

C 语言

第四讲、字符串与格式化输入输出

张晓平

武汉大学数学与统计学院

2017 年 3 月 14 日

1. 字符串简介

2. 常量与预处理器

3. 格式化输出

4. 格式化输入

1. 字符串简介

字符串简介 I

```
1 // talkback.c:
2 #include <stdio.h>
3 #include <string.h>
4 #define DENSITY 62.4
5 int main(void)
6 {
7     float weight, volume;
8     int size;
9     unsigned long letters;
10    char name[40];
11    printf("Hi! What's your name?\n");
```

字符串简介 II

```
12 scanf("%s", name);
13 printf("%s, what's your weight in pounds?\n",
    name);
14 scanf("%f", &weight);
15 size = sizeof name;
16 letters = strlen(name);
17 volume = weight/DENSITY;
18 printf("Well, %s, your volume is %2.2f cubic
    feet.\n", name, volume);
19 printf("Also, your first name has %lu letters,
    \n", letters);
```

字符串简介 III

```
20     printf("and we have %d bytes to store it in.\n", size);  
21     return 0;  
22 }
```

字符串简介

Hi! What's your name?

Xiaoping

Xiaoping, what's your weight in pounds?

139

Well, Xiaoping, your volume is 2.23 cubic feet.

Also, your first name has 8 letters,

and we have 40 bytes to store it in.

字符串简介

定义 字符串 (character string) 就是一个或多个字符的序列。例如：

```
"Once more you open the door!"
```

注 字符串用双引号括起来，但双引号不是字符串的一部分。

字符串简介

- ▶ C 没有为字符串定义专门的数据类型，而是把它存储在 `char` 数组中。
- ▶ 字符串的字符存放在相邻的存储单元中，每个字符占用一个单元。
- ▶ 而数组由相邻存储单元组成，故把字符串存储在数组中是自然的。

字符串简介

Once more you open the door!\0

↑
每个单元占 1 个字节

↑
空字符

字符串简介

- ▶ 数组中的最后一个位置显示字符 `\0`，该字符是空字符 (null character)，C 用它来标记字符串的结束。
- ▶ 空字符不是数字 0，它是非打印字符，其 ASCII 码的值为 0。
- ▶ 空字符的存在意味着数组的单元数至少比要存储的字符数多 1。

字符串简介

定义 数组 (array) 是同一类型的数据元素的有序序列。

字符串简介：如何创建数组？

```
char name[40];
```

该声明语句创建一个有 40 个存储单元的数组，其中每个单元可存储一个 char 型值。

字符串简介：如何创建数组？

```
char name[40];
```

该声明语句创建一个有 40 个存储单元的数组，其中每个单元可存储一个 char 型值。

- ▶ 方括号说明 name 是一个数组
- ▶ 方括号中的 40 指出数组中的元素个数
- ▶ char 标识每个元素的类型

字符串简介：字符串的使用

要使用字符串，必须创建一个数组，把字符串中的字符逐个放入数组中，最后还需在结尾添加一个空字符 `\0`。

字符串简介：字符串的使用

```
1 // praise1.c
2 #include <stdio.h>
3 #define PRAISE "What a super marvelous name!"
4 int main(void)
5 {
6     char name[40];
7
8     printf("What's your name?\n");
9     scanf("%s", name);
10    printf("Hello, %s. %s\n", name, PRAISE);
11    return 0;
12 }
```


字符串简介：字符串的使用

```
$ gcc praise1.c
```

```
$ ./a.out
```

```
What's your name?
```

```
Xiaoping Zhang
```

```
Hello, Xiaoping. What a super marvelous name!
```

字符串简介：字符串的使用

关于 scanf() 函数

- ▶ 无须把空字符插入 name 数组中，scanf 会在读取输入时完成此任务。
- ▶ name 前无须加 &，因为 name 本身就是地址。
- ▶ 使用 %s 的 scanf 语句会在遇到的第一个空格、制表符或换行符处停止读取，它只会把第一个单词而不是把整条语句作为字符串读入。

因此，该程序只读取了 Xiaoping。

字符串简介：字符与字符串

请注意"x" 与'x' 的差别

- ▶ 'x' 为字符，而"x" 为字符串
- ▶ "x" 由两个字符'x' 和'\0' 组成

字符串简介：strlen 函数

- ▶ 对 sizeof 运算符，以字节为单位给出数据大小；
- ▶ 对 strlen 函数，以字符为单位给出字符串的长度，不包含空字符。

字符串简介：strlen 函数 I

```
1 // praise2.c:
2 #include <stdio.h>
3 #include <string.h>
4 #define PRAISE "What a super marvelous name!"
5 int main(void)
6 {
7     char name[40];
8     printf("What's your name?\n");
9     scanf("%s", name);
10    printf("Hello, %s. %s\n", name, PRAISE);
```

字符串简介：strlen 函数 II

```
11  printf("Your name of %lu letters occupied %lu  
memory cells.\n", strlen(name), sizeof name);  
12  printf("The phrase of PRAISE has %lu letters",  
    strlen(PRAISE));  
13  printf(" and occpied %lu memory cells.\n",  
    sizeof PRAISE);  
14  return 0;  
15 }
```

字符串简介：strlen 函数

```
$ gcc praise2.c
$ ./a.out
What's your name?
Xiaoping
Hello, Xiaoping. What a super marvelous name!
Your name of 8 letters occupied 40 memory cells.
The phrase of PRAISE has 28 letters and occupied
29 memory cells.
```

字符串简介：strlen 函数

- ▶ 头文件 `string.h` 包含许多与字符串相关的函数的原型，包括 `strlen` 函数。
- ▶ C 把函数库分成多个相关函数的序列，并为每个序列提供一个头文件。比如：
 - (1) `printf` 和 `scanf` 属于标准输入输出序列，使用 `stdio.h`。
 - (2) `strlen` 和其它一些与字符串相关的函数同属一个系列，使用 `string.h`。

字符串简介：printf 函数处理长字符串

- ▶ 一条 printf 语句占用两行，但只能在参数之间断行，不允许在字符串中间断行。
- ▶ 使用两个 printf 语句输出一行，换行符只出现在第二条语句。

字符串简介：sizeof 运算符与 strlen 返回值

设 `name = "Morgan"`, 则

```
sizeof name : 40
```

```
strlen(name) : 6
```

M	o	r	g	a	n	\0							
---	---	---	---	---	---	----	--	--	--	--	--	--	--

字符串简介：sizeof 运算符与 strlen 返回值

```
sizeof PRAISE : 29
```

```
strlen(PRAISE): 28
```

sizeof 运算符在处理字符串变量时，会将空字符也计算在内。

字符串简介：sizeof 运算符后的圆括号

- ▶ 圆括号对于数据类型是必需的，而对于具体量则是可选的。

```
sizeof(float)
```

```
sizeof(char)
```

```
sizeof name
```

```
sizeof 2.15
```

- ▶ 建议在所有情况下都使用圆括号。

```
sizeof(name)
```

```
sizeof(2.15)
```

2. 常量与预处理器

常量与预处理器

```
1 // circle1.c:
2 #include <stdio.h>
3 int main(void)
4 {
5     float radius, circum, area;
6     radius = 1;
7     area = 3.1415926 * radius * radius;
8     circum = 2 * 3.1415926 * radius;
9     printf("radius = %f, circum = %f, area = %f\n"
10           , radius, circum, area);
11     return 0;
12 }
```

常量与预处理器

```
radius = 1.000000, circum = 6.283185, area =  
3.141593
```

常量与预处理器

```
1 // circle2.c:
2 #include <stdio.h>
3 int main(void) {
4     float radius, circum, area;
5     float pi = 3.1415926;
6     radius = 1;
7     area = pi * radius * radius;
8     circum = 2 * pi * radius;
9     printf("radius = %f, circum = %f, area = %f\n"
10           , radius, circum, area);
11     return 0;
12 }
```


常量与预处理器

```
1 // circle3.c:
2 #include <stdio.h>
3 #define PI 3.1415926
4 int main(void) {
5     float radius, circum, area;
6     radius = 1;
7     area = PI * radius * radius;
8     circum = 2 * PI * radius;
9     printf("radius = %f, circum = %f, area = %f\n"
10           , radius, circum, area);
11     return 0;
12 }
```

常量与预处理器

```
1 // circle4.c:
2 #include <stdio.h>
3 int main(void) {
4     float radius, circum, area;
5     const float PI = 3.1415926;
6     radius = 1.;
7     area = PI * radius * radius;
8     circum = 2 * PI * radius;
9     printf("radius = %f, circum = %f, area = %f\n"
10           , radius, circum, area);
11     return 0;
12 }
```

常量与预处理器：宏定义

宏定义的一般形式

```
#define NAME value
```

- ▶ 没有使用分号是因为这是一种替代机制，而不是 C 的语句。
- ▶ 符号常量请使用大写，其好处在于当看到它时便可立即知道是常量。
- ▶ 符号常量的命名请遵循变量命名规则。

常量与预处理器：宏定义

`#define` 语句也可用于定义字符和字符串变量，前者用单引号，后者用双引号。

```
#define BEEP '\a'
```

```
#define TEE 'T'
```

```
#define ESC '\033'
```

```
#define OOPS "Now you have done it!"
```

常量与预处理器：宏定义

`#define` 语句也可用于定义字符和字符串变量，前者用单引号，后者用双引号。

常量与预处理器：宏定义

`#define` 语句也可用于定义字符和字符串变量，前者用单引号，后者用双引号。

常见错误

```
#define B = 20
```

如果这样做，`B` 将会被 `= 20` 而不是 `20` 代替。这样以下语句

```
c = a + B;
```

会被替换成如下错误的表达：

```
c = a + = 20;
```

常量与预处理器：const 修饰符

C90 允许使用关键字 `const` 把一个变量声明转换为常量声明：

```
const int MONTHS = 12;
```

这使得 `MONTHS` 成为一个只读值。你可以显示它，并把它用于计算中，但不能改变它的值。

3. 格式化输出

格式化输出

表: 格式说明符

格式说明符	输出
%a	浮点数、十六进制和 p-计数法
%A	浮点数、十六进制和 P-计数法
%c	一个字符
%d	有符号十进制数

格式化输出

表: 格式说明符

格式说明符	输出
%e	浮点数、e-计数法
%E	浮点数、E-计数法
%f	浮点数、十进制计数法
%g	根据数值不同自动选%f 或%e。%e 格式在指数小于-4 或大于等于精度时使用
%G	根据数值不同自动选%f 或%E。%E 格式在指数小于-4 或大于等于精度时使用

格式化输出

表: 格式说明符

格式说明符	输出
%i	有符号十进制整数 (同%d)
%o	无符号八进制整数
%p	指针
%s	字符串

格式化输出

表: 格式说明符

格式说明符	输出
%x	使用十六进制数字 0-f 的无符号十六进制整数
%X	使用十六进制数字 0-F 的无符号十六进制整数

格式化输出

printf 的使用格式

```
printf(Control-string, item1, item2, ...);
```

- ▶ item1, item2 等是要打印的项目，它们可以是变量，也可以是常量，甚至是在打印之前进行计算的表达式。
- ▶ 控制字符串 (Control-string) 是一个描述项目如何打印的字符串，它为每个要打印的项目包含一个格式说明符。

格式化输出

printf("You look great in %s\n",	color);
---------	---------------------------	-------	----

控制描述

变量列表

格式化输出

不要忘记给控制字符串后面的列表中的每个项目都使用一个格式说明符。

格式化输出

- ▶ 如果只想打印一个语句，则不需要任何格式说明符；
- ▶ 如果只想打印数据，则无须加入任何说明内容。
- ▶ 想打印%，必须使用两个%% 符号。

```
printf("Once more you open the door!\n");  
printf("%s%d\n", "area = ", area);  
printf("%d%% = %f\n", 30, 0.3);
```


格式化输出：格式说明符%d

- ▶ %d: 按整型数据的实际长度输出
- ▶ %md: 输出字段的宽度为 m, 右对齐
若数据位数 < m, 左端补空格; 若 $\geq m$, 按实际位数输出。
- ▶ %-md: 输出字段的宽度为 m, 左对齐
若数据位数 < m, 右端补空格; 若 $\geq m$, 按实际位数输出。
- ▶ %0md: 输出字段的宽度为 m, 右对齐
若数据位数 < m, 右端补 0; 若 $\geq m$, 按实际位数输出。

格式化输出：格式说明符%d

```
1 // width.c:
2 #include <stdio.h>
3 #define N 1000
4 int main(void)
5 {
6     printf("%d*\n", N);
7     printf("%2d*\n", N);
8     printf("%10d*\n", N);
9     printf("%-10d*\n", N);
10    printf("%010d*\n", N);
11    return 0;
12 }
```

格式化输出：格式说明符%d

```
*1000*
```

```
*1000*
```

```
*_____1000*
```

```
*1000_____*
```

```
*00000001000*
```

格式化输出：格式说明符%f、%e 和%E

```
1 // floats.c:
2 #include <stdio.h>
3 int main(void)
4 {
5     const double RENT = 3852.99;
6     printf("%f\n", RENT);
7     printf("%e\n", RENT);
8     printf("%4.2f\n", RENT);
9     printf("%3.1f\n", RENT);
10    printf("%10.3f\n", RENT);
11    printf("%10.3e\n", RENT);
```

格式化输出：格式说明符%f、%e 和%E II

```
12  printf("%10.3E*\n", RENT);  
13  printf("%+4.2f*\n", RENT);  
14  printf("%-10.2f*\n", RENT);  
15  printf("%010.2f*\n", RENT);  
16  return 0;  
17 }
```

格式化输出：格式说明符%f、%e 和%E

3852.990000

3.852990e+03

3852.99

3853.0

_3852.990

_3.853e+03

_3.853E+03

+3852.99

3852.99_

0003852.99

格式化输出：格式说明符%f、%e 和%E

`%m.nf`

`%m.ne`

`%m.nE`

- ▶ m 为字段宽度
- ▶ n 为小数点右边数字的个数

格式化输出：格式说明符%f、%e 和%E

`%.nf`

- ▶ 整数部分以实际长度输出
- ▶ n 为小数点右边数字的个数

格式化输出：格式说明符%f、%e 和%E

`%m.f`

- ▶ 字段宽度为 m
- ▶ 不输出小数点后的数字

格式化输出 I

```
1 // flags.c:
2 #include <stdio.h>
3 int main(void)
4 {
5     printf("%x %X %#x %#X\n", 31, 31, 31, 31);
6     printf("*%d*\n", 42);
7     printf("*% d*\n", 42);
8     printf("*% d*\n", -42);
9     printf("*%5d*\n", 6);
10    printf("*%5.3d*\n", 6);
11    printf("*%05d*\n", 6);
```

格式化输出 II

```
12     printf("%05.3d*\n", 6);  
13     return 0;  
14 }
```

格式化输出

```
1f_1F_0x1f_0X1F
```

```
*42*
```

```
*_42*
```

```
*-42*
```

```
*____6*
```

```
*__006*
```

```
*00006*
```

```
*__006*
```

- ▶ 使用 `%x` 输出 `1f`
- ▶ 使用 `%X` 输出 `1F`
- ▶ 使用 `%#x` 输出 `0x1f`
- ▶ 使用 `%#X` 输出 `0X1F`

使用% d 在正值之前产生一个前导空格，在负值之前不产生前导空格。这使得有效位相同的正值和负值以相同字段宽度打印输出。

- ▶ `%5.3d` 为精度说明符，用于在整数格式中来产生足够的前导零以填满要求的最小数字位数。
- ▶ `%05d` 将会用前导零填满整个字段宽度。
- ▶ 在 `%05.3d` 中，`0` 标志和精度说明符同时出现，此时 `0` 标志将会忽略。

格式化输出：关于字符串的打印

```
1 // strings.c:
2 #include <stdio.h>
3 #define WORD "Hello World!"
4 int main(void)
5 {
6     printf("%2s*\n", WORD);
7     printf("%15s*\n", WORD);
8     printf("%15.5s*\n", WORD);
9     printf("%*-15.5s*\n", WORD);
10    return 0;
11 }
```


格式化输出：关于字符串的打印

```
*Hello_World!*
```

```
*__Hello_World!*
```

```
*_____Hello*
```

```
*Hello_____*
```

格式化输出：关于字符串的打印

`%15.5s` 为精度说明符，告诉 `printf` 函数只打印 5 个字符。修饰符‘-’使文本左对齐输出。

格式化输出：printf() 的返回值

```
1 #include <stdio.h>
2 int main(void)
3 {
4     int bph2o = 100;
5     int rv;
6     rv = printf("%d C is water's boiling point.\n",
7     bph2o);
8     printf("the printf function printed %d
9     character.\n", rv);
10    return 0;
11 }
```

格式化输出：printf() 的返回值

```
100 C is water's boiling point.  
the printf function printed 32 character.
```

格式化输出：printf() 的返回值

printf() 返回所有打印字符的个数，包括空格和不可见的换行字符。

格式化输出：printf() 中的%n

在 `printf()` 中，`%n` 是一个格式化说明符，它将获取 `%n` 出现之前的所有字符的个数，并将其传递给后面对应的变量。

格式化输出：printf() 中的%n

```
1 // printf_n.c:
2 #include<stdio.h>
3 int main(void)
4 {
5     int c1, c2;
6     printf("Hello Wuhan %nUniversity!%n\n", &c1, &
7         c2);
8     printf("c1 = %d, c2 = %d\n", c1, c2);
9     return 0;
10 }
```

格式化输出：printf() 中的%n

```
$ gcc printf_n.c  
$ ./a.out  
Hello Wuhan University!  
c1 = 12, c2 = 23
```


4. 格式化输入

格式化输入

同 `printf()` 一样, `scanf()` 也使用控制字符串和参数列表, 主要区别在参数列表。`printf()` 使用变量名、常量和表达式; 而 `scanf()` 使用指向变量的指针。

格式化输入：scanf() 的参数列表

- ▶ 若使用 `scanf()` 来读取某种基本类型的值，请在变量名前加一个 `&`。
- ▶ 若使用 `scanf()` 把一个字符串读入一个字符数组，请不要使用 `&`。

格式化输入：scanf() 的参数列表

```
1 // input.c:
2 #include <stdio.h>
3 int main(void) {
4     int age;
5     double weight;
6     char name[20];
7     printf("Enter your name, age and weight:\n");
8     scanf("%s", name);
9     scanf("%d and %lf", &age, &weight);
10    printf("%s: %d %f\n", name, age, weight);
11    return 0;
12 }
```

格式化输入：scanf() 的参数列表

```
$ gcc input.c  
$ ./a.out  
Enter your name, age and weight:  
Xiaoming  
23 100  
Xiaoming: 23 100.000000
```

格式化输入：scanf() 的参数列表

- ▶ `scanf()` 使用空格（换行、制表符和空格）来决定如何把输入分成几个字段。它依次把格式说明符与字段相匹配，并跳过它们之间的空格。
- ▶ 也可以分一行或多行输入，只要每个输入项目之间至少有一个换行符、空格或制表符。
- ▶ `%c` 是个例外，即使下一个字符是空白字符，它也会读取。

格式化输入：printf 与 scanf 格式说明符的区别

- ▶ `printf()` 把 `%f`、`%e`、`%E`、`%g` 和 `%G` 同时用于 `float` 和 `double` 类型
- ▶ `scanf()` 只把它们用于 `float` 类型，而用于 `double` 类型时要求加上 `l` 修饰符。

格式化输入：scanf 格式说明符

格式说明符	意义
%c	把输入解释成一个字符
%d	把输入解释成一个有符号十进制数
%e, %f, %g, %a	把输入解释成一个浮点数
%E, %F, %G, %A	把输入解释成一个浮点数
%i	把输入解释成一个有符号十进制数
%o	把输入解释成一个有符号八进制数
%p	把输入解释成一个指针

格式化输入：scanf 格式说明符

格式说明符	意义
%s	把输入解释成一个字符串：输入内容以第一个非空白字符作为开始，并且包含到下一个空白字符的全部字符
%u	把输入解释成一个无符号十进制数
%x, %X	把输入解释成一个有符号十六进制数

格式化输入：scanf 格式说明符

可在格式说明符中使用修饰符，修饰符出现在% 与格式字符之间。

格式化输入：scanf 格式说明符

修饰符	意义
*	滞后赋值, 如"%*d"
digit	最大字段宽度: 在达到最大字段宽度或遇到第一个空白字符时停止对输入项的读取, 如"%10s"
hh	把整数读作 signed char 或 unsigned char, 如"%hhd" 或"%hhu"
ll	把整数读作 long long 或 unsigned long long, 如"%lld" 或"%llu"

格式化输入：scanf 格式说明符

修饰符	意义
"%hd", "%hi"	以 short 存储
"%ho", "%hx", "%hu"	以 unsigned short 存储
"%ld", "%li"	以 long 存储
"%lo", "%lx", "%lu"	以 unsigned long 存储
"%le", "%lf", "%lg"	以 double 存储
"%Le", "%Lf", "%Lg"	以 long double 存储

格式化输入：scanf 格式说明符

若没有这些修饰符，则 `%d`，`%i`，`%o` 和 `%x` 指示 `int` 类型，而 `%e`，`%f` 和 `%g` 指示 `float` 类型。

格式化输入：格式字符串中的常规字符

`scanf()` 允许把普通字符放在格式字符串中，除了空格字符之外的普通字符一定要与输入字符串准确匹配。

格式化输入：格式字符串中的常规字符

```
scanf ("%d, %d", &n, &m);
```

合法的输入方式

12, 23

12, _ _ _ _ 23

12_ , _ 23

12,

23

格式化输入：格式字符串中的常规字符

```
scanf ("%d_and_%d", _&n, _&m);
```

合法的输入方式

12_and_23

12_and_23

12and23

格式化输入：格式字符串中的常规字符

除了%c 之外的说明符会自动跳过输入项之前的空格，故以下两条语句的效果相同：

```
scanf ("%d%d", &n, &m) ;
```

```
scanf ("%d_ %d", &n, &m) ;
```

格式化输入：格式字符串中的常规字符

对于%c 来说，向格式字符串中添加一些空格将导致一些差别。如：

```
scanf ("%c", &ch)
```

读取在输入中遇到的第一个字符，而

```
scanf ("%c", &ch)
```

则读取遇到的第一个非空白字符。

格式化输入：scanf 的返回值

scanf() 返回成功读入的项目个数。

- ▶ 若没有读取任何项目，则返回 0；
- ▶ 若检测到文件结尾 (end of file)，则返回 EOF。
(EOF 是 `stdio.h` 中定义的特殊值，一般为 -1)

格式化输入：printf() 的 * 修饰符 I

```
1 // varwidth.c:
2 #include <stdio.h>
3 int main(void)
4 {
5     unsigned width, precision;
6     int number = 256;
7     double weight = 123.5;
8     printf("What field width?\n");
9     scanf("%d", &width);
10    printf("The number is:%*d\n", width, number);
```

格式化输入：printf() 的 * 修饰符 II

```
11  printf("Now enter a width and a precision:\n")  
    ;  
12  scanf("%d %d", &width, &precision);  
13  printf("Weight=%*.*f\n", width, precision,  
    weight);  
14  return 0;  
15 }
```

格式化输入：printf 的 * 修饰符

```
$_gcc_varwidth.c
```

```
$_./a.out
```

```
What_field_width?
```

```
6
```

```
The_number_is:___256
```

```
Now_enter_a_width_and_a_precision:
```

```
8_3
```

```
Weight=_123.500
```

格式化输入：scanf() 的 * 修饰符

在 `scanf()` 中，把 `*` 放在 `%` 与格式字符之间时，会使函数跳过相应的输入项目。

格式化输入：scanf() 的 * 修饰符

```
1 // skip2.c:
2 #include <stdio.h>
3 int main(void)
4 {
5     int n;
6     printf("Please enter three integers:\n");
7     scanf("%*d %*d %d", &n);
8     printf("The last integer was %d\n", n);
9     return 0;
10 }
```


格式化输入：scanf() 的 * 修饰符

```
$_gcc_skip2.c  
$_./a.out  
Please_enter_three_integers:  
10_20_30  
The_last_integer_was_30
```