字符数组的声明

字符数组是指元素为字符的数组。

字符数组是用来存放字符序列或字符串的。

字符数组也有一维、二维和三维之分。

字符数组的声明与一般数组相同，只是数组类型是字符型，第一个元素同样是从下标 0 开始，而不是从下标 1 开始。具体格式如下：

```
1. char 数组名[常量表达式 1]…;
```

例如：

```
1. char ch1[5]; //数组 ch1 是一个具有 5 个字符元素的一维字符数组
2. char ch2[3][5]; //数组 ch2 是一个具有 15 个字符元素的二维字符数组
```

字符数组的赋值

字符数组赋值类似于一维数组，赋值分为数组的初始化和数组元素的赋值。初始化的方式有用字符初始化和用字符串初始化两种，也有用初始值表进行初始化的。

【用字符初始化数组】

例如：

```
1. char chr1[5]={'a', 'b', 'c', 'd', 'e'};
```

初始值表中的每个数据项是一个字符，用字符给数组 chr1 的各个元素初始化。

a	b	c	d	e

当初始值个数少于元素个数时，从首元素开始赋值，剩余元素默认为空字符（'\0'）。

```
1. char chr2[5]={'a', 'b', 'c'};
```

a	b	c	\0	\0

字符数组中也可以存放若干个字符，也可以来存放字符串。两者的区别是字符串末尾有结束符（'\0'）。

例如，下面代码在数组 chr3 中存放着一个字符串“abcd”：

```
1. char chr3[5]={'a', 'b', 'c', 'd', '\0'};
```

在一维字符数组中存放着带有结束符的若干个字符称为字符串。因此，字符串是一维数组，但是一维字符数组不等于字符串。

【用字符串初始化数组】

用一个字符串初始化一个一维字符数组，可以写成下列形式：

```
1. char chr4[5]="abcd";
```

使用这种方式初始化字符数组要注意字符串的长度应小于字符数组的大小。

二维字符数组可存放若干个字符串，因此可使用由若干个字符串组成的初始值表给二维字符数组初始化。

例如，下面代码声明的字符数组 chr5 存放 3 个字符串，每个字符串的长度不得大于 3。

```
1. char chr5[3][4]={"abc", "mno", "xyz"};
```

【数组元素赋值】

字符数组的赋值是给该字符数组的各个元素赋一个字符值。

例如：

```
1. char chr[3];
2. chr[0]='a';
3. chr[1]='b';
4. chr[2]='c';
```

对二维、三维字符数组也是如此。

当需要将一个数组的全部元素值赋予另一数组时，不可以用数组名直接赋值的方式，要使用后面介绍的字符串拷贝函数来完成。

字符常量和字符串常量的区别

	字符	字符串
表示方法	'a'、''	"a"、"abc"、""
定界符	单引号	双引号
内容	单个字符	零个、单个或多个字符
存储长度	1 个字节	字符数+1 个字节 增加的一个字节存放字符串结束标志'\0'。

表1. 字符常量和字符串常量的区别

两者的定界符不同，字符常量由单引号括起来，字符串常量由双引号括起来。

字符常量只能是单个字符，字符串常量则可以是零个、单个或多个字符。

可以把一个字符常量赋给一个字符变量，但不能把一个字符串常量赋给一个字符变量。

字符常量占一个字节，而字符串常量占用字节数等于字符串的字节数加 1。增加的一个字节中存放字符串结束标志'\0'。例如：字符常量'a'占一个字节，字符串常量"a"占两个字节。

字符串的输入

可以使用 cin、cin.getline 等多种方式从键盘输入字符串。

【cin 语句：输入一个字符串】

运行下面的程序，输入 I love C++，观察输出结果：

```
#include<iostream>
using namespace std;
int main()
{
    char s[100];
    cin >> s;
    cout << '[' << s << ']' << endl;
    return 0;
}
```

运行结果如下：

```
I love C++
[I]
```

可以发现，如果仅有一个输入字符串，cin 仅仅获取空格前的内容，如果这一行没有空格，才读到行尾。

【cin 语句：输入多个字符串】

运行下面的程序，输入 I love C++，观察输出结果：

```
#include<iostream>
using namespace std;
int main()
{
    char s1[100], s2[100], s3[100];
    cin >> s1 >> s2 >> s3;
    cout << '[' << s1 << ']' << endl;
    cout << '[' << s2 << ']' << endl;
    cout << '[' << s3 << ']' << endl;
    return 0;
}
```

运行结果如下：

```
I love C++
[I]
[love]
[C++]
```

可以发现，三个字符串分别获取了三个单词。

【cin.getline 语句】

使用 cin.getline 只能输入一个字符串，格式如下：

```
cin.getline(字符串名称, 字符串含\0 最大长度);
```

运行下面的程序，输入 I love C++，观察输出结果：

```
#include<iostream>
using namespace std;
int main()
{
    char s[100];
    cin.getline(s, 100);
    cout << '[' << s << ']';
    return 0;
}
```

运行结果如下：

```
I love C++
[I love C++]
```

可以发现，cin.getline 读入的是完整一行，不包括行尾的换行符。

字符串的输出

向屏幕输出字符串可以使用 cout 语句。

【cout 语句】

使用 cout 语句输出一个字符串的格式如下:

```
cout << 字符串名称;
```

观察下面的程序的运行结果:

```
#include<iostream>
using namespace std;
int main()
{
    char s[100] = "I love C++";
    cout << '[' << s << ']';
    return 0;
}
```

运行结果为:

```
[I love C++]
```

常用的字符串处理函数

下面是一些常用的字符串处理函数，使用的时候引入<cstring>，即

```
1. #include<cstring>
```

函数格式	函数功能
char * strcat (char * destination, const char * source);	将 source 连接到 destination 后边，返回 destination
char * strcpy (char * destination, const char * source);	将 source 复制到 destination，返回 destination
int strcmp (const char * str1, const char * str2);	比较 str1 和 str2: 如果 str1>str2，返回一个正整数; 如果 str1=str2，返回 0; 如果 str1<str2，返回一个负整数;
size_t strlen (const char * str);	计算 str 的长度，终止符'\0'不算在长度之内。
char * strstr (char * str1, const char * str2);	判断 str2 是否是 str1 的子串：如果是，返回 str2 在 str1 中首次出现的地址；否则，返回 NULL。

表2. 常用的字符串函数

下面是两个常用的字符大小写转换函数，使用的时候引入<cctype>，这个库是用来检查和测试字符类型的常用库，即

```
1. #include<cctype>
```

int toupper (int c);	将字符 c 转换成大写字母，返回值是大写字母。
int tolower (int c);	将字符 c 转换成小写字母，返回值是小写字母。

表3. 常用的字符串函数