

# 顺序程序设计

程 淼

E-mail: [mew\\_cheng@outlook.com](mailto:mew_cheng@outlook.com)

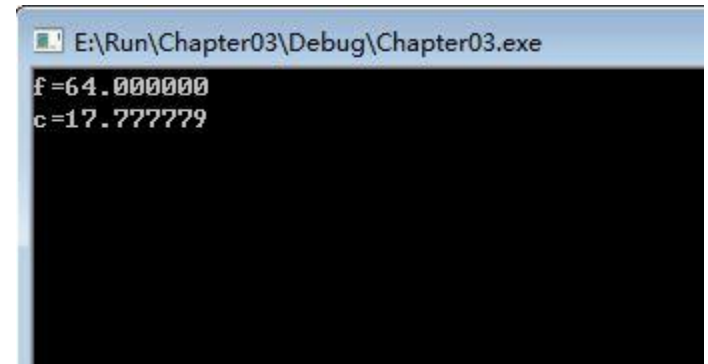
# 顺序程序设计

- 先看一个例子：
- 3.1 有人用温度计测量出用华氏法表示的温度(如 69 F)，今要求把它转换为以摄氏法表示的温度(如 20 C)
  - 关键在于华氏与摄氏温度之间的转换公式：
  - $c = 5/9 (f - 32)$

# 顺序程序设计

- 用C语言实现设计思路

```
#include <stdio.h>
int main( )
{
    float f, c;
    f = 64.0;
    c = (5.0 / 9) * (f - 32);
    printf("f=%f\nc=%f\n", f, c);
    return 0;
}
```



```
E:\Run\Chapter03\Debug\Chapter03.exe
f=64.000000
c=17.777779
```

# 顺序程序设计

- 计算存款利息。有1000元，想存一年。有3种方法可选：
  - (1) 活期，年利率为 $r_1$
  - (2) 一年期定期，年利率为 $r_2$
  - (3) 存两次半年定期，年利率为 $r_3$
- 请分别计算出一年后按3种方法所得到的本息和

# 顺序程序设计

- 思路：不同存款时间条件下的不同存款本息计算方式

- 活期存款一年后的本息和为

$$p1=p0(1+r1)$$

- 一年期定期存款，一年后本息和为

$$p2=p0(1+r2)$$

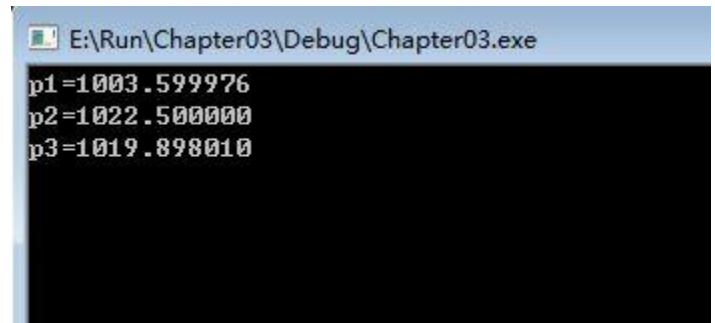
- 两次半年定期存款，一年后本息和为

$$p3=p0\left(1+\frac{r3}{2}\right)\left(1+\frac{r3}{2}\right)$$

# 顺序程序设计

- 用C语言实现设计思路

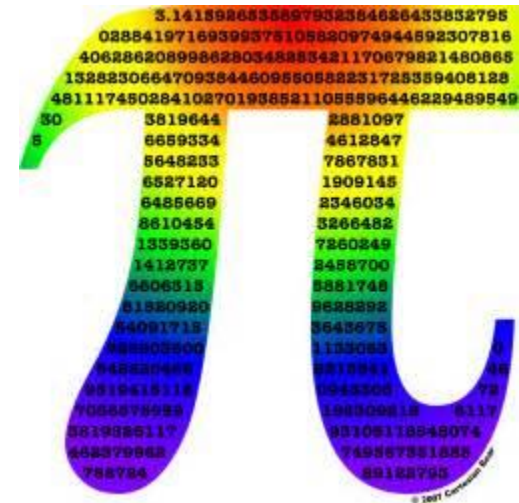
```
#include <stdio.h>
int main( )
{
    float p0 = 1000, r1 = 0.0036, r2 = 0.0225, r3 = 0.0198, p1, p2, p3;
    p1 = p0*(1+r1);
    p2 = p0*(1+r2);
    p3 = p0*(1+r3/2)*(1+r3/2);
    printf("p1=%f\np2=%f\np3=%f\n", p1, p2, p3);
    return 0;
}
```



```
E:\Run\Chapter03\Debug\Chapter03.exe
p1=1003.599976
p2=1022.500000
p3=1019.898010
```

# 数据的表现形式

- 常量：在程序运行过程中，其值不能被改变的量
  - 整型常量
  - 实型常量
    - 十进制小数形式
    - 指数形式



# 数据的表现形式

- 字符常量
  - 普通字符：'a','Z','3','?', '#'
  - 转义字符：以字符\开头的字符序列



# 数据的表现形式

- 字符串常量：用双撇号把若干个字符括起来，字符串常量是双撇号中的全部字符(但不包括双撇号本身)
- 符号常量
  - 用`#define`指令，指定用一个符号名称代表一个常量
    - `#define PI 3.1415`

# 变量

- **变量**代表一个有名字的、具有特定属性的一个存储单元
  - 必须先定义，后使用

无论怎么变，我都叫“孙悟空”



# 常变量

- 常变量具有变量的基本属性，有类型，占存储单元，只是不允许改变其值
  - `const int a = 3;`
- 常变量是有名字的不变量，而常量是没有名字的不变量

# 常变量

- 符号常量：
  - 预编译指令，只是用符号常量代表一个字符串
  - 在预编译时进行字符替换，预编译后，符号常量就不存在了
  - 不分配存储单元
- 常变量：
  - 要占用存储单元

# 标识符

- 在计算机高级语言中，用来对变量、符号变量名、函数、数组、类型等命名的有效字符序列
- 一个对象的名字
- 比如：变量名，符号常量名，函数名

来者  
何人？

# 数据类型

- 所谓**类型**，就是对数据分配存储单元的安排，包括**存储单元的长度**(占多少字节)以及数据的**存储形式**

# 整型数据

- 编译系统分配给int型数据2个字节或4个字节
  - Turbo C 2.0为每一个整型数据分配2个字节(16个二进制位)
  - Visual C++为每一个整型数据分配4个字节(32位)

# 整型数据

- 用整型数据的补码(complement)形式存放

## 5的补码

0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

## 5的原码

0		0		0		0		0		0		0		0		0		0		0		0		1		0		1
---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---

## 按位取反

1   1   1   1   1   1   1   1	1   1   1   1   1   0   1   0
-------------------------------	-------------------------------

再加1  
(-5的补码)

1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



# 整型数据

- 短整型： `short int` 或 `short`
  - VC 的编译系统分配给 `int` 数据4个字节，短整型2个字节
  - 短整型变量的取值范围：  $-32768 \sim 32768$
- 长整型： `long int`
  - 一般分配8个字节
  - 长整型变量的取值范围：  $-2147483648 \sim 2147483647$

# 双长整型

- 类型名为long int或long long，一般分配8个字节
- C标准没有具体规定各种类型数据所占用存储单元的长度
  - $\text{sizeof}(\text{short}) \leq \text{sizeof}(\text{int}) \leq \text{sizeof}(\text{long}) \leq \text{sizeof}(\text{long long})$

# 整型变量

- 在实际应用中，对于只有正值的变量，可以将其定义为“无符号”类型，在类型符号前面加上修饰符unsigned，表示该变量是“无符号整数”类型；sign表示“有符号类型”

# 整型变量

- 注意：
  - 只有整型(包括字符型)数据可以加signed或unsigned修饰符，实型数据不能加
  - 对无符号整型数据用“%u”格式输出。%u表示用无符号十进制数的格式输出

```
unsigned short price = 50;  
printf("%u\n", price);
```

```
unsigned short price = -1;    X  
printf("%d\n", price);
```

得到的结果为65535

# 字符型数据

- 各种ASCII字符集包括了127个字符，**最多用7个二进制位就可以表示**，其中包括
  - 字母
  - 数字
  - 专门字符：29个
    - ! ” # & ‘ ( ) \* + , - . / : ; < =
  - 空格符
  - 不能显示的字符

# 字符变量

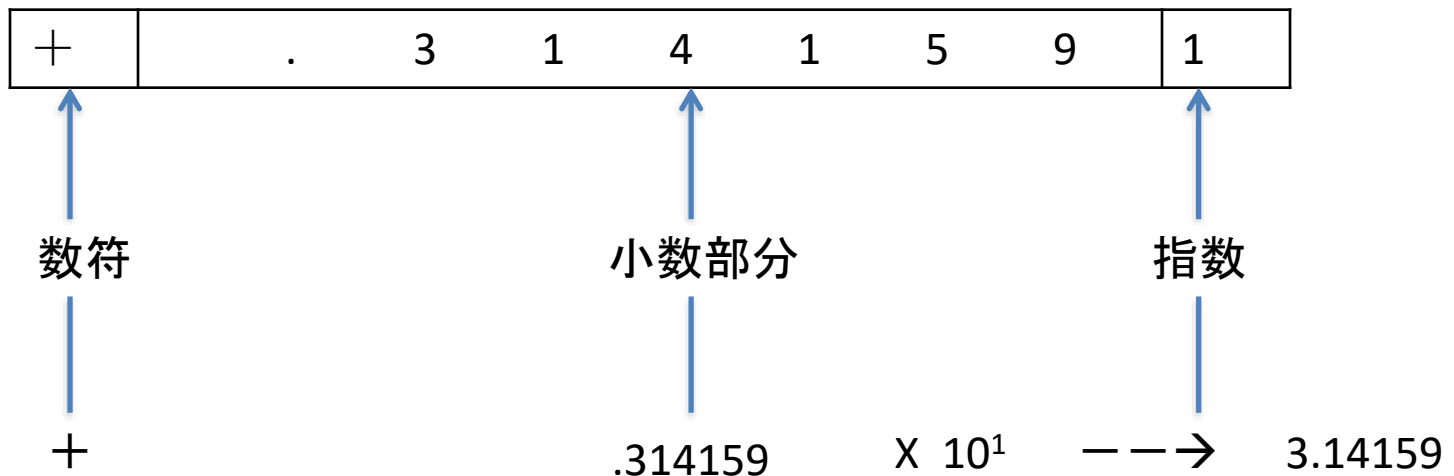
- 字符变量是用类型符char定义字符变量
  - char c = ‘?’
- signed char (有符号字符型):  $-2^7 \sim 2^7-1$
- unsigned char(无符号字符型):  $0 \sim (2^8-1)$

# 浮点型数据

- 浮点型数据表示具有小数点的实数
- **规范化的指数形式**：把小数部分中小数点前的数字为0、小数点后第1位数字不为0的表示形式
  - **唯一性**，如0.314159e001

# 浮点型数据

- float型：单精度浮点型
  - 每个float型变量分配4个字节
  - 6位有效数字，数值范围为 $-3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$
  - 数值以规范化的二进制数指数形式存放在存储单元中





# 双精度浮点型

- double型(双精度浮点型)
  - 用8个字节存储一个double型数据
  - 15位有效数字数值，数值范围为 $-1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$

# 长双精度型

- Turbo C对long double型分配16个字节，而VC对long double型和double型一样处理，分配8个字节

# 确定常量的类型

- 从常量的表示形式可以判定其类型
  - 整型常量
  - 浮点型常量
- 可以在常量的末尾加专用字符，如l和f

# 运算符和表达式

- 基本的算术运算符

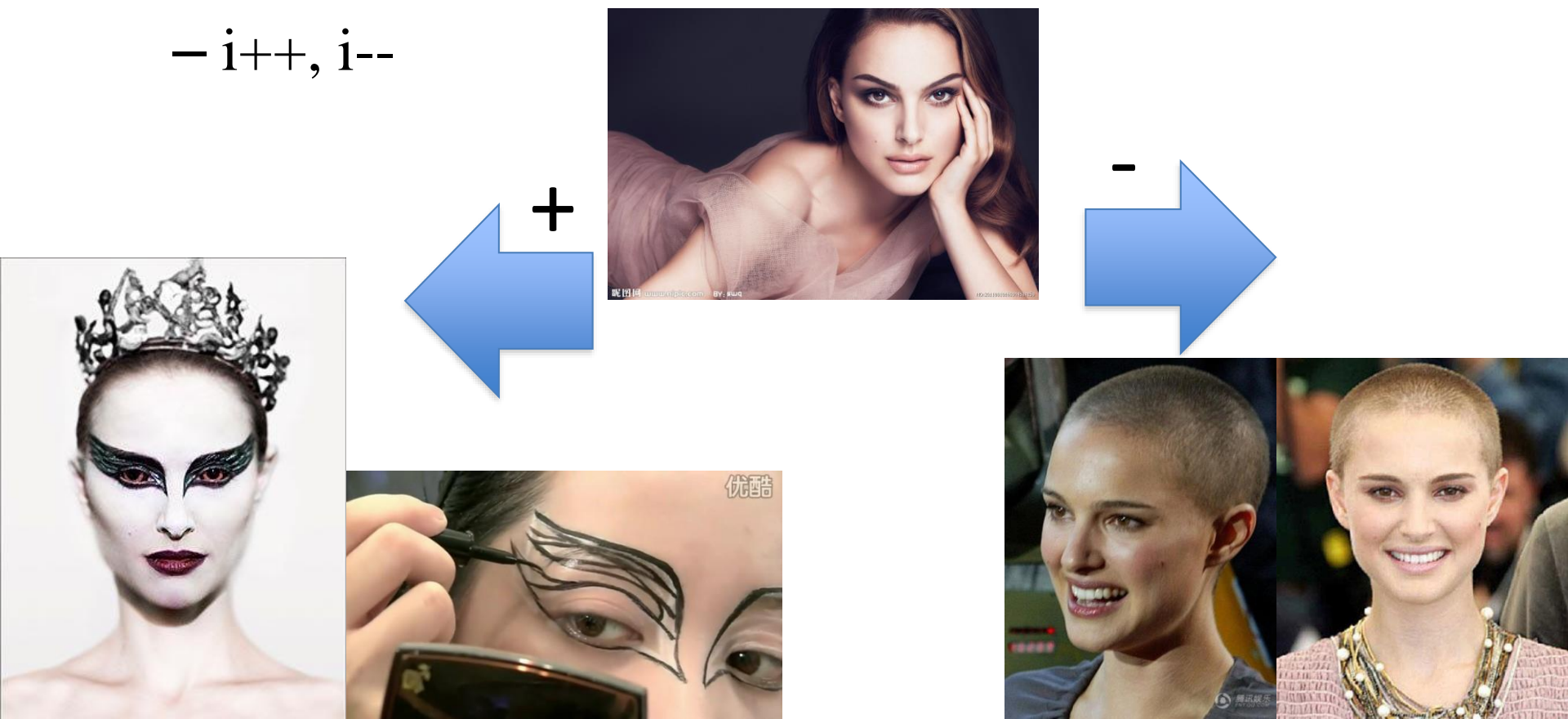
$+$   $-$   $*$   $/$   $\%$   $++$   $--$

# 自增、自减

- 使变量的值加1或减1，例如

$--i$ ,  $++i$

$i--$ ,  $i++$



# 优先级与结合性

- 在表达式求值时，先按运算符的优先级别顺序执行

# 不同类型数据间的混合运算

- $+$ 、 $-$ 、 $*$ 、 $/$  运算中的两个数中有一个数为float或double型
  - 将所有float型数据转换为double型，然后进行运算
- int、float或double型数据的运算
  - 先将int和float型数据转换为double型，然后进行运算
- 字符(char)型数据与整型数据进行运算
  - 其实是把字符的ASCII代码进行运算

# 不同类型数据间的混合运算

- 假设已指定i为整型变量，值为3，f为float型变量，值为2.5，d为double型变量，值为7.5，a的值是97

$$-10 + 'a' + i * f - d / 3$$



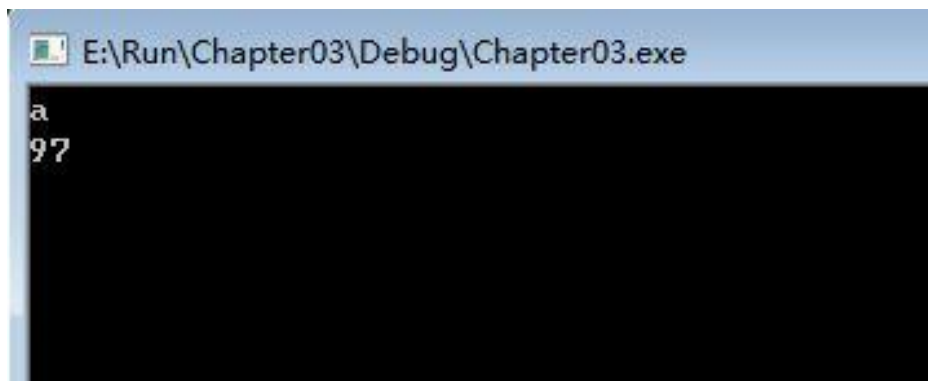
# 一个例子

- 给定一个大写字母，要求用小写字母输出

```
#include <stdio.h>

int main()
{
    char c1, c2;
    c1 = 'A';
    c2 = c1 + 32;
    printf("%c\n", c2);
    printf("%d\n", c2);

    getchar();
    return 0;
}
```



# 强制类型转换

- 利用强制类型转换运算符将一个表达式转换成所需类型
  - (double)a
  - (int) (x + y)
  - (float) (5%3)



# 强制类型转换

- (类型名)(表达式)
  - (int)x+y
  - a = (int)x
- 两种类型转换
  - 系统自动转换
  - 强制类型转换

# C运算符

- 算术运算符
- 关系运算符
- 逻辑运算符
- 位运算符
- 赋值运算符
- 条件运算符
- 逗号运算符
- 指针运算符
- 求字节数运算符
- 强制类型转换运算符
- 成员运算符
- 下标运算符

# C语句

- C语句的作用和分类
  - 控制语句: goto X
  - 函数调用语句
  - 表达式语句
  - 空语句
  - 复合语句

# 赋值语句

- 作用：对变量赋值

# 赋值语句

- 先看一个例子：给出三角形的三边长，求三角形面积

$$- area = \sqrt{s(s-a)(s-b)(s-c)}$$

$$- s = (a + b + c)/2$$

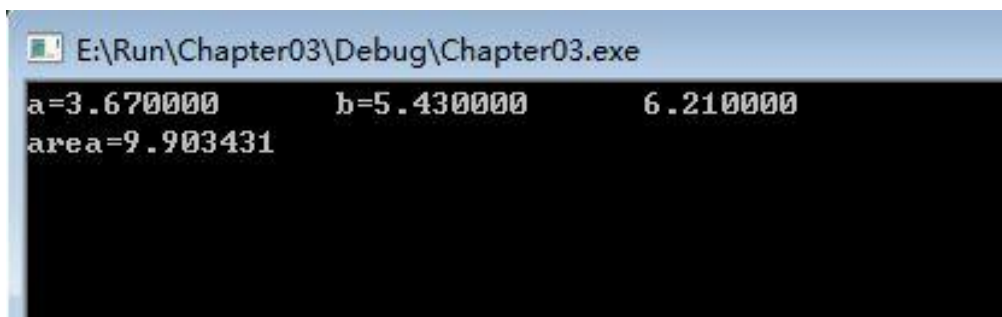
# 赋值语句

- 运行结果

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

int main()
{
    double a, b, c, s, area;
    a = 3.67;
    b = 5.43;
    c = 6.21;
    s = (a+b+c)/2;
    area = sqrt(s*(s-a)*(s-b)*(s-c));
    printf("a=%f\tb=%f\t%f\n", a, b, c);
    printf("area=%f\n", area);

    getchar();
    return 0;
}
```



```
E:\Run\Chapter03\Debug\Chapter03.exe
a=3.670000      b=5.430000      6.210000
area=9.903431
```



# 赋值语句

- 赋值运算符
  - 赋值符号=就是赋值运算符
- 复合的赋值运算符
  - 在赋值符=之前加上其他运算符，可以构成复合的运算符
  - $a+=3$
  - $x*=y+8$
  - $x\%=3$

# 赋值语句

- 赋值表达式：由赋值运算符将一个变量和一个表达式连接起来的式子称为“赋值表达式”
  - 变量 赋值运算符 表达式

# 赋值语句

- $b = a;$
- $c = b;$
- $a = (b = 5);$
- $a = b = c = 5;$
- $a = 5 + (c = 6);$
- $a = (b = 4) + (c = 6);$
- $a = (b = 10) / (c = 2);$
- $a = (b = 3 * 4);$
- $(a = b) = 3 * 4;$
- $a = (a = b) = 3 * 4;$
- $a+ = a -= a * a;$

# 赋值语句

- 赋值过程中的类型转换

- 如果赋值运算符两侧的类型一致
- 如果赋值运算符两侧的类型不一致
  - 将浮点型数据(包括单、双精度)赋给整型变量
  - 将整型数据赋给单、双精度变量
  - 将一个double型数据赋给float变量
  - 字符型数据赋给整型变量
  - 将一个占字节多的整型数据赋给一个占字节少的整型变量或字符变量

# 赋值语句

- 赋值表达式可以出现在其他表达式中
  - `if ((a = b) > 0)    max = a;`
  - `if ((a = b;) > 0)    max = a;`    ✗

# 赋值语句

- 变量赋初值
  - `int a = 3;`
  - `float f = 3.56;`
  - `char c = 'a';`

# 数据的输入和输出

- 输入设备和输出设备



# 数据的输入输出

- 输入输出是以计算机主机为主体而言的
- C语言本身不提供输入输出语句
  - 输入和输出操作是由C标准函数库中的函数来实现的，scanf函数和printf函数





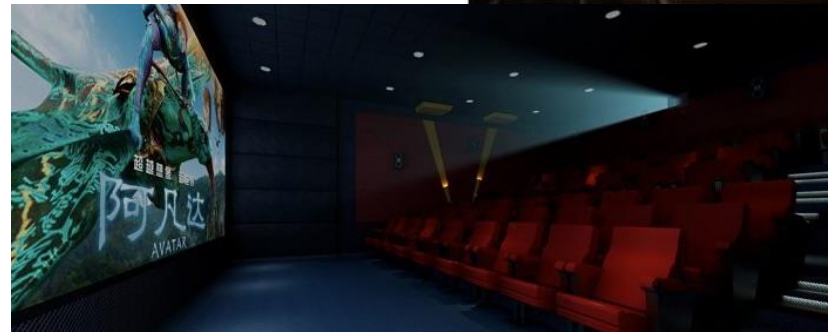
# 数据的输入输出

- 利用printf函数进行数据输出的程序
- 求 $ax^2+bx+c=0$ 方程的根

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$$p = \frac{-b}{2a}, q = \frac{\sqrt{b^2 - 4ac}}{2a}$$

$$x_1 = p + q, x_2 = p - q$$



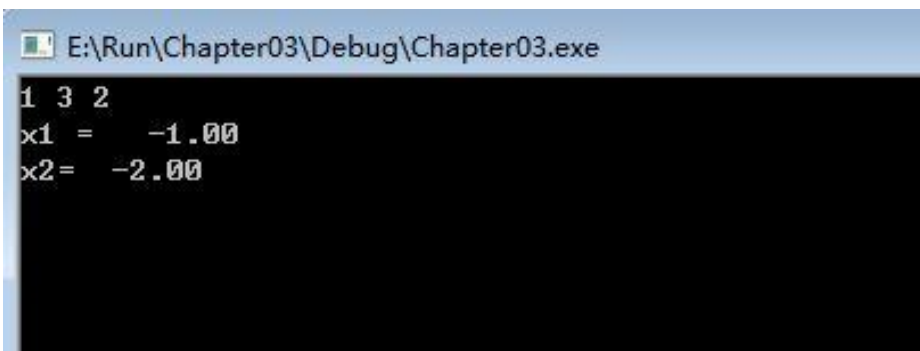
# 数据的输入输出

- 运行结果

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

int main()
{
    double a, b, c, disc, x1, x2, p, q;
    scanf("%lf%lf%lf", &a, &b, &c);
    disc = b*b - 4*a*c;
    p = -b/(2.0*a);
    q = sqrt(disc)/(2.0*a);
    x1 = p + q;
    x2 = p - q;
    printf("x1 = %7.2f\nx2=%7.2f\n", x1, x2);

    //getchar();
    return 0;
}
```



```
E:\Run\Chapter03\Debug\Chapter03.exe
1 3 2
x1 = -1.00
x2 = -2.00
```

# 数据的输入和输出

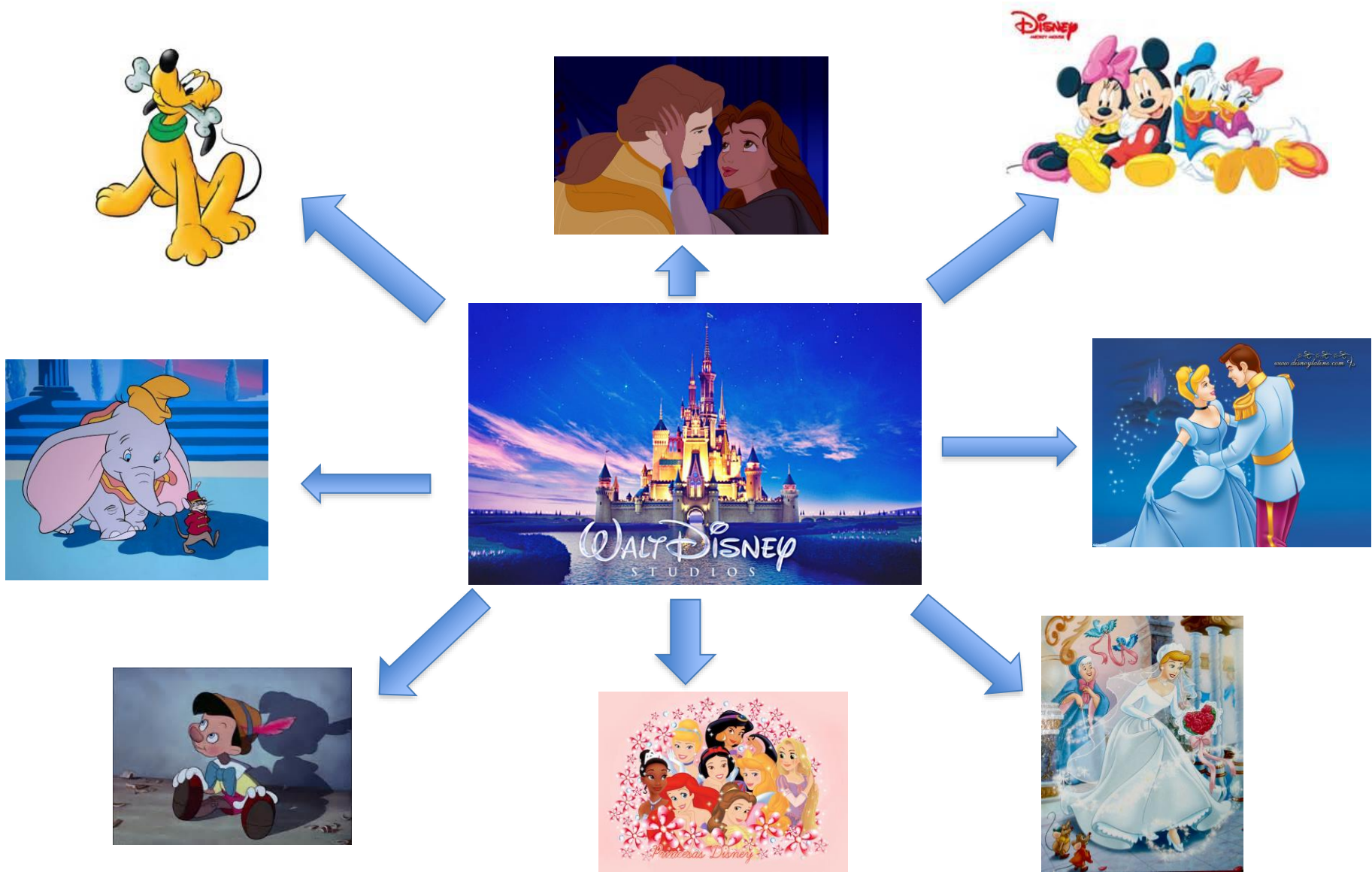
- 在使用系统库函数时，要在程序文件的开头用预定义指令#include把有关头文件放在本程序中
  - 如：#include <stdio.h> 或 #include “stdio.h”

It is 'stdio', not  
'studio' !



# 数据的输入和输出

- #include “disney.h”



# 数据的输入和输出

- 用printf函数输出数据
  - printf(格式控制, 输出表列)
  - 例如:
    - printf(“%d, %c\n”, i, c);
  - “格式控制”
    - 格式声明, 如%d、%f
    - 普通字符

# 数据的输入和输出

- “输出表列”是程序需要输出的一些数据，可以是常量、变量或表达式
- `printf(“a = %d b = %d”, a, b);`

# 数据的输入和输出

- 格式字符
  - d格式符：用来输出一个有符号的十进制整数
  - c格式符：用来输出一个字符

```
char ch = 'a';  
printf("%c", ch);  
输出    a
```

```
short a = 121;  
printf("%c", a);  
int a = 377;  
printf("%c", a);  
都是输出  y
```

0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

# 数据的输入和输出

- s格式符：用来输出一个字符串
  - `printf(“%s”, “CHINA”);`
- f格式符：用来输出实数(包括单、双精度、长双精度)，以小数形式输出
  - 基本型，用`%f`
  - 指定数据宽度和小数位数，用`%m.nf`
  - 输出的数据向左对齐，用`%-m.nf`
    - `printf(“%-25.15f, %25.15f\n”, a, a);`



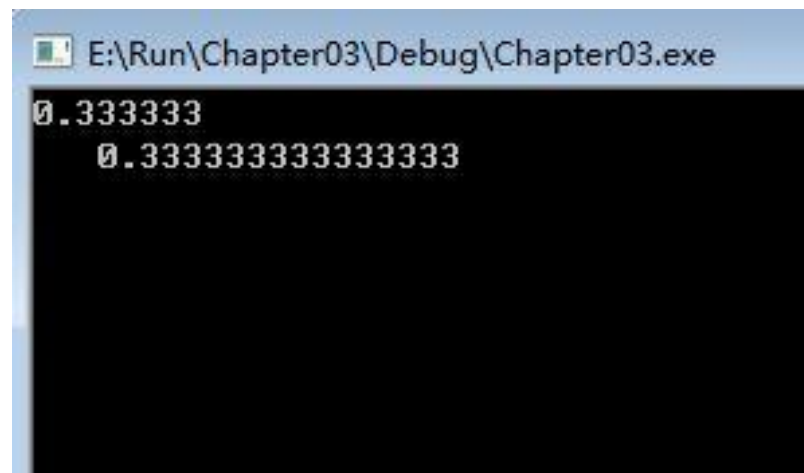
# f格式符

- 用%f输出实数

```
#include <stdio.h>

int main()
{
    double a = 1;
    printf("%f\n", a/3);
    printf("%20.15f\n", a/3);

    getchar();
    return 0;
}
```



E:\Run\Chapter03\Debug\Chapter03.exe

0.333333

0.33333333333333333

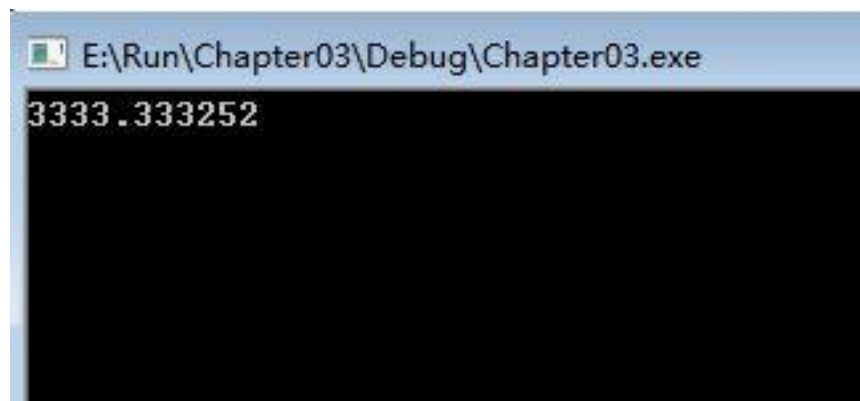
# f格式符

- float型数据的有效位数

```
#include <stdio.h>

int main()
{
    float a;
    a = 10000 / 3.0;
    printf("%f\n", a);

    return 0;
}
```



E:\Run\Chapter03\Debug\Chapter03.exe  
3333.333252

# 数据的输入和输出

- e格式符：指定以指数形式输出实数
  - `printf(“%e”, 123.456);`
- 也可以用 “%**m**.**n**e” 形式的格式声明
  - `printf(“%13.2e”, 123.456);`

# 数据的输入和输出

- C语言还提供了几种输出格式符
  - i格式符：作用与d格式符相同
    - 一般习惯用%d而少用%i
  - o格式符：以八进制整数形式输出

```
int a = -1;  
printf("%d\t%o\n", a, a);
```

- x格式符：以十六进制数形式输出整数

```
int a = -1;  
printf("%d\t%o\t%x\n", a, a, a);
```

# 数据的输入和输出

- u格式符：用来输出无符号型数据，以十进制整数形式输出
- g格式符：用来输出浮点数，系统自动选f格式或e格式输出

```
double a = 12345678954321  
printf("%f\t%e\t%g\n", a, a, a);
```

# 数据的输入和输出

- % 附加字符 格式字符

- 附加字符，又称为“修饰字符”，如 l, m, n, -等

字符	说明
l	用于长整型整数，可加在格式符d、o、x、u前面
m（代表一个正整数）	数据最小宽度
n（代表一个正整数）	对实数，表示输出n位小数；对字符串，表示截取的字符个数
-	输出的数字或字符在域内向左靠

# 数据的输入和输出

- 用scanf函数输入数据
  - 一般形式：scanf(格式控制，地址表列)
  - 格式声明以%开始，以一个格式字符结束，中间可以插入附加的字符
    - scanf(“a=%f, b=%f, c=%f”, &a, &b, &c);

# 数据的输入和输出

- scanf中的格式控制
  - 格式字符: d, i, u, o, x, X, c, s, f, e, E, g, G
  - 格式附加字符

字符	说明
l	用于输入长整型数据(可用%d, %lo, %lx, %lu)以及double型数据(%lf或%le)
h	用于输入短整型数据(可用%hd, %ho, %hx)
域宽	指定输入数据所占宽度(列数), 域宽应为正整数
*	表示本输入项在读入后不赋给相应的变量



# 数据的输入和输出

- 使用scanf函数的注意问题
  - scanf函数中的“格式控制”后面是变量地址，而不是变量名
- 如果在“格式控制字符串”中除了格式声明以外还有其他字符，则在输入数据时在对应的位置上应输入与这些字符相同的字符

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

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

```
a=1, b=3, c=2
```

# 数据的输入和输出

- 在用 “%c” 格式声明输入字符时，空格字符和 “转义字符” 中的字符都作为有效字符输入
  - 例如，scanf(“%c%c%c”, &c1, &c2, &c3);

abc	✓
a b c	✗

# 数据的输入和输出

- 在输入数值数据时，如输入空格、回车、Tab键或遇非法字符(不属于数值的字符)

```
scanf(“%d%c%f”, &a, &b, &c);
```

```
1234a 123o.26
```

# 数据的输入和输出

- 除printf和scanf函数外，C函数库还提供
  - putchar函数输出一个字符
  - getchar函数输入一个字符

# 数据的输入和输出

- 先后输出BOY三个字符

```
#include <stdio.h>

int main()
{
    char a = 'B', b = 'O', c = 'Y';
    putchar(a);
    putchar(b);
    putchar(c);
    putchar('\n');

    getchar();
    return 0;
}
```

