按任意键继续. . .
```

解说

将代数式的下标巧妙地转化为数组下标，在处理数学及工程问题时经常使用，这也是数组的优势。

试问：输出结果中的第一行(空行)是如何产生的？

5.2 字符数组与字符串

在 C 语言的基本类型中，不存在字符串型；字符串是存放在字符型数组中。

5.2.1 char 型数组的定义

char c[]={ 'a', 'b', 'c' };

输出时，语法格式有以下两种。

格式一：采用循环，逐个元素输出。

```
for(i=0; i<3; i++)
    printf("%c\n", c[i]);
```

格式二：以字符串方式一并输出。

```
printf("%s\n", c);              /* 输出格式书写为"%s",而非"%c" */
```

5.2.2 字符串的定义

char str[]="abc";　　//字符串"abc"，存放到了字符数组 str 中

5.2.3 字符数组与字符串的区别

字符串 str 的最后位置(str[3])存放的是 '\0'，故此字符串 str 的数组的大小为 4。(用 strlen()函数计算字符串长度时，不计'\0'，strlen(str)=3。)

char 型的数组 c，其末尾若追回 '\0'，就变成了字符串。即：

```
char c[]={'a', 'b', 'c', '\0'};
```

问题 4　典型例：字符数组与字符串操作

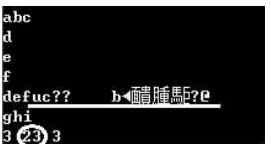
写出以下程序的运行结果，并分析。

```
#include "stdio.h"
#include "string.h"

main(){
    char str[]="abc";
    char c[]={'d', 'e', 'f'};
    char s[]={'g', 'h', 'i', '\0 };
    int i;

    printf("%s\n", str);
```

```
    for(i=0;i<3;i++)
        printf("%c\n", c[i]);

    printf("%s\n", c);          //字符数组,无'\0',连同当前的内存垃圾一并输出了
    printf("%s\n", s);          //字符串,结尾是'\0',输出结束

    printf("%d %d %d\n", strlen(str), strlen(c), strlen(s));

    return 0;
}
```

答案

程序运行结果：



解说

输出结果中第 5 行“def”之后输出的是内存垃圾(俗称乱码)。

strlen()：获取字符串长度的函数。此函数定义在头文件 string.h 中。

c 并非字符串,所以 strlen(c)无任何意义,即 23 没有参考价值;而 str 和 s 是字符串,其值均为 3。此外,全角(包括汉字)的长度为 2,如：

```
strlen("问题 4 解说 A")=9
```

注意

在实际开发中,大量使用的是字符串,特别是在嵌入式产品研发时,大多是通过字符串传递消息。赋值前必定要清除内存,往往使用 memset()函数来初始化。

问题 5　字符串的覆盖赋值

写出以下程序运行结果。

```
#include "stdio.h"

main(){
    char str[10]="abcdef";          //字符串赋值,且 str[6]～str[9] ← '\0'
    int i;

    scanf("%s", str);               //格式为"%s",而非"%c";str[3] ← '\0'
    printf("%s\n", str);            //'\0'前的内容,输出

    for(i=0;i<10;i++)
        printf("str[%d]:%c\n", i, str[i]);
    return 0;
}
```

答案

程序运行结果：

```
opq↙
opq
str[0]:o
str[1]:p
str[2]:q
str[3]:
str[4]:e
str[5]:f
str[6]:
str[7]:
str[8]:
str[9]:
```

解说

标准输入"opq"↙(Enter)后，str[0]～str[2]中存入了'o'，'p'，'q' 三个字符，str[3]中自然会存入表示字符串终止的空控制符'\0'。

str[4]～str[9]中的数组元素值，依旧如初始赋值状态不变。

注意

在标准输入(scanf)赋值前后，用 strlen 函数分别求 str 的长度值，应为多少？

```
printf("%d\n", strlen(str));
```

(答案：6，3)

问题 6　字符串逐字符输出

求下列程序中 len 的值、for 循环的次数、i 的终值。

```
#include "stdio.h"
#include<string.h>

main(){
    int i;
    char str[10]="edu.cn";
    int len=strlen(str);
        for(i=0; i<len; i++){
              printf("%c", str[i]);
        }
    return 0;
}
```

答案

6、6、6

解说

str 数组中后 4 个数组元素中全部存放的是'\0'，故 strlen(str)＝6。

问题 7　字符统计

统计字符串中小写字母出现的次数，并将出现次数最多的小写字母输出。

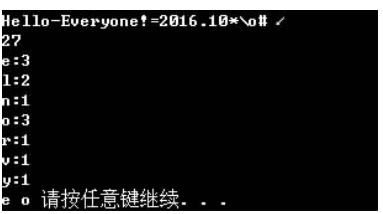
答案

程序清单如下：

```
#include "stdio.h"
```

```
#include "string.h"

main(){
    int i,t[26]={0},k,max=0;                          //t[26]: 存放各字母出现个数
    char str[100];
    scanf("%s",&str);                                 //Debug 时也可写成固定字符串
    printf("%d\n",strlen(str));
    for(i=0;i<strlen(str);i++){
        k=(int)str[i]-97;                             //字母与数组下标对应
        t[k]++;                                       //分别计数
    }
    for(i=0;i<26;i++){
        if(t[i]!=0) printf("%c:%d\n",97+i,t[i]); //仅输出曾出现过的字母
        if(max<t[i]){max=t[i];}                       //最值
    }

    for(i=0;i<26;i++){
        if(max==t[i]){                                //最值可能有多个
            printf("%c ",i+97);
        }
    }
    return 0;
}
```

程序运行结果：



解说

a 的 ASCII 值为 97(十进制数)，需要特殊记忆。其后续字母的 ASCII 值，可以通过依次累加 1 获取，例如，A 的 ASCII 值为 65(97－32＝65)。

利用 ASCII 值，可建立字母和存储其出现次数的数组下标的关系式。

寻找字母出现次数最多者，但输出时要注意，最值可能不止一个。

键盘输入总字符数为 27，其中出现字母次数最多为三次，分别是 e 和 o 两个字母。

注意

scanf()开始读取输入以后，会在遇到空格(Blank)、制表符(Tab)或者换行符(Newline)处停止读取。诸如标准输入中间有空格时，会使程序员产生与代码中的 strlen 计算值不一致的误判。若使用：

```
scanf("%[^\n]s",&str);
```

则可实现只有在读入回车符时才终止读取。

5.3 字符串函数

在使用字符串处理函数时,需在程序的头文件中追加 string.h,如:

```
#include "string.h"
```

string.h 是 C 语言的标准库文件,声明、定义了对字符串操作的函数。无须死记硬背,需要时,查阅库函数手册或在线帮助便可。此外,不同的编译系统所提供的函数名及其功能不尽相同,使用时应适当留意。

常用函数有 strlen()、strcpy()、strncpy()、strcmp()、strcat()、memset()、memcpy()等。

5.3.1 strlen(): 求字符串长度函数

功能: 计算给定字符串的长度。

说明:

计算长度不包括空字符'\0'在内。

5.3.2 strcpy(字符串数组名,代入字符串): 字符串复制函数

功能: 复制字符串函数。

说明:

(1) 数组长度要足够大;

(2) 代入字符串也可以是字符串数组;

(3) 开发工程中几乎不使用,其原因是速度过慢、效率低下。

例如:

```
char str[6]="abcde";
printf("%s\n",str);        //abcde
strcpy(str,"Cha");
printf("%s",str);          //Cha
```

5.3.3 strncpy(str, "Cha", 2): 字符串定长复制函数

功能: 指定长字符串复制。

说明:

用带入的字符串"Cha"前两位,取代原有字符串 str 中最前两个字符。

例如:

```
char str[6]="abcde";
printf("%s\n",str);        //abcde
strncpy(str,"Cha",2);
printf("%s",str);          //Chcde
```

5.3.4 strcmp(Str1,Str2)：字符串比较函数

功能：顺序、逐字符比较，相异或遇到'\0'时，停止比较。

Str1＝Str2，返回值＝0；

Str1＞Str2，返回值＞0；

Str1＜Str2，返回值＜0。

说明：

比较的是 ASCII 值的大小。

例如：

```
strcmp("9", "11");              //1
strcmp("imut", "imut");         //0
strcmp("abc", "bcd");           //-1
```

5.3.5 strcat(str1，str2)：字符串连接函数

功能：连接两个 char 型数组 str1 和 str2。

说明：

(1) str1 足够长；

(2) str2 接到 str1 之后，存入 str1。

例如：

```
char str1[20]="Hello";
char str2[]="C 语言";
printf("%s", strcat(str1,str2));          //HelloC 语言
```

5.3.6 memset()：内存初始化函数

功能：将指定的内存地址进行初始化(俗称清 0)操作。

```
memset(&cnf_msg, 0x00, sizeof(cnf_msg));    //0x00 是十六进制的 0
memset(strbuf, 0, sizeof(strbuf));
```

5.3.7 memcpy()：内存复制函数

功能：memcpy 是内存复制函数。其作用是从源内存地址的起始位置(RoomNo)开始，复制 n 个字节到目标内存地址的起始位置(TermRoom[i])中。

memset 与 memcpy 经常配合使用，效率远高于 strcpy。

例如：

```
memset(TermRoom[i], 0x00, sizeof(RoomNo));        //置空,NULL
memcpy(TermRoom[i], RoomNo, sizeof(RoomNo));      //复制
```

5.4 二维数组

5.4.1 二维数组的定义

顾名思义，二维数组拥有两个长度变量，所谓二维是行(Row)、列(Column)的排列形式，故二维数组也称矩阵(Matrix)。

定义格式：

```
数据类型  数组名[行长][列长]=初始值;
```

格式一：

```
int da[2][3]={ {1,2,3},
               {4,5,6}};
```

格式二：

```
int da[2][3]={1,2,3,4,5,6};
```

各数组元素的值：同一维数组，下标从 0 开始；下标最大的数组元素为 da[1][2]。

```
da[0][0]=1;  da[0][1]=2;  da[0][2]=3;
da[1][0]=4;  da[1][1]=5;  da[1][2]=6;
```

默认赋值：

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

各数组元素的值：

```
da[0][0]=1;    da[0][1]=2;    da[0][2]=0;    da[0][3]=0;
da[1][0]=3;    da[1][1]=0;    da[1][2]=0;    da[1][3]=0;
da[2][0]=0;    da[2][1]=0;    da[2][2]=0;    da[2][3]=0;
```

问题 8　定义考查

请指出下列数组说明中的正误。

(1) float a[0];

(2) int b(2)(3);

(3) int k=3, a[k];

(4) float a[3,4],b[5,10];

(5) int a[3][4]; a[3][4]=3;

答案

全部错误。

解说

(1) 数组大小误设为 0，没有意义。

(2) 不能使用圆括号。