**c语言字符数组与字符串的使用详解**

**1、字符数组的定义与初始化**字符数组的初始化，最容易理解的方式就是逐个字符赋给数组中各元素。  
char str[10]={ 'I',' ','a','m',' ',‘h','a','p','p','y'};  
即把10个字符分别赋给str[0]到str[9]10个元素  
如果花括号中提供的字符个数大于数组长度，则按语法错误处理；若小于数组长度，则只将这些字符数组中前面那些元素，其余的元素自动定为空字符（即 '\0' )。

**2、字符数组与字符串**在c语言中，将字符串作为字符数组来处理。（c++中不是）  
在实际应用中人们关心的是有效字符串的长度而不是字符数组的长度，例如，定义一个字符数组长度为100，而实际有效字符只有40个，为了测定字符串的实际长度，C语言规定了一个“字符串结束标志”，以字符'\0'代表。如果有一个字符串，其中第10个字符为'\0'，则此字符串的有效字符为9个。也就是说，在遇到第一个字符'\0'时，表示字符串结束，由它前面的字符组成字符串。  
系统对字符串常量也自动加一个'\0'作为结束符。例如"C Program”共有9个字符，但在内存中占10个字节，最后一个字节'\0'是系统自动加上的。（通过sizeof()函数可验证）  
有了结束标志'\0'后，字符数组的长度就显得不那么重要了，在程序中往往依靠检测'\0'的位置来判定字符串是否结束，而不是根据数组的长度来决定字符串长度。当然，在定义字符数组时应估计实际字符串长度，保证数组长度始终大于字符串实际长度。（在实际字符串定义中，常常并不指定数组长度，如char str[ ]）  
**说明**：'\n'代表ASCII码为0的字符，从ASCII码表中可以查到ASCII码为0的字符不是一个可以显示的字符，而是一个“空操作符”，即它什么也不干。用它来作为字符串结束标志不会产生附加的操作或增加有效字符，只起一个供辨别的标志。  
对C语言处理字符串的方法由以上的了解后，再对字符数组初始化的方法补充一种方法——即可以用字符串常量来初始化字符数组：  
char str[ ]={"I am happy"};       可以省略花括号，如下所示  
char str[ ]="I am happy";  
**注意**：上述这种字符数组的整体赋值只能在字符数组初始化时使用，不能用于字符数组的赋值，字符数组的赋值只能对其元素一一赋值，下面的赋值方法是错误的  
            **char str[ ];  
            str="I am happy";**不是用单个字符作为初值，而是用一个字符串（注意：字符串的两端是用双引号“”而不是单引号‘'括起来的）作为初值。显然，这种方法更直观方便。（**注意**：数组str的长度不是10，而是11，这点请务必记住，因为字符串常量"I am happy"的最后由系统自动加上一个'\0'）   
因此，上面的初始化与下面的初始化等价  
char str[ ]={'I',' ','a','m',' ','h','a','p','p','y','\0'};  
而不与下面的等价  
char str[ ]={'I',' ','a','m',' ','h','a','p','p','y'};  
前者的长度是11，后者的长度是10.  
**说明：**字符数组并不要求它的最后一个字符为'\0'，甚至可以不包含'\0'，向下面这样写是完全合法的。  
char str[5]={'C','h','i','n','a'};  
++++++++  
可见，用两种不同方法初始化字符数组后得到的数组长度是不同的。

#include <stdio.h>  
void main(void)  
{  
char c1[]={'I',' ','a','m',' ','h','a','p','p','y'};  
char c2[]="I am happy";  
int i1=sizeof(c1);  
int i2=sizeof(c2);  
printf("%d\n",i1);  
printf("%d\n",i2);  
}  
结果：10   11

**3、字符串的表示形式**在C语言中，可以用两种方法表示和存放字符串：  
（1）用字符数组存放一个字符串  
          char str[ ]="I love China";  
（2）用字符指针指向一个字符串  
          char\* str="I love China";  
对于第二种表示方法，有人认为str是一个字符串变量，以为定义时把字符串常量"I love China"直接赋给该字符串变量，这是不对的。  
C语言对字符串常量是按字符数组处理的，在内存中开辟了一个字符数组用来存放字符串常量，程序在定义字符串指针变量str时只是把字符串首地址（即存放字符串的字符数组的首地址）赋给str。  
**两种表示方式的字符串输出都用**printf("%s\n",str);  
%s表示输出一个字符串，给出字符指针变量名str（对于第一种表示方法，字符数组名即是字符数组的首地址，与第二种中的指针意义是一致的），则系统先输出它所指向的一个字符数据，然后自动使str自动加1，使之指向下一个字符...，如此，直到遇到字符串结束标识符 " \0 "。  
  
**4、对使用字符指针变量和字符数组两种方法表示字符串的讨论**虽然用字符数组和字符指针变量都能实现字符串的存储和运算，但它们二者之间是有区别的，不应混为一谈。  
4.1、字符数组由若干个元素组成，每个元素放一个字符；而字符指针变量中存放的是地址（字符串/字符数组的首地址），绝不是将字符串放到字符指针变量中（是字符串首地址）  
4.2、**赋值方式：**     对字符数组只能对各个元素赋值，不能用以下方法对字符数组赋值  
     char str[14];  
     str="I love China";     （但在字符数组**初始化**时可以，即char str[14]="I love China";）  
     而对字符指针变量，采用下面方法赋值：  
     char\* a;  
     a="I love China";  
     或者是 char\* a="I love China";       都可以  
4.3、对字符指针变量赋初值（**初始化**）：  
         char\* a="I love China";        
等价于：  
           char\* a;  
           a="I love China";  
而对于字符数组的初始化  
          char str[14]="I love China";  
不能等价于：  
         char str[14];  
         str="I love China"; （这种不是初始化，而是赋值，而对数组这样赋值是不对的）  
4.4、如果定义了一个字符数组，那么它有确定的内存地址；而定义一个字符指针变量时，它并未指向某个确定的字符数据，并且可以多次赋值。

**5、字符串处理函数**5.1  
char \*strcat(char \*str1,const char \*2 );  
char \*strcat(char \*strDestination,const char \*strSource );  
功能：函数将字符串str2 连接到str1的末端，并返回指针str1  
注：连接前两个字符串的后面都有一个' \0 '，连接时将字符串1后面的 ' \0 ‘去掉，只在新串最后保留一个 ' \0 ‘  
5.2  
char \*strcpy(char \*str1,const char \*2 );  
char \*strcpy(char \*strDestination,const char \*strSource );  
功能：复制字符串strSource中的字符到字符串strDestination，包括空值结束符。返回值为指针strDestination。  
注：1、“字符数组1”必须写成数组名形式，“字符串2"可以是字符数组名，也可以是一个字符串常量  
2、复制时连同字符串后面的 ' \0 ' 一起复制到数组1中  
3、不能用赋值语句直接将一个字符串常量或者字符数组直接赋给一个字符数组（同普通变量数组是一样的），而只能用strcpy函数处理。  
4、可以用strcpy函数将字符串2中的前若干个字符复制到字符数组1中去。