



程序设计与算法基础

程红蓉

信息与软件工程学院

课程主要内容

- 第1章 C语言概述
- 第2章 C语言基本概念
- 第3章 格式化输入/输出
- 第4章 表达式
- 第5章 选择语句
- 第6章 循环
- 第7章 基本类型
- 第8章 数组
- 第9章 函数
- 第10章 程序结构
- 第11章 指针
- 第12章 指针和数组
- 第13章 字符串
- 第14章 预处理
- 第15章 编写大型程序
- 第16章 结构、联合和枚举
- 第22章 输入/输出

- 字符串是以**空字符** `'\0'` 为结尾的字符数组。
- 字符串分为：字符串**字面量**（**常量**）和字符串**变量**。
- 通过**数组或指针**操作字符串
- C提供了操作字符串的一系列库函数。

本章要点

- **字符串字面量**
- **字符串变量**
- **字符串的读和写**
- **访问字符串中的字符**
- **使用C语言的字符串库**
- **字符串惯用法**
- **字符串数组**

13.5 使用C语言的字符串库

```
char str1[10], str2[10];
```

```
str1 = "abc";
```

易错情况

```
str2 = str1;
```

数组名不可赋予新值

```
if (str2 == str1);
```

不同地址比较无意义

13.5 使用C语言的字符串库

- 常用字符串操作可通过**库函数**实现
- 需包含头文件

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

- strcpy, strlen, strcat, strcmp

13.5.1 strcpy字符串复制函数

■ strcpy 字符串复制函数 教材P.207

✓ 函数原型

`char *strcpy(char *dest, const char *src);`

✓ 拷贝**src**指向的字符串到**dest**指向的字符数组；

✓ 返回字符数组的地址，即**dest**的值。

注意：strcpy以空字符作为结束标志，并将空字符一并拷贝到dest指向的字符数组中。

13.5.1 strcpy字符串复制函数

■ 举例1

```
#define N 30  
char dest[N];  
char src[] = "efg" ;
```

```
strcpy(dest, "abcd");
```

dest为abcd

```
strcpy(dest, src);
```

dest为efg

strcpy不检查dest指向的字符数组是否能容纳下src指向的字符串。


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

```
#define N 30
```

```
int main (void){
    char src[] = "abc";
    char dest[N];
    char *p;
```

```
    p = strcpy(dest, src);
    puts(dest);
    puts(p);
    return 0;
```

```
}
```



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

```
#define N 3
```



字符数组容量不够

```
int main (void){
    char src[] = "abc";
    char dest[N];
    char *p;

    p = strcpy(dest, src);
    puts(dest);
    puts(p);
    return 0;
}
```

13.5.1 strcpy字符串复制函数

■ strncpy 字符串复制函数 教材P.207

■ 函数原型

`char *strncpy(char *dest, const char *src, size_t n);`

- ✓ 从src复制n个字符到dest指向的字符数组。
- ✓ 返回字符数组的地址，即dest的值。

在stddef.h中
定义，主要用于
增强移植性

13.5.1 strcpy字符串复制函数

■ 举例2

```
#define N 30  
char dest[N];  
char src[] = "efg" ;
```

```
strncpy(dest, src, sizeof(dest));
```

```
strncpy(dest, src, sizeof(dest) - 1);  
dest[sizeof(dest)-1] = '\0';
```

更安全

13.5.2 strlen字符串长度函数

■ strlen求字符串长度的函数 教材P.208

■ 函数原型

`size_t strlen(const char *s);`

✓ 返回字符串s的长度

特别注意：

1. 返回第一个空字符之前的字符个数，不包括空字符。
2. 不是整个字符数组的长度。

```
#include <stdio.h>
#include <string.h>
#define N 30
int main (void){
    char str1[] = "abc";
    char str2[N];
    int n, len1, len2, len3;
```

```
    n = sizeof(str2);
    len1 = strlen("abc");
    len2 = strlen("");
    strcpy(str2, str1);
    len3 = strlen(str2);
```

```
    printf("n=%d\n", n);
    printf("len1=%d\n", len1);
    printf("len2=%d\n", len2);
    printf("len3=%d\n", len3);
```

```
    return 0;
```

```
}
```

```
n=30
len1=3
len2=0
len3=3
```

对比两者

13.5.3 strcat字符串拼接函数

- **strcat 字符串拼接函数** 教材P.208

- **函数原型**

char *strcat(char *dest, **const char *src);**

✓ 追加字符串src (**包括空字符**) 的内容到字符串dest的末尾 (**删除原有的空字符**) , 返回dest的值。

dest字符串 src字符串

13.5.3 strcat字符串拼接函数

■ 举例3

```
strcpy(str1, "abc");  
strcat(str1, "def");
```

/* str1 now contains "abcdef" */

```
strcpy(str1, "abc");  
strcpy(str2, "def");  
strcat(str1, str2);
```

/* str1 now contains "abcdef" */


```
#include <stdio.h>
#include <string.h>
#define N 30
int main (void){
    char str1[N];
    char str2[N];
    char *p1, *p2;
    int n, len1, len2, len3;

    strcpy(str1, "abc");
    strcpy(str2, "def");
    p1 = strcat(str2, "ghi");
    printf("%s\n",str2);
    printf("%s\n",p1);
    p2 = strcat(str1, p1);
    printf("%s\n",p2);
    return 0;
}
```

"D:\C programs\string-10.exe"

```
defghi
defghi
abcdefghi
```

13.5.3 strcat字符串拼接函数

```
char str1[6] = "abc";
```

```
strcat(str1, "def");
```

易错情况

空字符的存放将越过数组空间

13.5.3 strcat字符串拼接函数

■ strncat字符串拼接函数 教材P.208-209

✓ 函数原型

`char *strncat(char *dest, char *src, size_t n);`

✓ 把src所指字符串的**前n个字符**添加到dest所指字符串的**结尾处**。

✓ **删除dest原有的空字符，在拼接后添加一个空字符。**

13.5.3 strcat字符串拼接函数

■ 举例4

```
#define N 30  
char str1[N] = "abc";  
char str2[6] = "123";
```

```
strncat(str1, str2, sizeof(str1) - strlen(str1) - 1);
```

abc123

13.5.4 strcmp字符串比较函数

■ strcmp 字符串比较函数 教材P.209

✓ 函数原型

`int strcmp(const char *s1, const char *s2);`

✓ s1小于s2返回**负数** ;

✓ s1等于s2返回**0** ;

✓ s1大于s2返回**正数**。

注意：s1和s2的比较基于其ASCII码序列比较。

13.5.4 strcmp字符串比较函数

■ 举例5

- ✓ “27” 与 “28” <
- ✓ “abc” 与 “abe” <
- ✓ “abc” 与 “abcde” <
- ✓ “27” 与 “27” ==
- ✓ “abc” 与 “Abc” >
- ✓ “abc” 与 “0bc” >

```
#include <stdio.h>
#include <string.h>
#define N 30
```

```
#define bool int
```

```
int main (void){
```

```
    char str1[N];
```

```
    char str2[N];
```

```
    bool flag;
```

```
    strcpy(str1, "27");
```

```
    strcpy(str2, "27");
```

```
    flag = strcmp(str1,str2);
```

```
    if (flag == 0)
```

```
        printf("two strings are same.\n");
```

```
    else if (flag < 0)
```

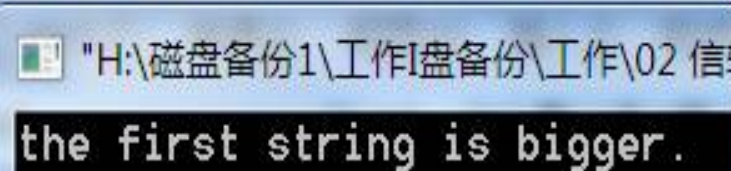
```
        printf("the second string is bigger.\n");
```

```
    else
```

```
        printf("the first string is bigger.\n");
```

```
    return 0;
```

```
}
```



"H:\磁盘备份1\工作I盘备份\工作\02 信

the first string is bigger.

← 字符串 "27" 改为 "5"

13.5.4 strcmp字符串比较函数

■ 还可以这样写

```
if (strcmp(str1, str2) < 0)
```

```
.....
```

```
if (strcmp(str1, str2) == 0)
```

```
.....
```

选择合适的运算符 (<, <=, >, >=, ==, !=)

显示一个月的提醒列表

- 用户输入需要提醒的信息，每条提醒先指明是哪一天（1-31）。
- 输入 0，显示已录入的提醒列表（按日期排序）

输入提醒信息

Enter day and reminder : 24 Susan's birthday
Enter day and reminder : 5 6:00 - Dinner with Marge and Russ
Enter day and reminder : 26 Movie - "Chinatown "
Enter day and reminder : 7 10:30 - Dental appointment
Enter day and reminder : 12 Movie - "Dazed and Confused "
Enter day and reminder : 5 Saturday class
Enter day and reminder : 12 Saturday class
Enter day and reminder : 0

排序后的结果

Day Reminder
5 Saturday class
5 6:00 - Dinner with Marge and Russ
7 10:30 - Dental appointment
12 Saturday class
12 Movie - "Dazed and Confused "
24 Susan's birthday
26 Movie - "Chinatown"

```
int read_line(char str[], int n) {  
    int ch, i = 0;  
    while ((ch = getchar()) != '\n')  
        if (i < n)  
            str[i++] = ch;  
    str[i] = '\0';  
    return i;  
}
```

```
#define MAX_REMIND 50 //最多提醒数量  
#define MSG_LEN 60 //最长提醒字符数  
int read_line(char str[], int n);
```

- 读取一系列**日期和提醒**的组合
- 按日期排序
- 显示结果

输入

```
for (;;) {  
    printf("Enter day and reminder : ");  
    scanf("%2d", &day);  
    .....  
    read_line(...,MSG_LEN);  
    .....  
}
```

排序
跳出循环



```
char reminders[MAX_REMIND][MSG_LEN];
```

输出

```
printf( "\nDay reminder\n");  
for (i=0; i<...; i++)  
    printf( "%s\n" , reminders[i]);
```

```

for (;;) {
    printf("Enter day and reminder : ");
    scanf("%2d", &day);
    if (day == 0)
        break;
    sprintf(day_str, "%2d", day);
    read_line(msg_str, MSG_LEN);
    ..... ← 排序
}

```

```
char msg_str[MSG_LEN+1];
```

```
sprintf(char day_str[2+1];
```

出到第一个参数

```
int num_remind=0;
```

```
for (::) {  
    .....  
    for (i=0;i<num_remind;i++)  
        if (strcmp(day_str,reminders[i])<0)  
            break  
    for (j=num_remind; j>i; j--)  
        strcpy(reminders[j],reminders[j-1]);  
    .....  
}
```

```
strcpy(reminders[...],day_str);  
strcat(reminders[...],msg_str);
```



```
for (;;) {  
    .....  
    for (i=0;i<num_remind;i++)  
        if (strcmp(day_str,reminders[i])<0)  
            break  
    for (j=num_remind; j>i;j--)  
        strcpy(reminders[j],reminders[j-1]);  
    strcpy(reminders[i],day_str);  
    strcat(reminders[i],msg_str);  
    num_remind++;  
}
```

显示一个月的提醒列表（小结）

- **字符串排序两种类型：**
 - ✓ **将字符串全部存入二维数组后再排序；**
 - ✓ **每读入一个字符串就立即为其寻找合适位置。**

显示一个月的提醒列表（小结）

- 从本例子学习到的重要编程思想：
 - ✓ 寻找放置字符串的位置(**for**语句)、然后将该位置及其后的字符串后移，腾出空间(**for**语句)。
 - ✓ **sprintf**函数及格式串“**%2d**”，将**int**型转化为字符串型，并能体现与数值对应的大小关系。
 - ✓ 二维数组**逐行**输出字符串，用的**reminders[i]**作为**printf**的参数。

本章要点

- **字符串字面量**
- **字符串变量**
- **字符串的读和写**
- **访问字符串中的字符**
- **使用C语言的字符串库**
- **字符串惯用法**
- **字符串数组**

13.6.1 搜索字符串的结尾

■ 搜索字符串的结尾 参考教材P.211-212

```
size_t strlen(const char *s) {  
    size_t n;  
  
    for (n = 0; *s != '\0'; s++)  
        n++;  
    return n;  
}
```

两者冲突吗？

13.6.1 搜索字符串的结尾

- 用while语句实现 参考教材P. 212

```
size_t strlen(const char *s) {  
    size_t n = 0;  
  
    while (*s++) 利用空字符ASCII码为0  
        n++;  
    return n;  
}
```

13.6.2 复制字符串

■ 实现字符串拼接

```
char *strcat(char *s1, const char *s2) {  
    char *p = s1;  
    while (*p != '\0')  
        p++;  
  
    while (*s2 != '\0') {  
        *p = *s2;  
        p++;  
        s2++;  
    }  
    *p = '\0';  
    return s1;  
}
```

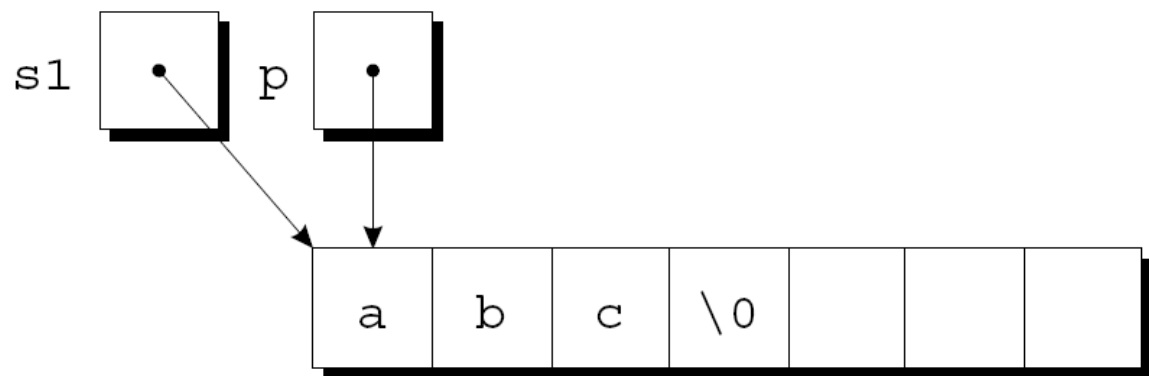


图1

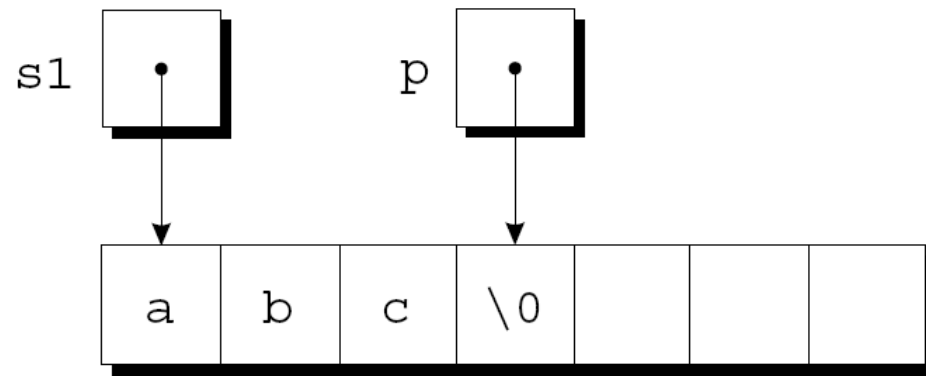


图2

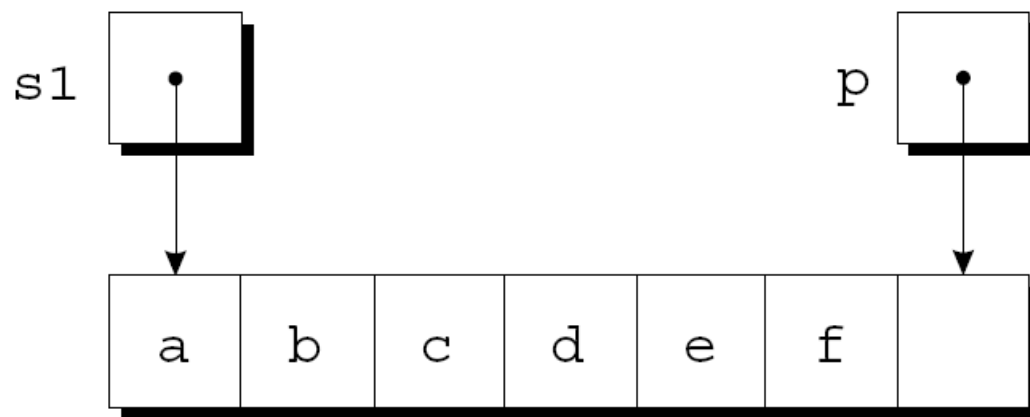
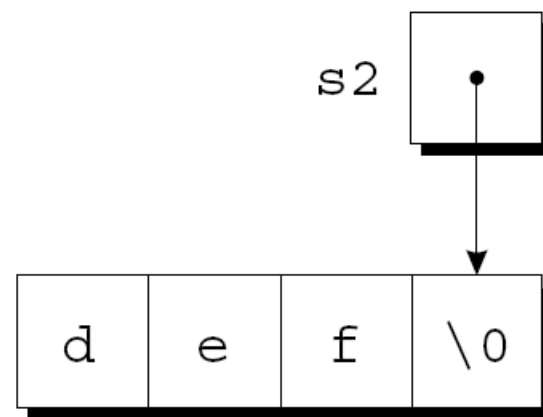


图3



13.6.2 复制字符串

■ 更精简的实现

```
char *strcat(char *s1, const char *s2) {  
    char *p = s1;  
    while (*p)  
        p++;  
    while (*p++ = *s2++);  
    return s1;  
}
```

本章要点

- **字符串字面量**
- **字符串变量**
- **字符串的读和写**
- **访问字符串中的字符**
- **使用C语言的字符串库**
- **字符串惯用法**
- **字符串数组**

13.7 字符串数组

将多个字符串用数组存储管理。

13.7 字符串数组

- 方法一：采用二维字符数组，每行一个字符串
- 举例1 参考教材P. 214

```
char planets[][8] = {"Mercury", "Venus", "Earth",  
                    "Mars", "Jupiter", "Saturn",  
                    "Uranus", "Neptune", "Pluto"};
```

注意：可以省略二维数组的行数，但必须指明列数。

13.7 字符串数组

	0	1	2	3	4	5	6	7
0	M	e	r	c	u	r	y	\0
1	V	e	n	u	s	\0	\0	\0
2	E	a	r	t	h	\0	\0	\0
3	M	a	r	s	\0	\0	\0	\0
4	J	u	p	i	t	e	r	\0
5	S	a	t	u	r	n	\0	\0
6	U	r	a	n	u	s	\0	\0
7	N	e	p	t	u	n	e	\0
8	P	l	u	t	o	\0	\0	\0

参考教材P. 215

如何实现不同
长度的行？

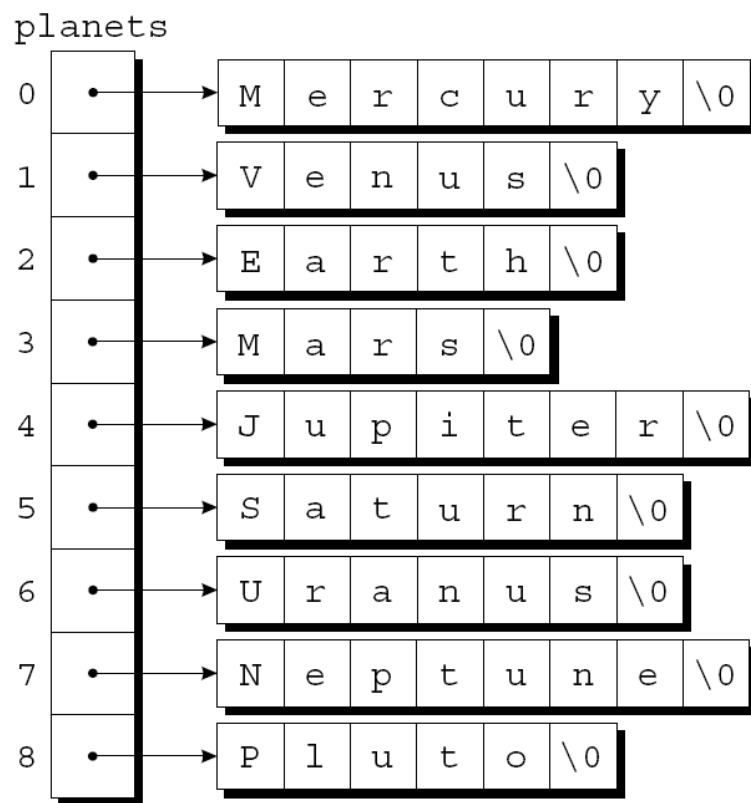
13.7 字符串数组

■ 方法二：采用指针数组的方式

```
char *planets[] = {"Mercury", "Venus", "Earth",  
                  "Mars", "Jupiter", "Saturn",  
                  "Uranus", "Neptune", "Pluto"};
```

13.7 字符串数组

■ 这时planets的存储



一系列指针变量，依次指向各个字符串常量。

13.7 字符串数组

■ 举例2：搜索planets数组中以M开头的字符串。

planets

0	•	M	e	r	c	u	r	y	\0
1	•	V	e	n	u	s	\0		
2	•	E	a	r	t	h	\0		
3	•	M	a	r	s	\0			
4	•	J	u	p	i	t	e	r	\0
5	•	S	a	t	u	r	n	\0	
6	•	U	r	a	n	u	s	\0	
7	•	N	e	p	t	u	n	e	\0
8	•	P	l	u	t	o	\0		

```
for (i = 0; i < 9; i++)
```

```
    if (planets[i][0] == 'M')
```

```
        printf("%s begins with M\n", planets[i]);
```


13.7 字符串数组

- 命令行参数 参考教材P. 216
- 举例3 : main函数可以有形式参数

```
int main(int argc, char *argv[]) {  
    .....  
}
```

13.7 字符串数组

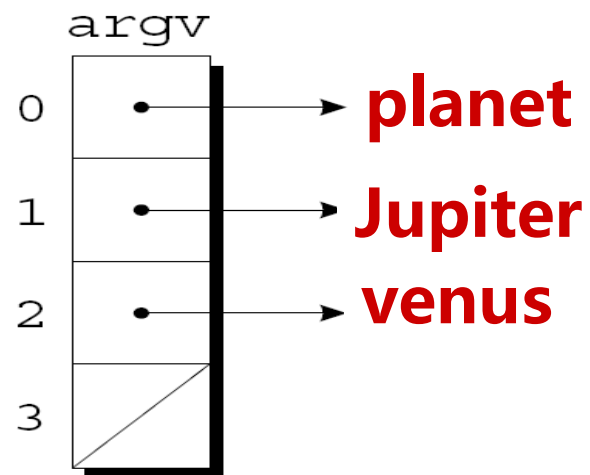
- argc : 命令行参数的**个数**。
- argv : **指针数组** , 指向各个命令行参数 , 命令行参数以字符串方式存储。
 - ✓ argv[0] 指向程序名
 - ✓ argv[1] 至 argv[argc-1] 指向余下的命令行参数。
 - ✓ argv[argc]始终为NULL。

13.7 字符串数组

■ 举例4

planet Jupiter venus

argc 为3, argv 为如下表示：



13.7 字符串数组

- 举例5：编程实现planet.c，测试一系列字符串，找出哪些字符串是行星的名字，并将其编号显示出来（最靠近太阳的行星编号为1）。

```
D:\C programs\第13章 字符串>planet Jupiter venus Earth fred
Jupiter is planet 5
venus is not a planet
Earth is planet 3
fred is not a planet
```

本章作业

第十三章编程题1

Experiment 4

Q6 Find biggest and smallest word

第十三章编程题1

课后练习

第十三章练习题1-18 教材P.220-222

第十三章编程题3, 5, 6



Thank You!