西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

文字信息的表示与处理

1、字符串的表示与存储

2、字符串的输入与输出

6.4 字符串

- 从字符数组衍生出来的“字符串类型”，可以用来表示文本信息的数据需要。

6.4.1 字符数组

- 用来存放字符型数据的数组称为字符数组，其元素是一个字符，定义形式为：

```
char 字符数组名[常量表达式], ......;
```

- 示例

```
char s[20]; //定义字符数组
```

6.4.1 字符数组

- 字符数组就是一个一维数组，其初始化、引用方法与一维数组类似。

- 示例

```
char s[4]={'J','a','v','a'}; //字符数组初始化
```

6.4.1 字符数组

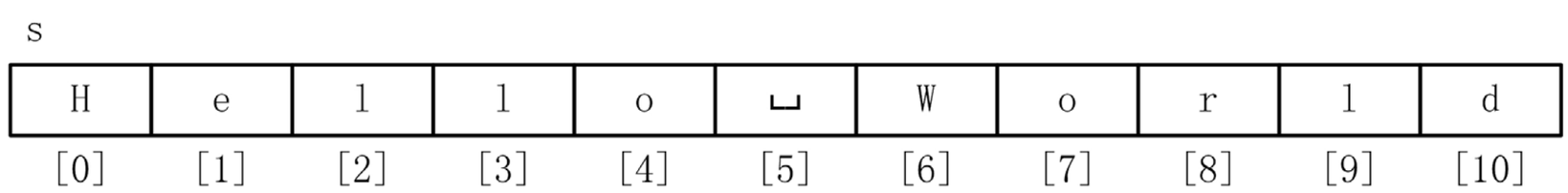
- 由于初值列表的字符通常很多，因此经常不给长度值。
- 例如：

```
char s[]={'H','e','l','l','o',' ␣ ','W','o','r','l','d'};
//字符数组初始化
```

- 这样做的好处是不用人工去数字符的个数，而由编译器自动确定。

6.4.1 字符数组

- 字符数组的内存形式与一维数组类似。例如数组s，初始化后内存形式如下：



- 实线框表示每个字符元素的内存形式，这里用的是字符记号。

6.4.1 字符数组

- 实际上数据应是字符的ASCII值，形式如下：

s

72	101	108	108	111	32	87	111	114	108	100
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]

- 一般在分析字符数组时，习惯采用字符记号。

6.4.1 字符数组

- 字符数组在使用时，同样只能逐个引用字符元素的值而不能一次引用整个字符数组对象，如不能进行赋值、算术运算等。

```
char s1[5]={'B','a','s','i','c'} , s2[5];
s2 = s1; //错误，数组不能赋值
s2[0] = s1[0]; //正确，数组元素赋值
```

6.4.1 字符数组

【例6.8】

连续输入多个字符，直到回车为止；将这一串字符过滤“*”字符后输出，即凡是“*”字符不输出。

6.4.1 字符数组

例6.8

```
1 #include <stdio.h>
2 int main()
3 {
4   char s[100];
5   int i , cnt=0;
6   //连续输入多个字符，直到回车'\n'为止
7   while ( (s[cnt]=getchar()) != '\n') cnt++;
8   for (i=0; i<cnt; i++)
9     if (s[i] != '*') //过滤'*'字符
10      printf("%c",s[i]);
11  return 0;
12 }
```

6.4.1 字符数组

例6.8

```
1 #include <stdio.h>
```

程序运行屏幕

```
ABC123DE456
```

```
ABC*123**DE******456↙
```

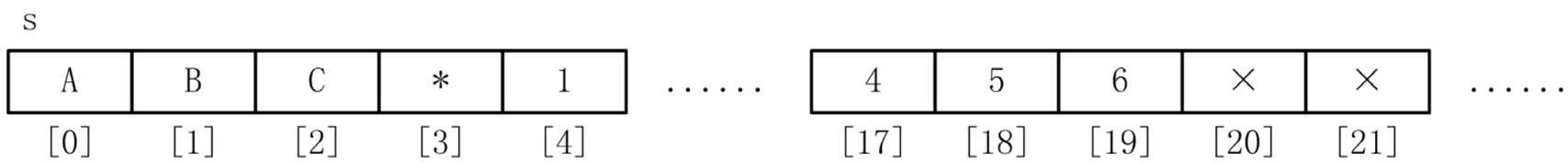
6.4.1 字符数组

例6.8

```
1  #include <stdio.h>
2  int main()
3  {
4    char s[100];
5    int i , cnt=0;
6    //连续输入多个字符，直到回车'\n'为止
7    while ( (s[cnt]=getchar()) != '\n') cnt++;
8    for (i=0; i<cnt; i++)
9      if (s[i] != '*') //过滤'*'字符
10       printf("%c",s[i]);
11   return 0;
12 }
```

6.4.1 字符数组

- 第7行输入字符后数组s的内存形式如下：



- 显然，数组s第19个元素是最后输入的字符'6'，输出打印到这里就要停下来。所以第8行是“i<cnt”而不是“i<100”。数组s自第20个元素后数据是不确定的，用“×”记号。

6.4.2 字符串

- 实际应用中，字符数组存储的实际字符个数未必总是数组长度，因此就要始终记录实际个数，当重新输入一串字符后，这个记录也要随之改变，这样的处理方式在很多情况下是不方便的。

6.4.2 字符串

- 1．字符串的概念
- C语言规定字符串是以'\0'（ASCII值为0）字符作为结束符的字符数组，其中'\0'字符称为空字符（NULL字符）或零字符（Z字符）。

6.4.2 字符串

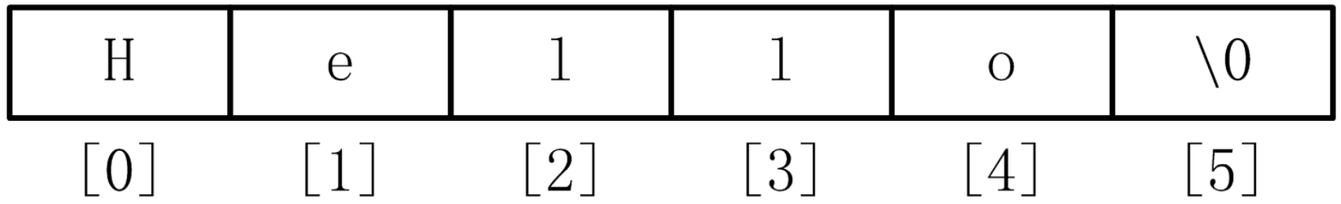
- 字符串概念的引入，解决了字符数组使用上的不方便。它在一串字符后面放上一个空字符，这样就不需要记录字符个数了。
- 因为在程序中可以通过判断数组元素是否为空字符来判断字符串是否结束，换言之，只要遇到数组元素是空字符，就表示字符串在此位置上结束。

6.4.2 字符串

- 字符串长度是指在第1个空字符之前的字符个数（不包括空字符）。特别的，如果第1个字符就是空字符则称该字符串为空字符串，空字符串的字符串长度为0。
- 由于字符串实际存放在字符数组中，所以定义字符数组时数组的长度至少为字符串长度加1（空字符也要占位）。这就要求定义字符数组时充分估计实际字符串的最大长度，保证数组长度始终大于字符串的长度，才不会发生数组越界。

6.4.2 字符串

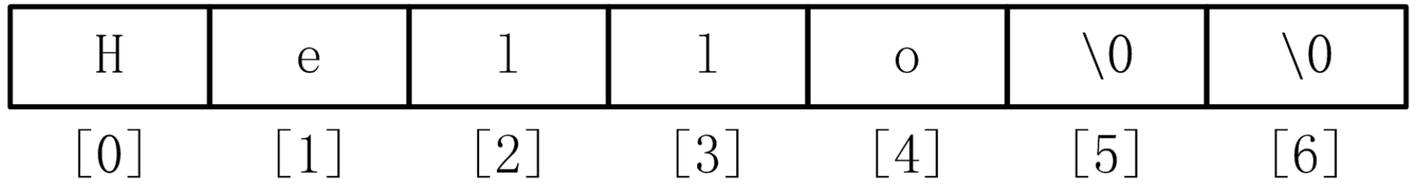
- 字符串常量是字符串的常量形式，它是以一对双引号括起来的字符序列。C语言总是在编译时为字符串常量自动在其后增加一个空字符，例如：
- “Hello”的存储形式为：



- 数组长度是6，字符串长度是5。

6.4.2 字符串

- 即使人为在后面加上空字符也是如此，例如：
- “Hello\0”的存储形式为：



- 数组长度是7，字符串长度也是5。

6.4.2 字符串

- 如果在字符串常量中插入空字符，则字符串常量的长度会比看到的字符数目少，例如：
- “ABC\0DEF”的存储形式为：

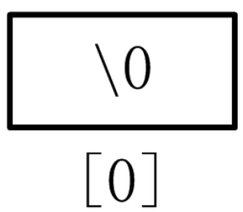
A	B	C	\0	D	E	F	\0
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

- 数组长度是8，字符串长度是3。字符串实际结束在第1个空字符的位置上，这样的现象称为截断字符串。

6.4.2 字符串

- 空字符串尽管字符串长度是0，但它依然要占据字符数组空间，例如：

- 空字符串""的存储形式为：



6.4.2 字符串

- 空字符是字符串处理中最重要的信息。如果一个字符数组中没有空字符而把它当作字符串使用，程序往往因为没有结束条件而数组越界。

6.4.2 字符串

- 空字符（'\0'）容易与字符'0'混淆，其实它们是有区别的。'\0'的ASCII值为0，'0'的ASCII值为48。输出时，'0'会在屏幕上显示0这个符号，而'\0'什么也没有。

6.4.2 字符串

- 文本信息用途非常广，无处不在。如姓名、通信地址、邮箱等，即使像邮政编码这样的数字，也属于文本信息范畴。有了字符串的概念，C程序能方便地表示文本信息。
- 尽管字符串不是C语言的内置数据类型，但应用程序通常都将它当作基本类型来用，称为C风格字符串（C-style string）。

6.4.2 字符串

- 由于数组的整体操作是有限制的，例如不能赋值、运算、输入输出，所以C语言标准库函数中专门针对字符串定义了许多函数，可以方便地处理字符串。

6.4.2 字符串

- 2．字符串的定义和初始化
- 字符串使用字符数组存放，其定义与字符数组完全相同，形式为：

```
char 字符串名[常量表达式], ......;
```

6.4.2 字符串

- C语言允许使用字符串常量初始化字符数组，例如：

```
char s[12]={"Hello World"}; //数组长度为字符串长度加1
```

- 可以不加大括号，直接写成：

```
char s[12]="Hello World"; //字符串初始化
```

6.4.2 字符串

- 由于字符串的字符个数数起来不方便，直接让编译器自动去确定，例如：

```
char s[]="Hello World"; //字符串初始化
```

- 这样为字符串初始化，直观方便。

6.4.2 字符串

- 由于字符串常量结尾是NULL字符，所以上面的初始化与下面等价：

```
char s[]={'H','e','l','l','o',' ␣ ','W','o','r','l','d','\0'};
```

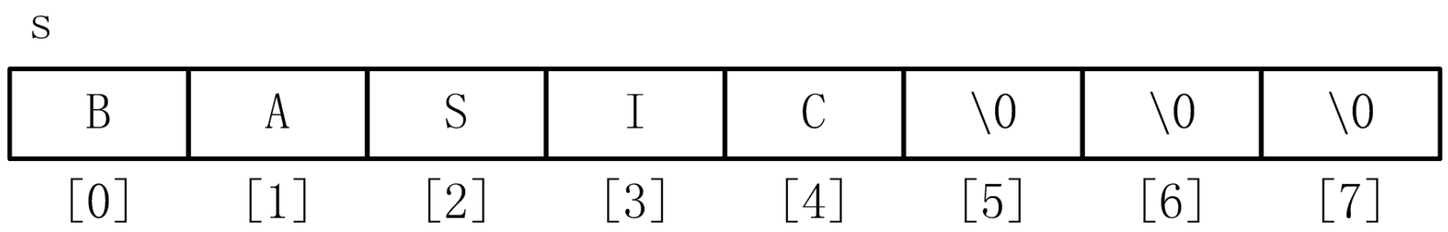
- 而

```
char s[4]={'J','a','v','a'};
```

- 就不能算字符串，因为它没有空字符。

6.4.2 字符串

- 如果字符数组长度值大于初值个数，按照一维数组的规定，只初始化前面的字符元素，剩余元素初始化为0，即空字符。例如：
- char s[8]="BASIC";



6.4.3 字符串的输入和输出

- 字符串的输入和输出有三种方法。
- 1．逐个字符输入输出
- 通过遍历数组元素，采用“%c”格式逐个字符输入输出。

6.4.3 字符串的输入和输出

- 2．使用标准输入输出函数
- 使用格式化输入输出函数，将整个字符串一次输入或输出。例如：

```
char str[80]; //定义字符串也即定义字符数组
scanf("%s",str); //使用%s格式，输入字符串，不需要&
printf("%s",str); //使用%s格式，输出字符串
scanf("%s",str[0]); //错误，%s格式要求字符串而非字符
printf("%s",str[0]); //错误，%s格式要求字符串而非字符
```

- scanf和printf函数允许字符串输入和输出，前提是格式必须用%s，输出项必须是字符数组名，而不能是字符元素。

6.4.3 字符串的输入和输出

- 说明：
- ①使用scanf输入字符串时，从键盘上输入的字符个数应小于字符串定义的数组长度，否则过长的输入导致数组越界；
- ②由于scanf函数将空格、TAB、回车作为输入项的间隔，所以输入字符串时遇到这三个字符就结束；换言之，这种输入方式是不能输入空格、TAB、回车的；

6.4.3 字符串的输入和输出

- ③scanf输入完成后，在字符串末尾添加空字符；

- ④printf函数输出字符串时，只要遇到第1个空字符就结束，而不管是否到了字符数组的末尾。

6.4.3 字符串的输入和输出

- 例如

```
char str[80]="Basic\0Java\0C++";
printf("%s",str); //输出字符串
```

- 程序运行结果如下：

```
Basic
```

- 如果字符串没有空字符，printf就会引起数组越界。printf输出的字符不包含空字符。

6.4.3 字符串的输入和输出

- 3．使用字符串输入输出函数
- （1）gets函数

```
char *gets(char *s);
```

- gets函数输入一个字符串到字符数组s中。s是字符数组或指向字符数组的指针，其长度应该足够大，以便能容纳输入的字符串。

6.4.3 字符串的输入和输出

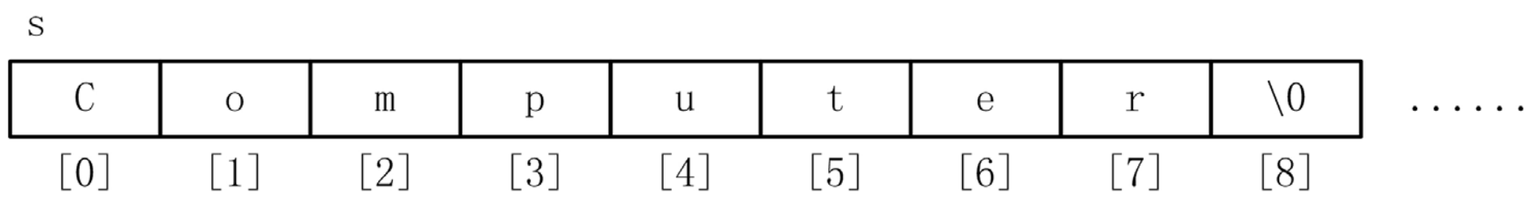
- 例如：

```
char str[80];
gets(str);  //输入字符串
```

- 从键盘上输入：

```
Computer↙
```

- 则字符串str的内存形式为：



6.4.3 字符串的输入和输出

- 说明：
- ①函数调用时用字符数组名，而不是字符元素。例如：

```
gets(str[0]); //错误
```

6.4.3 字符串的输入和输出

- ②使用gets输入字符串时，从键盘上输入的字符个数应小于字符串定义的数组长度，过长的输入将导致数组越界。

- ③gets函数可以输入空格和TAB，但不能输入回车。

- ④gets函数输入完成后，在字符串末尾自动添加空字符。

6.4.3 字符串的输入和输出

- （2）puts函数

```
int puts(char *s);
```

- puts函数输出s字符串，遇到空字符结束，输完后再输出一个换行（'\n'）。s是字符数组或指向字符数组的指针，返回值表示输出字符的个数。

6.4.3 字符串的输入和输出

- 例如：

```
char str[80]="Programming";
puts(str);  //输出字符串
```

- 程序运行结果如下：

```
Programming
```

6.4.3 字符串的输入和输出

- 说明：
- ①函数调用时用字符数组名，而不是字符元素。例如：

```
puts(str[0]); //错误
```

6.4.3 字符串的输入和输出

- ②puts输出字符串时，只要遇到第1个空字符就结束，而不管是否到了数组的末尾。如果字符串没有空字符，puts会引起数组越界。
- ③puts输出的字符不包含空字符。


程序设计