

We'll hear and say more.



廈門大學
XIAMEN UNIVERSITY



信息学院 黃 煒
(特色化示范性软件学院) 博士·副教授
School of Informatics Wei Huang

4

字符串和 格式化输入输出

理论课程



廈門大學
XIAMEN UNIVERSITY



信息学院 黄 焯
(特色化示范性软件学院) 博士, 副教授
School of Informatics Wei Huang

知识框架

- 利用数组解决问题
- 字符串：有格式的数组
- 格式化输出 printf()
- 输出函数的常见错误
- 格式化输入 scanf()
- 输入函数的常见错误

内容纲要

| | |
|---|----------------|
| 1 | 利用数组解决问题 |
| 2 | 字符串：有格式的数组 |
| 3 | 格式化输出 printf() |
| 4 | 输出函数的常见错误 |
| 5 | 格式化输入 scanf() |

实例

- 题目

- 设输入10位学生的成绩，计算平均值

- 样例输入

- 90.5 60.2 （此处略去7个实数） 85.3

- 样例输出

- Average score is: 81.620

- 最直接的想法

- 声明10个变量，依次相加，除以10

```
// score_average.c -- calculate the average score
#include <stdio.h>
int main()
{
    float score, score_1, score_2, score_3, score_4, score_5,
    score_6, score_7, score_8, score_9;
    float score_average;
    scanf("%f", &score);
    scanf("%f", &score_1);
    scanf("%f", &score_2);
    scanf("%f", &score_3);
    scanf("%f", &score_4);
    scanf("%f", &score_5);
    scanf("%f", &score_6);
    scanf("%f", &score_7);
    scanf("%f", &score_8);
    scanf("%f", &score_9);
    score_average = (score + score_1 + score_2 + score_3 +
    score_4 + score_5 + score_6 + score_7 + score_8 + score_9) / 10;
    printf("Average score is: %f\n", score_average);
    return 0;
}
```

设明年学生有11人，没法用了？
设明年学生有30000人，没法写了？

学生人数和计算流程是无
关的，不应该为每个人数
设计一个新的程序

```
// score_average.c -- calculate the average score
```

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

```
int main()
```

```
{
```

```
    float score[10];
```

```
    float score_average;
```

```
    scanf("%f", &score[0]);
```

```
    scanf("%f", &score[1]);
```

```
    scanf("%f", &score[2]);
```

```
    scanf("%f", &score[3]);
```

```
    scanf("%f", &score[4]);
```

```
    scanf("%f", &score[5]);
```

```
    scanf("%f", &score[6]);
```

```
    scanf("%f", &score[7]);
```

```
    scanf("%f", &score[8]);
```

```
    scanf("%f", &score[9]);
```

```
    score_average = (score[0] + score[1] + score[2] +  
score[3] + score[4] + score[5] + score[6] + score[7] +  
score[8] + score[9]) / 10;
```

```
    printf("Average score is: %f\n", score_average);
```

```
    return 0;
```

```
}
```

数组的声明：

类型 数组名[数组长度];

设明年学生数量有变化，修改
方括号里面的数字即可。

数组的使用：

数组名[下标]，s[2]即 s_2

只用数组处理问题，仍然不够
简洁，需要将相近的流程合并

```
// score_average.c -- calculate the average score
```

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

```
int main()
```

```
{
```

```
    float score[10];
```

```
    float score_average;
```

```
    for (unsigned int i = 0; i < 10; i++)
```

```
    {
```

```
        scanf("%f", &score[i]);
```

```
    }
```

```
    score_average = 0;
```

```
    for (unsigned int i = 0; i < 10; i++)
```

```
    {
```

```
        score_average += score[i];
```

```
    }
```

```
    score_average /= 10;
```

```
    printf("Average score is: %f\n", score_average);
```

```
    return 0;
```

```
}
```

学生人数变了，改程序还是较为麻烦

下标是一个无符号整数，C99允许其在for首部声明

数组配合循环，解决有规律的问题

将有规律的环节用循环表示

这几个10是一样的吗？
改一个是否要改另一个？


```
// score_average.c -- calculate the average score
```

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

```
#define SCORE_LENGTH 10
```

利用宏定义，人数变更时
修改这一处定义即可

```
int main()
```

```
{
```

```
    float score[SCORE_LENGTH];
```

```
    float score_average;
```

阅读程序的程序员，理解这几处
10是有联系的，理解具体含义。

```
    for (unsigned int i = 0; i < SCORE_LENGTH; i++)
```

```
        scanf("%f", &score[i]);
```

```
    score_average = 0;
```

```
    for (unsigned int i = 0; i < SCORE_LENGTH; i++)
```

```
        score_average += score[i];
```

```
    score_average /= SCORE_LENGTH;
```

```
    printf("Average score is: %6.3f\n", score_average);
```

注意输出的格式

```
    return 0;
```

```
}
```

```
// score_average.c -- calculate the average score
#include <stdio.h>
#define SCORE_LENGTH 10
int main()
{
    float score[SCORE_LENGTH];
    float score_average = 0;
    for (unsigned int i = 0; i < SCORE_LENGTH; i++)
    {
        scanf("%f", &score[i]);
        score_average += score[i];
    }
    score_average /= SCORE_LENGTH;
    printf("The average score is %6.3f.\n", score_average);
    return 0;
}
```

两个for循环临近，
有合并的可能性

数组基础知识

- 数组的声明

- 格式

```
<类型> 数组名[数组长度];
```

- 示例

```
float score[SCORE_LENGTH];
```

- 数组长度应为非负整数，一般地，不小于3。

- 数组元素调用

- 格式

```
数组名[元素下标]
```

- 示例

```
score[1]
```

- 数组元素下标应为非负整数，小于数组长度。
 - 数组元素下标为偏移量，第一个元素下标为0。

内容纲要

| | |
|---|----------------|
| 1 | 利用数组解决问题 |
| 2 | 字符串：有格式的数组 |
| 3 | 格式化输出 printf() |
| 4 | 输出函数的常见错误 |
| 5 | 格式化输入 scanf() |

字符串常量

- 字符串特指由可见字符组成以0为结尾的数组
 - 字符串常量以一对双引号 " 为起止
 - 期间的转义字符 \" 除外
 - 以可见字符或转义字符的组成
 - 转义字符以反斜杠 \ 开始，紧随1个或多个字符，表示特殊字符

• 字符串的内存存储

"Hello world!"

| | | | | | | | | | | | | |
|---|---|---|---|---|--|---|---|---|---|---|---|----|
| H | e | l | l | o | | w | o | r | l | d | ! | \0 |
|---|---|---|---|---|--|---|---|---|---|---|---|----|

"Hello \" world!"

| | | | | | | | | | | | | | | |
|---|---|---|---|---|--|---|--|---|---|---|---|---|---|----|
| H | e | l | l | o | | " | | w | o | r | l | d | ! | \0 |
|---|---|---|---|---|--|---|--|---|---|---|---|---|---|----|

字符串基础知识

• 字符串的声明

| 格式 | 赋值语句 |
|--------------------|------------------------------|
| char 数组名[长度]; | char c[60]; |
| char 数组名[长度]="内容"; | char c[60] = "Hello world!"; |
| char 数组名[]="内容"; | char c[] = "Hello world!"; |
| char *数组名="内容"; | char *c = "Hello world!"; |

- 长度应至少为字符串字符数加1。

不可赋值

• 字符串的调用

| 功能 | 格式 | 表达式 |
|---------|---------|------|
| 表示字符串整体 | 数组名 | c |
| 表示字符串元素 | 数组名[下标] | c[3] |

```

// talkback.c -- nosy, informative program
#include <stdio.h>
#include <string.h>
#define DENSITY 62.4
int main()
{
    float weight, volume;
    int size, letters;
    char name[40];
    printf("Hi! What's your first name?\n");
    scanf("%s", name);
    printf("%s, what's your weight in pounds?\n", name);
    scanf("%f", &weight);
    size = sizeof name;
    letters = strlen(name);
    volume = weight / DENSITY;
    printf("Well, %s, your volume is %2.2f cubic feet.\n",
name, volume);

```

字符串函数的定义在这里

密度 (DENSITY) 是62.4的物理含义，有助于理清思路

字符串是字符数组

// name is an array of 40 chars

输入字符串的方法

求数组的空间

使用strlen()求字符串的长度

打印字符串

```
printf("Also, your first name has %d letters,\n",  
letters);  
printf("and we have %d bytes to store it.\n", size);  
  
return 0;  
}
```

这里只能输入一个单词，多于一个单词将产生结果不可控。

Hi! What's your first name?

Sharla↓

Sharla, what's your weight in pounds?

139↓

Well, Sharla, your volume is 2.23 cubic feet.

Also, your first name has 6 letters,
and we have 40 bytes to store it.

内容纲要

| | |
|---|----------------|
| 1 | 利用数组解决问题 |
| 2 | 字符串：有格式的数组 |
| 3 | 格式化输出 printf() |
| 4 | 输出函数的常见错误 |
| 5 | 格式化输入 scanf() |

格式输出函数 printf()

• 声明语句格式

```
int printf( const char *format [, argument]... );
```

— 参数

- format：格式字符串，指定后续参数的显示格式
 - 格式字符串中包含格式说明符（以 % 开头）
- argument, ...：不定长的参数列表

— 返回值

- 最终输出的字符个数

• 示例

格式说明符

格式字符串

参数

```
printf("The %d contestants ate %f berry pies.\n", number, pies);
```

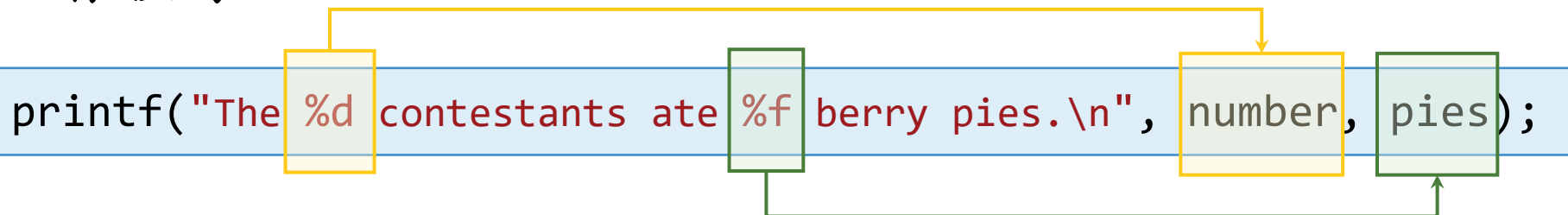
函数 printf() 原理

- 从函数第一个参数获取格式字符串
- 从函数后续参数取值，依次按格式说明符生成字符串
- 将值生成的字符串替换原格式字符串部分
- 将替换后的字符串显示在标准输出流中
 - 默认的标准输出流是显示器，但可以更改

函数 printf() 原理

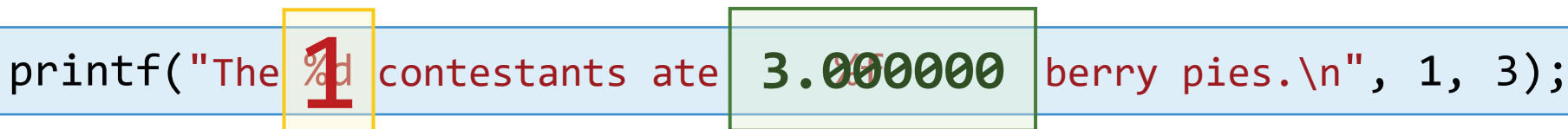
• 替换原理

`printf("The %d contestants ate %f berry pies.\n", number, pies);`



- 从左向右读取格式字符串，找出所有格式标识符。
- 遇到格式标识符时，计算对应参数的值，转换为指定格式。
 - 如果具有比格式说明更多的参数，则忽略额外的参数。
 - 如果没有足够的参数，则结果视编译器而定。

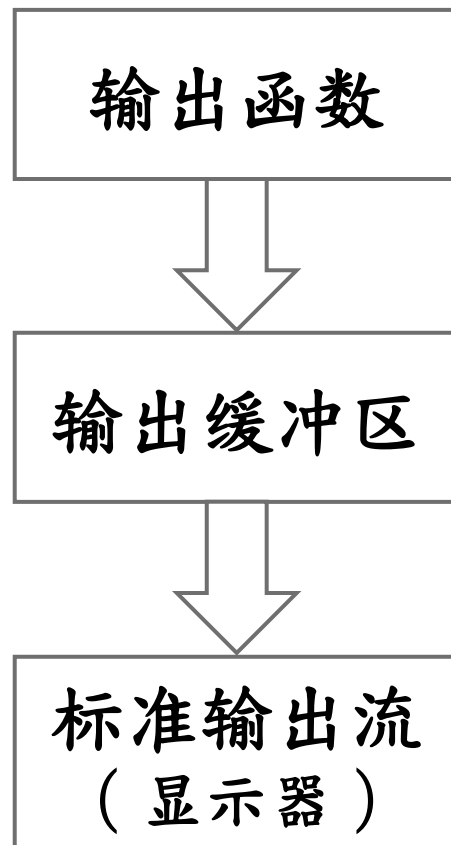
`printf("The 1 contestants ate 3.000000 berry pies.\n", 1, 3);`



• 使用 printf() 函数应有 `#include <stdio.h>` 指令

输出缓冲区

- 缓冲区是一段内存区域
- 输出函数将字符按顺序写入缓冲区
- 直至缓冲区释放时按顺序写入输出流
 - 输出换行符将释放缓冲区
- 标准输出流是显示器
 - 输出流可以重新定向到其它设备



```

/* printout.c -- uses conversion specifiers */
#include <stdio.h>
#define PI 3.141593
int main(void)
{
    int number = 7;
    float pies = 12.75;
    int cost = 7800;

    printf("The %d contestants ate %f berry pies.\n", number,
pies);
    printf("The value of pi is %f.\n", PI);
    printf("Farewell! thou art too dear for my possessing,\n");
    printf("%c%d\n", '$', 2 * cost);
    -----
    return 0;
}

```

可以使用多个格式化标记

格式输出函数的格式说明

- 格式字符串中包含若干个格式说明

Times字体的方括号表示该内容可选且可忽略

- 格式说明的形式：

%[标志][宽度][.精密度][修饰符]类型

— 示例：

%09.51f

001.70000

| 成分 | 可选 | 说明 |
|-----|----|--|
| 标志 | 可选 | 用于控制符号、空白、前导零点、小数点、八进制和十六进制前缀的输出对齐和输出。 |
| 宽度 | 可选 | 十进制整数，指定输出的最小字符数。 |
| 精密度 | 可选 | 它指定为字符串打印的最大字符数、浮点值小数点字符之后的有效位数或数字位数，或为整数值打印的最小位数。 |
| 类型 | 必选 | 确定相关联的参数被解释为字符、字符串、整数或浮点数。 |

格式输出函数的格式说明

• 主要的标志说明

| 标志 | 含义 | 默认 |
|----|---|----------|
| 0 | 添加前导零直到达到最小宽度为止。 如果出现左对齐或精度说明，0被忽略。 | 不填充。 |
| - | 在给定的字段宽度对结果左对齐。 | 右对齐 |
| + | 正数显示+号，负数显示-号。 | 仅负值显示负号。 |
| # | 当它与o，x或X格式一起使用时，#标志分别使用0、0x、0X，来对任何非零输出值进行前缀。 | 不出现空白。 |
| 空格 | 正数显示空格，负数显示-号。 | 仅负值显示负号。 |

格式输出函数的格式说明

• 主要的类型说明

| 类型符 | 参数 | 输出格式 |
|-----|-----|--|
| d | 整数 | 有符号的十进制整数 |
| s | 字符串 | 显示字符串。 |
| f | 浮点 | 单精度浮点数。 |
| x或X | 整数 | 无符号的十六进制整数；x用“abcdef”；X用“ABCDEF”。 |
| c | 字符 | 字符（包括：单字节字符，宽字符）。 |
| e或E | 浮点 | 以指数形式输出浮点数，可指定指数部分宽度。 |
| u | 整数 | 无符号的十进制整数。 |
| g或G | 浮点 | 当值的指数小于-4或者大于或等于精度参数时使用e格式，否则使用f格式。G使用大写E。 |
| o | 整数 | 无符号的八进制整数 |
| p | 指针 | 显示参数为十六进制数字的地址。 |

```

/* bases.c--prints 100 in decimal, octal, and hex */
#include <stdio.h>
int main(void)
{
    int x = 100;

    printf("dec = %d; octal = %o; hex = %x\n", x, x, x);
    printf("dec = %d; octal = %#o; hex = %#x\n", x, x, x);

    return 0;
}

```

要显示0和0x前缀，必须在说明符中加入#符号。

```

dec = 100; octal = 144; hex = 64
dec = 100; octal = 0144; hex = 0x64

```

格式输出函数的格式说明

- 主要的修饰符说明

| 修饰符 | 含义 |
|-----|-----------------------------------|
| l | 表示(unsigned) long int和double类型 |
| ll | 表示(unsigned) long long int类型 |
| L | 表示(unsigned) long double类型 |
| z | 表示size_t类型 (若编译器不支持可以用%u代替%zd) |

格式输出函数的格式说明

- 宽度规格永远**不会**导致一个值被截断。
 - 如果转换的结果比指定值宽，展开包含强制转换结果。
- 宽度规格可以用星号 (*) 从参数列表读取

```
printf("%0*f", 5, 3);          /* "00003" is output */
```

- 精度规格**会**导致一个值被截断。

```
printf( "%.0d", 0 );          /* No characters output */
```

- 精度规格可以用星号 (*) 从参数列表读取

```
printf( "%.*f", 3, 3.14159265 ); /* "3.142" output */
```

```

/* width.c -- field widths */
#include <stdio.h>
#define PAGES 959
int main(void)
{
    printf("%d*\n", PAGES);
    printf("%2d*\n", PAGES);
    printf("%10d*\n", PAGES);
    printf("%-10d*\n", PAGES);

    return 0;
}

```

指定宽度，左右对齐

```

*959*
*959*
*          959*
*959          *

```

```
// floats.c -- some floating-point combinations
#include <stdio.h>
int main(void)
{
    const double RENT = 3852.99;    // const-style constant
    printf("%f*\n", RENT);
    printf("%e*\n", RENT);
    printf("%4.2f*\n", RENT);
    printf("%3.1f*\n", RENT);
    printf("%10.3f*\n", RENT);
    printf("%10.3E*\n", RENT);
    printf("%+4.2f*\n", RENT);
    printf("%010.2f*\n", RENT);
    return 0;
}
```

```
*3852.990000*
*3.852990e+003*
*3852.99*
*3853.0*
* 3852.990*
*3.853E+003*
*+3852.99*
*0003852.99*
```

```

/* flags.c -- illustrates some formatting flags */
#include <stdio.h>

int main(void)
{
    printf("%x %X %#x\n", 31, 31, 31);
    printf("**%d**% d**% d**\n", 42, 42, -42);
    printf("**%5d**%5.3d**%05d**%05.3d**\n", 6, 6, 6, 6);

    return 0;
}

```

```

1f 1F 0x1f
**42** 42**-42**
**      6**    006**00006**    006**

```

```

/* stringf.c -- string formatting */
#include <stdio.h>

#define BLURB "Authentic imitation!"

int main(void)
{
    printf("[%2s]\n", BLURB);
    printf("[%24s]\n", BLURB);
    printf("[%24.5s]\n", BLURB);
    printf("[%24.5s]\n", BLURB);
    return 0;
}

```

```

[Authentic imitation!]
[      Authentic imitation!]
[                               Authe]
[Authe                          ]

```



```
/* intconv.c -- some mismatched integer conversions */
```

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

```
#define PAGES 336
```

```
#define WORDS 65618
```

```
int main(void)
```

```
{
```

```
    short num = PAGES;
```

```
    short mnum = -PAGES;
```

```
    printf("num as short and unsigned short:  %hd %hu\n", num,  
num);
```

```
    printf("-num as short and unsigned short: %hd %hu\n",  
mnum, mnum);
```

```
    printf("num as int and char: %d %c\n", num, num);
```

```
    printf("WORDS as int, short, and char: %d %hd %c\n",  
WORDS, WORDS, WORDS);
```

```
    return 0;
```

```
}
```

```
num as short and unsigned short:  336 336  
-num as short and unsigned short: -336 65200  
num as int and char: 336 P  
WORDS as int, short, and char: 65618 82 R
```

```

/* floatcnv.c -- mismatched floating-point conversions */
#include <stdio.h>
int main(void)
{
    float n1 = 3.0;
    double n2 = 3.0;
    long n3 = 2000000000;
    long n4 = 1234567890;

    printf("%.1e %.1e %.1e %.1e\n", n1, n2, n3, n4);
    printf("%ld %ld %ld %ld\n", n1, n2, n3, n4);

    return 0;
}

```

```

3.0e+000 3.0e+000 3.1e+046 0.0e+000
0 1074266112 2000000000 1234567890

```

```
/* prntval.c -- finding printf()'s return value */
```

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

```
int main(void)
```

```
{
```

```
    int bph2o = 212;
```

```
    int rv;
```

格式化输出的返回值是输出的字符个数

```
    rv = printf("%d F is water's boiling point.\n", bph2o);
```

```
    printf("The printf() function printed %d characters.\n", rv);
```

```
    return 0;
```

```
}
```

212 F is water's boiling point.

The printf() function printed 32 characters.

```
/* longstrg.c -- printing long strings */
```

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

```
int main(void)
```

```
{
```

```
    printf("Here's one way to print a ");
```

```
    printf("long string.\n");
```

```
    printf("Here's another way to print a \n  
long string.\n");
```

```
    printf("Here's the newest way to print a "  
           "long string.\n");
```

```
/* ANSI C */
```

```
    return 0;
```

```
}
```

将一个字符串分行书写方便阅读代码，此时可以在分割处用反斜杠加回车表示，注意新行的字符串应顶格书写，否则会有多余的空格输出。

将一个字符串分行书写方便阅读代码，可以将其拆分成两个字符串相邻书写。

Here's one way to print a long string.

Here's another way to print a long string.

Here's the newest way to print a long string.

```
/* varwid.c -- uses variable-width output field */
```

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

```
int main(void)
```

```
{
```

```
    unsigned width, precision;
```

```
    int number = 256;
```

```
    double weight = 242.5;
```

```
    printf("Enter a field width:\n");
```

```
    scanf("%d", &width);
```

```
    printf("The number is :%*d:\n", width, number);
```

```
    printf("Now enter a width and a precision:\n");
```

```
    scanf("%d %d", &width, &precision);
```

```
    printf("Weight = %*.*f\n", width, precision, weight);
```

```
    printf("Done!\n");
```

```
    return 0;
```

```
}
```

Enter a field width:

15↓

The number is : 256:

Now enter a width and a precision:

5 3↓

Weight = 242.500

Done!

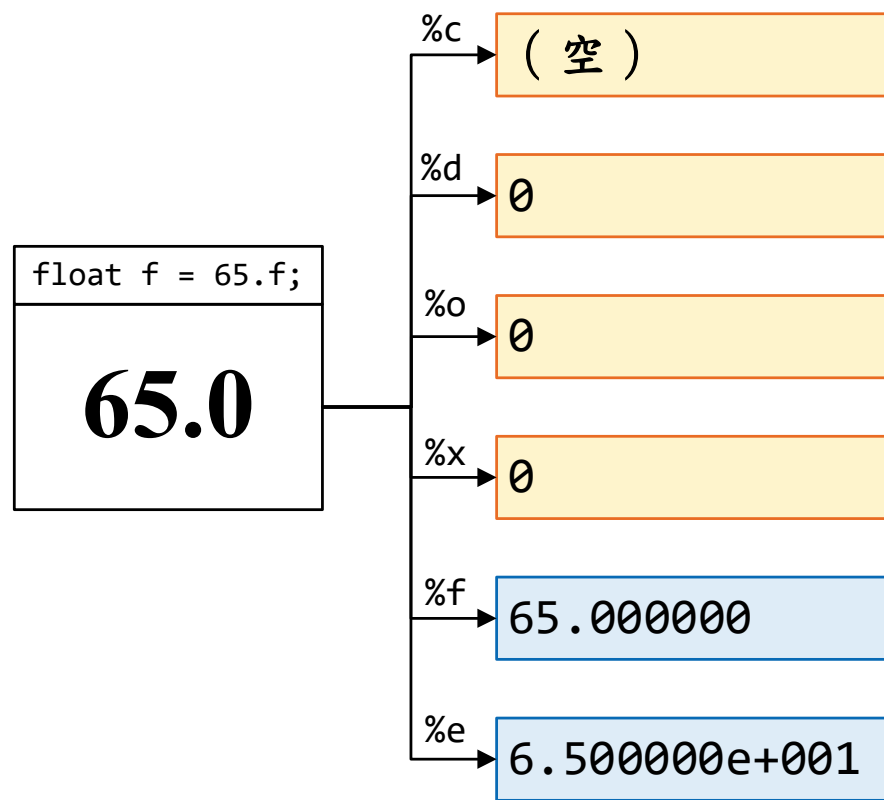
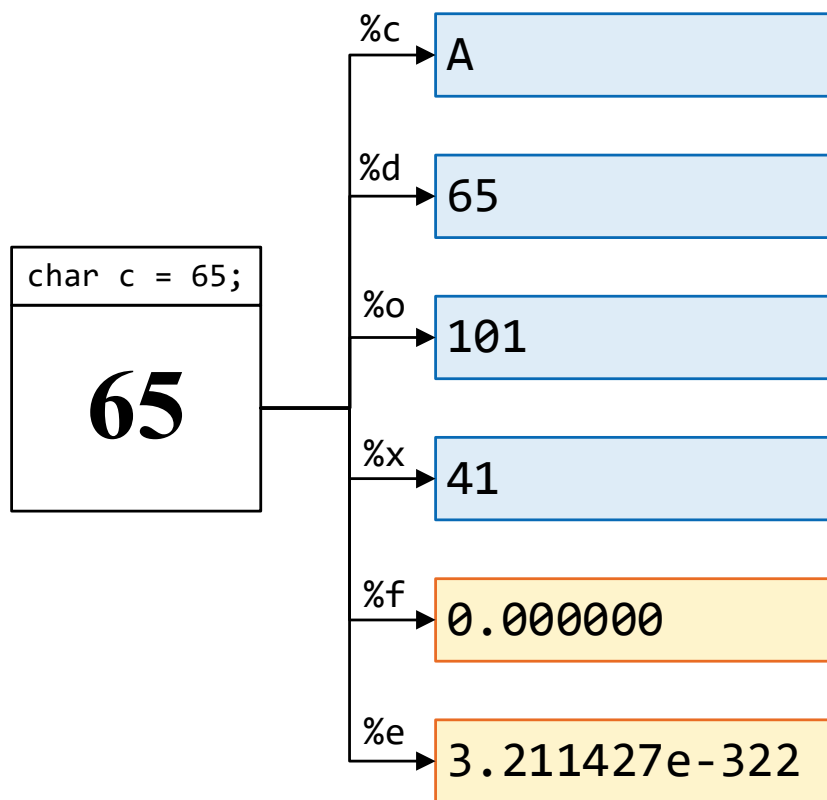
内容纲要

| | |
|---|----------------|
| 2 | 字符串：有格式的数组 |
| 3 | 格式化输出 printf() |
| 4 | 输出函数的常见错误 |
| 5 | 格式化输入 scanf() |
| 6 | 输入函数的常见错误 |

整型和浮点型说明符之间的误用

- 整型说明符显示浮点数，或浮点型说明符显示整数

```
printf("%f", c);
```



不同整型之间的误用

• 不同整型之间的误用

| 声明并赋值 | 值 | %d / %ld | %u / %lu | %lld | %llu |
|----------------------------|------------|------------|------------|------------|------------|
| char c = -1; | -1 | -1 | $2^{32}-1$ | $2^{32}-1$ | $2^{32}-1$ |
| short c = -1; | | | | | |
| int c = -1; | | | | | |
| unsigned char c = -1; | 2^8-1 | 2^8-1 | 2^8-1 | 2^8-1 | 2^8-1 |
| unsigned short c = -1; | $2^{16}-1$ | $2^{16}-1$ | $2^{16}-1$ | $2^{16}-1$ | $2^{16}-1$ |
| unsigned int c = -1; | $2^{32}-1$ | -1 | $2^{32}-1$ | $2^{32}-1$ | $2^{32}-1$ |
| long long c = -1; | -1 | -1 | $2^{32}-1$ | -1 | $2^{64}-1$ |
| unsigned long long c = -1; | $2^{64}-1$ | -1 | $2^{32}-1$ | -1 | $2^{64}-1$ |

说明符和参数个数不相等

- 参数个数如果少于转换说明个数，结果将不可预料

```
printf("%d %d %d\n", n); /* too few arguments */
```

- 参数个数如果多于转换说明个数，冗余部分无效

```
printf("%d\n", n, m); /* too many arguments */
```

将普通字符串置于第一个参数

- 将字符串直接置于第一个参数，可能出错
 - 第一个参数是格式化字符串专用
 - 如果该字符串包含%s之类的格式化符号

```
printf( name ); /* Danger! If name contains "%s", program will crash */
```

- 避免由软件用户输入格式化字符串
 - 用户不是专业人员，无法设计格式化字符串

```
/* badcount.c -- incorrect argument counts */
```

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

```
int main(void)
```

```
{
```

```
    int n = 4;
```

```
    int m = 5;
```

```
    float f = 7.0f;
```

```
    float g = 8.0f;
```

参数多了不替换，少了乱替换，类型错了乱替换

```
    printf("%d\n", n, m);    /* too many arguments */
```

```
    printf("%d %d %d\n", n); /* too few arguments */
```

```
    printf("%d %d\n", f, g); /* wrong kind of values */
```

```
    return 0;
```

```
}
```

```
4
4 5 134513906
0 1075576832
```

```

/* print1.c-displays some properties of printf() */
#include <stdio.h>
int main(void)
{
    int ten = 10;
    int two = 2;

    printf("Doing it right: ");
    printf("%d minus %d is %d\n", ten, 2, ten - two );
    printf("Doing it wrong: ");
    printf("%d minus %d is %d\n", ten ); // forgot 2 arguments

    return 0;
}

```

```

print1.c: In function 'main':
print1.c:11:5: warning: format '%d' expects a matching 'int' argument [-Wformat=]
    printf("%d minus %d is %d\n", ten ); // forgot 2 arguments
    ^
print1.c:11:5: warning: format '%d' expects a matching 'int' argument [-Wformat=]

```

运行结果

- 不同的编译器获得不同的结果
 - 不同编译器的处理方式不一样

Visual Studio (Debug)

Doing it right: 10 minus 2 is 8

Doing it wrong: 10 minus 0 is 0

Visual Studio (Relase)

Doing it right: 10 minus 2 is 8

Doing it wrong: 10 minus 20521256 is 20521236

Ubuntu GCC

Doing it right: 10 minus 2 is 8

Doing it wrong: 10 minus 2 is 8

内容纲要

| | |
|---|----------------|
| 3 | 格式化输出 printf() |
| 4 | 输出函数的常见错误 |
| 5 | 格式化输入 scanf() |
| 6 | 输入函数的常见错误 |
| 7 | 总结 |

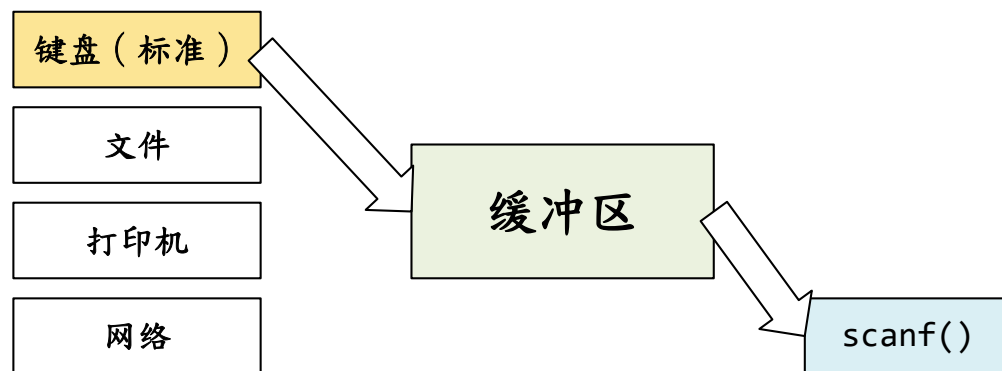
格式化输入的作用

- 格式化输入的作用

- 标准输入流字符按顺序写入缓冲区
- 格式化输入语句从缓冲区读出字符串
 - 如果缓冲区为空，程序暂停等待用户输入
 - 直至缓冲区释放，输入换行符将释放缓冲区

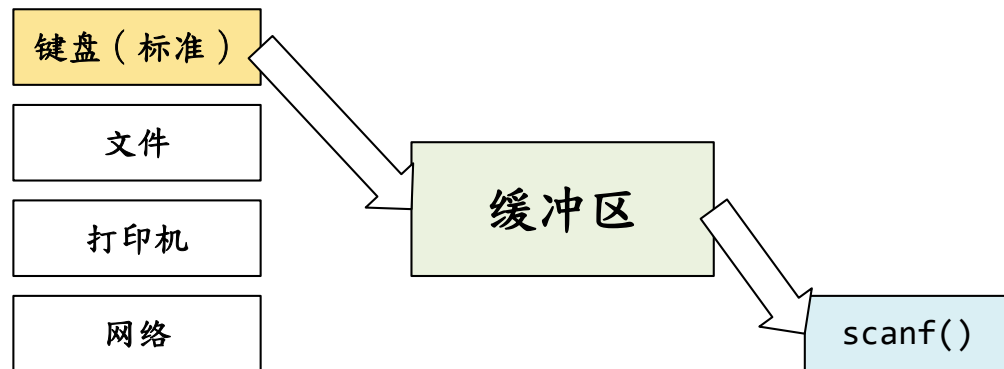
- 标准输入流是键盘

- 输入流可以重新定向到其它设备



输入流的行为

- 缓冲区默认为空



- 当运行到scanf()函数时

- 如果曾经按下 Ctrl+Z ，则返回-1继续

- 如果按下 Ctrl+C ，则程序终止

- 如果缓冲区为空，则暂停程序接受输入

- 如果缓冲区不为空，则依次读入字符并按格式字符串识别

输入流的行为

- 当运行到scanf()函数时
 - 如果缓冲区不为空，则依次读入字符并按格式字符串识别
 - 遇到（部分）格式符合则向对应的赋值（这部分字符取出缓冲区）
 - 如果所有变量已赋值，返回成功赋值的变量个数
 - 如果所有变量未全被赋值，则继续读入直到缓冲区空
 - 遇到格式不符合则停止读取，返回成功赋值的变量个数
 - 不符合格式的部分，仍留在缓冲区待下一次读取
- 程序结束后，缓冲区释放

格式化输入语句的作用

- 从函数第一个参数获取格式字符串
- 从标准输入流中读取字符串
- 从函数后续参数取值，依次按格式说明符解析成值
- 将值赋值到参量指示的内存区域
- 返回正确识别并赋值的参量个数

格式输入函数 scanf()

- 声明语句

```
int scanf( const char *format [, argument]... );
```

- 参数

- format：格式字符串，指定后续参数的输入格式
 - 格式字符串中包含格式说明符（以 % 开头）
- argument, ...：不定长的参数列表

- 返回值

- 正确识别并赋值参数的个数

格式输入函数 scanf()

• 替换原理

```
scanf ( "%f %s", &number, &pies );
```

- 始终以百分号 (%) 开头并从左向右读取。
- 输入字段定义为任何字符到第一个空白字符 (空格、制表符、或者换行符)，或者到无法根据格式规范转换的第一个字符，或直到字段宽度 (如果指定) 为止。
- 如果具有比格式说明更多的参数，则忽略额外的参数。
- 如果没有足够的参数，则结果视编译器而定。

格式输入函数的格式说明

- 格式字符串中包含若干个格式说明

- 格式说明的形式：`%[*][宽度][大小前缀]类型`

— 示例：

`%5lf`

9.8

| 成分 | 可选 | 说明 |
|------|----|-------------------------------------|
| 标志 | 可选 | 星号 (*) 表示接受输入但忽略该项。 |
| 宽度 | 可选 | 十进制整数，指定输入的最大字符数，超过该长度，则不会被赋值给相应参数。 |
| 大小前缀 | 可选 | 用于指示参数的大小。可以是：h、l、ll、I64 和 L。 |
| 类型 | 必选 | 确定相关联的参数被解释为字符、字符串、整数或浮点数。 |

格式输入函数的格式说明

• 主要的类型说明

| 类型符 | 参数 | 输出格式 |
|-----|-----|--|
| d | 整数 | 有符号的十进制整数 |
| f | 浮点 | 单精度浮点数。 |
| s | 字符串 | 显示字符串。 |
| c | 字符 | 字符（包括：单字节字符，宽字符）。 |
| x或X | 整数 | 无符号的十六进制整数；x用“abcdef”；X用“ABCDEF”。 |
| u | 整数 | 无符号的十进制整数。 |
| e或E | 浮点 | 以指数形式输出浮点数，可指定指数部分宽度。 |
| g或G | 浮点 | 当值的指数小于-4或者大于或等于精度参数时使用e格式，否则使用f格式。G使用大写E。 |
| o | 整数 | 无符号的八进制整数 |

格式输入函数的格式说明

- 主要的大小前缀符说明

| 修饰符 | 含义 |
|-----|------------------------------|
| l | 表示(unsigned) long int类型 |
| ll | 表示(unsigned) long long int类型 |
| L | 表示(unsigned) long double类型 |

```
// input.c -- when to use &
```

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

```
int main(void)
```

```
{
```

```
    int age;                // variable
```

```
    float assets;           // variable
```

```
    char pet[30];           // string
```

调用函数改变变量的值时，
应使用&操作符传入其所
在地址，数组名即为地址，
不能加&操作符。

```
    printf("Enter your age, assets, and favorite pet.\n");
```

```
    scanf("%d %f", &age, &assets); // use the & here
```

```
    scanf("%s", pet); // no & for char array
```

```
    printf("%d $%.2f %s\n", age, assets, pet);
```

Enter your age, assets, and favorite pet.

21 32.4↓

Cat↓

21 \$32.40 Cat

Enter your age, assets, and favorite pet.

12 45 Bird↓

12 \$45.00 Bird

输入时，按下回车缓冲区的内容生效，但在输入过程中回车与空格都可以区分不同的变量。


```

/* skiptwo.c -- skips over first two integers of input */
#include <stdio.h>
int main(void)
{
    int n;

    printf("Please enter three integers:\n");
    scanf("%*d %*d %d", &n);
    printf("The last integer was %d\n", n);

    return 0;
}

```

以 (*) 标记的格式化标识符不会赋值给变量。

Please enter three integers:
34 23 52↓
 The last integer was 52

```

/* praise1.c -- uses an assortment of strings */
#include <stdio.h>
#define PRAISE "You are an extraordinary being."
int main(void)
{
    char name[40];

    printf("What's your name? ");
    scanf("%s", name);
    printf("Hello, %s. %s\n", name, PRAISE);

    return 0;
}

```

把字符串定义为宏

开辟固定的空间，
存储字符串

暂停，等待输入至换行符继续运行。
但对数组的赋值到空白字符结束。

What's your name? Wei HUANG
Hello, Wei. You are an extraordinary being.

字符串的格式输入

- 通过 `scanf()` 配合 `%s` 输入一个单词

```
scanf("%s", name);
```

- 程序暂停直到用户输入回车为止

Wei HUANG↓

输入回车之前程序暂停

- 格式输入在空白字符处停止，添加数字0，写入指定内存

- 空格(blank)、制表符(tab)、换行符(newline)

| 偏移量 | 0 | 1 | 2 | 3 | 4 ... 39 |
|-----|----|-----|-----|----|----------|
| 字符 | w | e | i | \0 | 该范围的值未改变 |
| 值 | 87 | 101 | 105 | 0 | 该范围的值未改变 |

- 如果输入要包括空格使用 `gets()` 函数

```
gets(name);
```

字符串的使用

- 字符串输入时用格式标识符%s

— weight前有&符号，而name没有

```
scanf("%s", name);
```

```
scanf("%f", &weight);
```

- 基本类型的变量值是其值，存于内存中
- 数组（字符串）的名称值是其内存数据段的首地址

- 字符串应先声明，再赋值，后使用

— 如果不赋值即使用字符串，则输出乱码直至遇到0字符

Hi! What's your first name?

, what's your weight in pounds?

字符串的格式输出

- 通过 `printf()` 配合 `%s` 输出一个字符串
 - 在遇到数字0结束（不论是否超过字符串开辟的空间）

| 偏移量 | 0 | 1 | 2 | 3 | 4 ... 39 |
|-----|----|-----|-----|----|----------|
| 字符 | W | e | i | \0 | 该范围的值未改变 |
| 值 | 87 | 101 | 105 | 0 | 该范围的值未改变 |

- 打印出 “Wei”
- 字符串操作之前之前应确保其所在空间有一个数字0
 - 违反时，会产生运行错误

Run-Time Check Failure #2 - Stack around the variable 'name' was corrupted.

```

/* praise2.c */
// try the %u or %lu specifiers if your implementation
// does not recognize the %zd specifier
#include <stdio.h>
#include <string.h>      /* provides strlen() prototype */
#define PRAISE "You are an extraordinary being."
int main(void)
{
    char name[40];

    printf("What's your name? ");
    scanf("%s", name);
    printf("Hello, %s. %s\n", name, PRAISE);
    printf("Your name of %zd letters occupies %zd memory
cells.\n",
           strlen(name), sizeof name);

```

用宏代替一个字符串

%zd为输出空间大小专用，
非GCC编译器可能不支持

```
printf("The phrase of praise has %zd letters ",  
strlen(PRAISE));  
printf("and occupies %zd memory cells.\n", sizeof PRAISE);  
  
return 0;  
}
```

strlen是字符串长度函数
sizeof是求取常（变）量空间大小的操作符

What's your name? Wei Huang
Hello, Wei. You are an extraordinary being.
Your name of 3 letters occupies 40 memory cells.
The phrase of praise has 31 letters and occupies 32 memory cells.

计算字符串长度时不包括末尾的数字0，
字符串常量空间包括末尾数字0

内容纲要

| | |
|---|----------------|
| 3 | 格式化输出 printf() |
| 4 | 输出函数的常见错误 |
| 5 | 格式化输入 scanf() |
| 6 | 输入函数的常见错误 |
| 7 | 总结 |

基本数据类型参量未取地址

• 基本数据类型参量未取地址，将引起运行错误

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

```
int main()
```

```
{
```

```
    int a = 0;
```

```
    if (scanf("%d", &a) == 1)
```

```
        printf("%s\n", s);
```

```
    return 0;
```

```
}
```

函数调用时，传递给scanf的实际参数是：“%d”所在的内存首地址（如：3000）和a的内存地址（如：4000）。

函数调用后，scanf获得的参数是3000和4000，但不知道该值是何种计算得到的。它将正确解析的用户输入赋值给地址为4000的变量（即main函数中的a）。

函数调用时，传递给scanf的实际参数是：“%d”所在的内存首地址（如：3000）和a的内存地址（即：0）。

```
if (scanf("%d", a) == 1)
```

函数调用后，scanf获得的参数是3000和0，它将正确解析的用户输入赋值给地址为0的变量（此时出错）。

整型和浮点型说明符之间的误用

- 整型说明符输入浮点数，结果不可靠

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

```
int main()
```

```
{
```

```
    int a = 0;
```

```
    if (scanf("%f", &a) == 1)
```

```
        printf("%d\n", a);
```

```
    return 0;
```

```
}
```

这里使用了浮点型说明符来解析用户输入，却赋值到一个整型变量。

- 浮点型说明符输入整数没有问题

- 使用科学计数法输入，可以被 %f 识别

说明符和参数个数不相等

- 参数个数少于格式说明个数，可能引起运行错误

```
scanf ( "%f %f", &a ) ;
```

- 参数个数多于格式说明个数，多余的参数不被赋值

```
scanf ( "%f %f", &a, &b, &c ) ;
```

说明符有多余的字符

- 在格式字符串输入多余的字符，可能引发错误
 - 用户因不清楚程序所用的格式说明符而无法全部识别

```
scanf ( "%d,%d", &a , &b ) ; scanf ( "%s", s ) ;
```

- 此例中，用户必须在两个整数间输入字符，才能使b赋值

| 按键 | 空格 | 1 | 空格 | 5 | 回车 |
|-------|------------|-------|------------|--------|------|
| 输入缓存 | 0x20 | 0x31 | 0x20 | 0x35 | 0x0a |
| 格式字符串 | | %d | | , | |
| 识别结果 | 空白字符 忽略 | 数字已识别 | 空白字符 忽略 | 非逗号，终止 | |

scanf终止于此，返回正确识别的个数1

下次读取标准输入从这里开始

避免在格式字符串输入多余的字符

- 格式化输入的格式字符串应简单明了

```
scanf ( "%d,%d", &a , &b ) ; scanf ( "%s", s ) ;
```

- 正确输入为：□，整数，□，逗号，□，整数，回车符
- 输入会停留在解析错误的位置

□：可选Tab或空格

| 输入 | 前后函数返回值 | | a值 | b值 | s内容 |
|----------|---------|---|-----|-----|------|
| 1,3↓abc↓ | 2 | 1 | 1 | 3 | abc |
| ,3,1↓ | 0 | 1 | 未赋值 | 未赋值 | ,3,1 |
| 1↓3↓ | 1 | 1 | 1 | 未赋值 | 3 |
| 1,4a↓ | 2 | 1 | 1 | 4 | a |
| 1 5↓ | 1 | 1 | 1 | 未赋值 | 5 |

未接受并处理函数的返回值

- 未接受并处理函数的返回值，可能误导用户
 - 把错误结果当成正确结果提供给用户，是一种误导
 - 如果用户输入错误，程序应尽早停止，避免浪费时间

```
scanf ( "%d %d", &a, &b ) ;
```

```
if (scanf("%d %d", &a, &b) == 2)
```

无法辨别结果是否正确的程序，是低质量的程序。

内容纲要

| | |
|---|----------------|
| 3 | 格式化输出 printf() |
| 4 | 输出函数的常见错误 |
| 5 | 格式化输入 scanf() |
| 6 | 输入函数的常见错误 |
| 7 | 总结 |

谢谢观看



厦門大學
XIAMEN UNIVERSITY



信息学院 黄 焯
(特色化示范性软件学院) 博士, 副教授
School of Informatics Wei Huang