

第3章 简单C程序设计基础

3.1 引例-简单程序设计

3.2 数据表示及其运算

3.3 C语句

3.4 数据的输入输出

第3章 简单C程序设计基础

3.1 引例-简单程序设计

3.2 数据表示及其运算


3.3 C语句

3.4 数据的输入输出

3.1 简单程序设计示例

例3.1 有人用温度计测量出用华氏法表示的温度(如 f ，今要求把它转换为以摄氏法表示的温度(如 c))。

➤ 解题思路：找到二者间的转换公式

$$c = \frac{5}{9}(f - 32)$$


物理知识
数学公式

f 代表华氏温度， c 代表摄氏温度

3.1 简单程序设计示例

例3.1 有人用温度计测量出用华氏法表示的温度(如 f ，今要求把它转换为以摄氏法表示的温度(如 c))。

➤ 算法:

获取 f 的值
$c = \frac{5}{9}(f - 32)$
输出 c 的值

N-S图

3.1 简单程序设计示例

例3.2 求两个正整数的最大公约数。

➤ 解题思路1：所有公因子中的最大值

➤ 解题思路2：欧几里德算法

原理：两个正整数的最大公约数等于较小的那个数和两数相除余数的最大公约数。亦即： $\text{gcd}(m, n) = \text{gcd}(n, m \% n)$

(这里不妨假设 $m > n$)

➤ **S1**：求余数：用 n 除 m ，令 r 为余数（这里 $0 \leq r < n$ ）；

➤ **S2**：余数为0？如果 $r=0$ ，算法终止， n 就是答案；

➤ **S3**：减少，置 $m \leftarrow n, n \leftarrow r$ ，然后转回**S1**继续。

3.1 简单程序设计示例

```
#include <stdio.h> //ch3-1.c
```

```
int main ( )
```

```
{
```

```
    float f,c;  定义f和c为单精度浮点型变量
```

```
    f=64.0;    指定f的值
```

```
    c=(5.0/9)*(f-32);  计算c的值
```

```
    printf("f=%f\n c=%f\n",f,c);
```

```
    return 0;    输出f和c的值
```

```
}
```

```
f=64.000000
c=17.777778
```

3.1 简单程序设计示例

```
#include <stdio.h> //ch3-2.c(Euclid Alg.)
int main ( )
{
    int m, n;          //定义m和n为整型变量
    scanf("%d%d", &m, &n); //输入m和n的值
    int r = m%n;        //s1:r用来存储m除以n的余数
    while(r){           //s2:余数为0结束，否则去做S3
        m = n; n = r; r=m%n; //s3:减少(迭代)
    }
    printf("m和n的最大公约数是: %d\n", n);
    return 0;
}
```

3.2 数据表示及其运算

3.2.1 常量和变量

3.2.2 数据类型

3.2.3 整型数据

3.2.4 字符型数据

3.2.5 浮点型数据

3.2.6 怎样确定常量的类型

3.2.7 运算符和表达式

3.2.1 常量和变量

1. 常量：在程序运行过程中，其值不能被改变的量

➤ 整型常量：以数字开头，但没有小数点或指数部分。可能包含一个前缀用于指定其进制，一个后缀用于指定其类型

◆ 十进制整数：以非0数字开头，由0~9数字序列组成，如：1204

◆ 八进制整数：以0开头，由0~7数字序列组成，如：015

◆ 十六进制整数：以前缀0x开头，由0~9,A~F,a~f数字或字符序列组成，如：0x3a,0X3A

◆ 后缀：u,U无符号整数 l,L 长整型 ll,LL 双长整型

➤ 实型常量/浮点型常量

◆ 十进制浮点数形式：如0.34, -56.79f, 1.1l, 1., .1

◆ 十进制指数形式：如12.34e3 (代表 12.34×10^3)

◆ 十六进制浮点数：0x1.3p3, 0x.8p-2, 0x8.p-2
(代表 $0x1.3 \times 2^3$, $0x0.8 \times 2^{-2}$, $0x8.0 \times 2^{-2}$)

3.2.1 常量和变量

1. 常量

- 字符常量：是整数的一种，由单引号括起来的一个或多个字符构成的序列。如：'X'
 - ◆ 源字符集中的任何字符：字母、数字、符号等，除'和\外
 - ◆ 转义字符：由\开头的字符序列构成的字符
 - ◆ 一般转义字符序列：\0 \' \" \? \\ \a \b \f \n \r \t \v
 - ◆ 八进制转义字符序列：\和1~3个八进制数字组成
 - ◆ 十六进制转义字符序列：\x和1~2个十六进制数字组成
- 字符串常量：由双引号括起来的零个或多个字符构成的序列，如"boy", "", "03A!",
- 枚举常量：声明为枚举元素的标识符，属于整型类型
- 宏（无类型符号常量）：#define **PI** 3.1416

3.2.1 常量和变量

2. 变量：在程序运行期间，变量的值是可以改变的

- 变量必须先定义，后使用
- 定义变量时指定该变量的名字和类型
- 变量名和变量值是两个不同的概念
- 变量名实际上是以一个名字代表的一个存储地址
- 从变量中取值，实际上是通过变量名找到相应的内存地址，从该存储单元中读取数据

3.2.1 常量和变量

3.只读变量: **const int a=3; //常变量**

4.标识符: 一个对象的名字 **大小写字母是不同的字符**

- C语言规定标识符只能由**字母、数字和下划线**3种字符组成, 且**第一个字符必须为字母或下划线**
- 合法的标识符: 如**sum, average, _total, Class, day, BASIC, li_ling**
- 不合法的标识符: **M.D.John, ¥123, #33, 3D64, a>b**

3.2.2 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
- 不同的类型分配不同的长度和存储形式
- 变量在定义时都要为其指定类型
- 常量也是区分类型的

3.2.2 数据类型

C语言允许使用的数据类型:

➤ 基本类型

◆ 整型类型

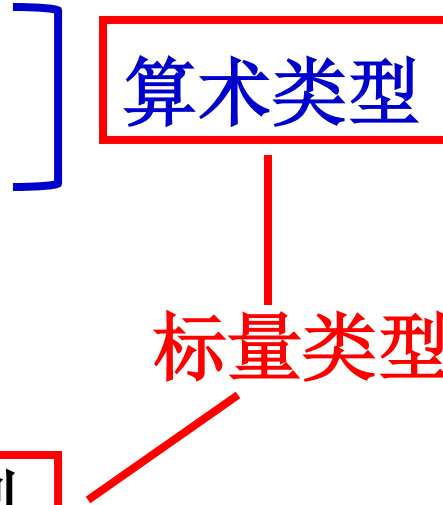
- 基本整型(**int 4**)
- 短整型(**short 2**)
- 长整型(**long 4**)
- 双长整型(**long long 8**)
- 字符型(**char 1**)
- 布尔型(**bool 1**)
 - ★ **<stdbool.h>**

◆ 浮点类型

- 单精度浮点型(**float 4**)
- 双精度浮点型(**double 8**)
- 长精度浮点型
 - (long double 16 GCC)
- 复数浮点型 **complex.h**
 - **<complex.h>**
 - **float complex 8**
 - **double complex 16**
 - **long double complex 32**

3.2.2 数据类型

C语言允许使用的数据类型：

- 基本类型
 - 枚举类型
 - 空类型
 - 派生类型
 - ◆ 指针类型
 - ◆ 数组类型
 - ◆ 结构体类型
 - ◆ 共用体类型
 - ◆ 函数类型
- 
- 算术类型
- 标量类型

3.2.3 整型数据

1. 整型数据的分类

➤ 最基本的整型类型

- ◆ 基本整型(**int**): 占4个字节 $-2^{31} \sim 2^{31}-1$
- ◆ 短整型(**short int**): 占2个字节 $-2^{15} \sim 2^{15}-1$
- ◆ 长整型(**long int**): 占4个字节
- ◆ 双长整型(**long long int**): 占8个字节

2. 整型变量的符号属性

- ◆ 整型变量的值的范围包括从负数到正数，默认有符号
- ◆ 在仅涉及到正整数的时候，可以使用**unsigned**关键字将变量声明为“无符号”类型

3.2.3 整型数据

扩展的整型类型:

- 有符号基本整型 **[signed] int** **$-2^{31} \sim 2^{31}-1$**
- 无符号基本整型 **unsigned int** **$0 \sim 2^{32}-1$**
- 有符号短整型 **[signed] short [int]**
- 无符号短整型 **unsigned short [int]**
- 有符号长整型 **[signed] long [int]**
- 无符号长整型 **unsigned long [int]**
- 有符号双长整型 **[signed] long long [int]**
- 无符号双长整型 **unsigned long long [int]**

整型数据的存储: 以二进制补码形式

3.2.4 字符型数据

- C语言把字符型数据作为整型类型的一种
- 默认为占**1**个字节的无符号整型
- 字符是按其**ASCII**码(整数)形式存储的

ASCII字符集

- ◆字母：**A ~Z, a ~z** 数字：**0~9**
- ◆图形字符：**29个：! " # & ' () *等**
- ◆空格符：空格、水平制表符、换行等
- ◆控制字符：**ASCII码值0~31和127**，其中空
(**null**)字符(以'\0'表示)、警告(以'\a'表示)、退
格(以'\b'表示)、回车(以'\r'表示)等

3.2.4 字符型数据

➤ 字符'1'和整数1是不同的概念:

◆ 字符'1'只是代表一个形状为'1'的符号，在需要时按原样输出，在内存中以**ASCII**码形式存储，占**1**个字节

0 0 1 1 0 0 0 1

◆ 整数1是以整数存储方式(二进制补码方式)存储的，占**4**个字节

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

3.2.4 字符型数据

字符变量

➤ 用类型符**char**定义字符变量

◆ **char c = '?';**

系统把“?”的**ASCII**代码**63**赋给变量**c**

◆ **printf("%d %c\n", c, c);**

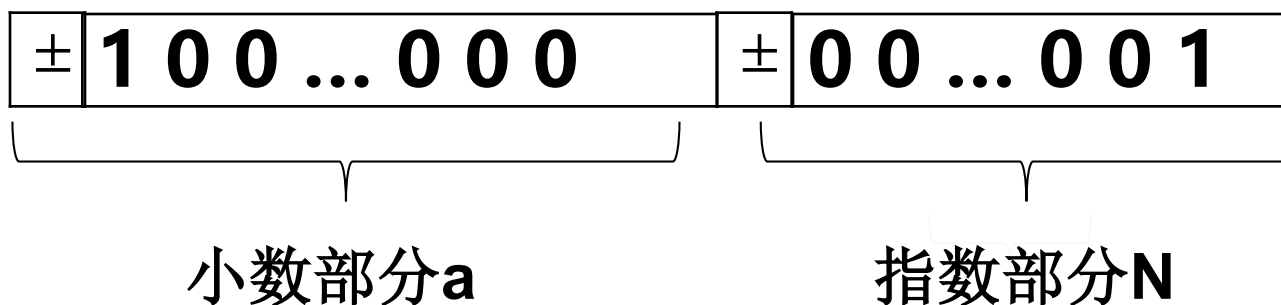
◆ 输出结果是:

63 ?

3.2.5 浮点型数据

浮点型数据：用来表示具有小数点的实数，C语言中，浮点数是以规范化的指数形式存储的，如十进制浮点数形式为： $\mathbf{a} \times 10^{\mathbf{N}}$ ，其中 $|\mathbf{a}|$ 为0~1之间的小数。

C语言中，编译器存储浮点数时，将存储单元分为两部分，一部分用来存储尾数 \mathbf{a} (小数)，一部分用来存储指数 \mathbf{N} (整数)，形式如下：



3.2.5 浮点型数据

➤ **float**型(单精度浮点型)

- ◆ **GCC**编译系统为**float**型变量分配**4**个字节

- ◆ **6**位有效数字

➤ **double**型(双精度浮点型)

- ◆ 编译系统为**double**型变量分配**8**个字节

- ◆ **15**位有效数字

➤ **long double**(长双精度)型

- ◆ **GCC-16Byte, MSVC-8Byte**

- ◆ **19**位有效数字, **15**位有效数字

3.2.6 怎样确定常量的类型

➤ 字符常量

- ◆ 由单撇号括起来的单个字符或转义字符

➤ 整型常量：不带小数点的数值

- ◆ 系统根据数值的大小确定是**int/long**型还是**long long**型等

➤ 浮点型常量

- ◆ 凡以小数形式或指数形式出现的实数
- ◆ **C**编译系统把浮点型常量都按双精度处理
- ◆ 分配**8**个字节

3.2.7 运算符和表达式

1. 基本的算术运算符:

+ : 正号运算符(单目运算符)

- : 负号运算符(单目运算符)

***** : 乘法运算符

/ : 除法运算符

% : 求余运算符

+ : 加法运算符

- : 减法运算符

3.2.7 运算符和表达式

说明

- 两个整数相除的结果为整数
 - ◆ 如**5/3**的结果值为1，舍去小数部分
 - ◆ 如果除数或被除数中有一个为负值，舍入方向不固定。例如，**-5/3**，有的系统中得到的结果为**-1**，在有的系统中则得到结果为**-2**
 - ◆ 一般编译器都是采取“向零取整”的方法
如**5/3=1**，**-5/3=-1**，取整后向零靠拢
- **%** 运算符要求参加运算的运算对象(即操作数)为整数，结果也是整数。如**8%3**，结果为**2**
- 2. 自增、自减运算符：
 - 作用是使变量的值1或减1
 - ◆ **++i**，**--i**：在使用**i**之前，先使**i**的值加（减）**1**
 - ◆ **i++**，**i--**：在使用**i**之后，使**i**的值加（减）**1**

3.2.7 运算符和表达式

3. 算术表达式和运算符的优先级与结合性：

- 用算术运算符和括号将运算对象（也称操作数）连接起来的、符合C语法规则的式子，称为C算术表达式
- 运算对象包括常量、变量、函数等
- C语言规定了运算符的优先级和结合性

4. 不同类型数据间的混合运算：

- (1) **+**、**-**、*****、**/** 运算的两个数中有一个数为**float**或**double**型，结果是**double**型。系统将**float**型数据都先转换为**double**型，然后进行运算
- (2) 如果**int**型与**float**或**double**型数据进行运算，先把**int**型和**float**型数据转换为**double**型，然后进行运算，结果是**double**型
- (3) 字符型数据与整型数据进行运算，就是把字符的**ASCII**代码与整型数据进行运算

3.2.7 运算符和表达式

例3.3 给定一个大写字母，要求用小写字母输出。

➤ 解题思路：

- ◆ 关键是找到大、小写字母间的内在联系
- ◆ 同一个字母，用小写表示的字符的**ASCII**代码比用大写表示的字符的**ASCII**代码大32

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

```
int main ( )
```

```
{
```

```
    char c1,c2;
```

```
    c1='A';           //将字符'A'的ASCII代码65放到c1中
```

```
    c2=c1+32;         //将65+32的结果放到c2中
```

```
    printf("%c\n",c2); //用字符形式输出
```

```
    printf("%d\n",c2); //用十进制形式输出
```

```
    return 0;
```

```
}
```

3.2.7 运算符和表达式

5. 强制类型转换运算符

➤ 强制类型转换运算符的一般形式为

(类型名) (表达式)

◆(double)a (将 a 转换成double类型)

◆(int) (x+y) (将x+y的值转换成int型)

◆(float)(5%3) (将5%3的值转换成float型)

➤ 有两种类型转换

◆ 系统自动进行的类型转换

◆ 强制类型转换

3.2.7 运算符和表达式

6. C 运算符

- | | |
|------------|-------------------|
| (1) 算术运算符 | (+ - * / % ++ --) |
| (2) 关系运算符 | (> >= < <= == !=) |
| (3) 逻辑运算符 | (! &&) |
| (4) 位运算符 | (<< >> ~ ^ &) |
| (5) 赋值运算符 | (=及其扩展赋值运算符) |
| (6) 条件运算符 | (? :) |
| (7) 逗号运算符 | (,) |
| (8) 求地址运算符 | (&) |
| (9) 间接运算符 | (*) |

3.2.7 运算符和表达式

6. C 运算符

- | | |
|-------------------|-----------------------------|
| (10) 求字节数运算符 | (sizeof) |
| (11) 强制类型转换运算符 | (()) |
| (12) 函数调用运算符 | (()) |
| (13) 结构/共用体成员运算符 | (.) |
| (14) 结构/共用体成指针运算符 | (->) |
| (15) 下标运算符 | ([]) |
| (16) 预处理运算符 | (# ##) |
| (17) 复合常量运算符 | ((){}) |
| (18) 其他 | (_Alignof _Pragma) |

3.3 C语句

3.3.1 C语句的作用和分类

3.3.2 最简单的语句--赋值表达式语句

3.3.1 C语句的作用和分类

C语言语句指定要执行的操作，除特别指定外，C语句按顺序依次执行

C语句分为以下**6**类：

- (1) 标号语句**labeled-statement**
- (2) 复合语句**compound-statement**
- (3) 表达式语句**expression-statement**
- (4) 选择语句**selection-statement**
- (5) 循环(迭代)语句**iteration-statement**
- (6) 跳转语句**jump-statement**

3.3.1 C语句的作用和分类

(1) 标号语句 **labeled-statement**

identifier: statement

case constant-expression: statement

default: statement

(2) 复合语句 **compound-statement**

复合语句就是一个语句块，是一组语句的集合

```
{  
    statementsopt  
}
```

3.3.1 C语句的作用和分类

(3) 表达式语句 **expression-statement**

expression_{opt};

; //空语句不执行任何操作，属表达式语句的一种

除赋值表达式语句和函数调用语句外，大部分表达式语句没有实际意义，如：**a+b;**

(4) 选择语句 **selection-statement**

选择语句根据控制表达式的值在一组语句中进行选择

if(expression) statement

if(expression) statement else statement

switch(expression) statement

3.3.1 C语句的作用和分类

(5) 循环(迭代)语句 **iteration-statement**

while(expression) statement

do statement while(expression);

**for(expression_{opt}; expression_{opt};
expression_{opt}) statement**

循环语句中被重复执行的语句成为循环体，直到控制表达式的值为**0**时，循环终止。

(6) 跳转语句 **jump-statement**

goto identifier; continue;

break; return expression_{opt};

3.3.2 最简单的语句--赋值表达式语句

➤ 在C程序中，最常用的语句是：

◆ 赋值语句

◆ 输入输出语句

➤ 其中最基本的是赋值语句

3.3.2 最简单的语句--赋值表达式语句

例**3.4** 给出三角形的三边长，求三角形面积。

- 假设给定的三个边符合构成三角形的条件
- 解题思路： **关键**是找到求三角形面积的公式
- 公式为：

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

其中 $s=(a+b+c)/2$

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

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

```
int main ( )
```

```
{
```

```
double a,b,c,s,area;
```

```
a=3.67;
```

```
b=5.43;
```

```
c=6.21;
```

对边长a、b、c赋值

```
s=(a+b+c)/2;
```

计算s

```
area=sqrt(s*(s-a)*(s-b)*(s-c));
```

计算area

```
printf("a=%f\tb=%f\t%f\n",a,b,c);
```

```
printf("area=%f\n",area);
```

```
return 0;
```

```
}
```

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

```
#include <math.h> 调用数学函数加此行
```

```
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\tc=%f\n",a,b,c);
```

```
  printf("area=%f\n",area);
```

数学函数，计算平方根

转义字符，使输出位置跳到下一个tab位置

```
a=3.670000
```

```
b=5.430000
```

```
6.210000
```

```
area=9.903431
```

➤ 归纳总结:

1.赋值运算符

- ◆ “=” 是赋值运算符

- ◆ 作用是将一个数据赋给一个变量

- ◆ 也可以将一个表达式的值赋给一个变量

➤ 归纳总结:

1. 赋值运算符

2. 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

◆ **$a+=3$** 等价于 **$a=a+3$**

➤ 归纳总结:

1. 赋值运算符

2. 复合的赋值运算符

3. 赋值表达式

◆ 一般形式为:

变量 赋值运算符 表达式

◆ 对赋值表达式求解的过程:

- 求赋值运算符**右侧**的“表达式”的值
- 赋给赋值运算符**左侧**的变量

➤ 归纳总结:

1. 赋值运算符

2. 复合的赋值运算符

3. 赋值表达式

◆ 赋值表达式 “ **$a=3*5$** ” 的值为**15**，对表达式求解后，变量 **a** 的值和表达式的值都是**15**

◆ “ **$a=(b=5)$** ” 和 “ **$a=b=5$** ” 等价

◆ “ **$a=b$** ” 和 “ **$b=a$** ” 含义不同

➤ 归纳总结:

1.赋值运算符

2.复合的赋值运算符

3.赋值表达式

4.赋值过程中的类型转换

- ◆ 两侧类型一致时，直接赋值

- ◆ 两侧类型不一致，但都是算术类型时，自动将右侧的类型转换为左侧类型后赋值

- ◆ 定义变量时要防止数据溢出

➤ 归纳总结:

1. 赋值运算符

2. 复合的赋值运算符

3. 赋值表达式

4. 赋值过程中的类型转换

5. 赋值表达式和赋值语句

- ◆ 赋值表达式的末尾没有分号，而赋值语句有分号
- ◆ 一个表达式可以包含赋值表达式，但决不能包含赋值语句

➤ 归纳总结:

- 1. 赋值运算符**
- 2. 复合的赋值运算符**
- 3. 赋值表达式**
- 4. 赋值过程中的类型转换**
- 5. 赋值表达式和赋值语句**
- 6. 变量赋初值**

int a=3,b=3,c;

int a=3; 相当于 **int a; a=3;**

➤ 简单程序设计的一般流程

- ◆ 确定程序运行过程中的量(常量、变量及其类型)
- ◆ 变量值的初始化
- ◆ 执行相关的运算，得到最终的结果
- ◆ 将结果输出

3.4 数据的输入输出

3.4.1 输入输出举例

3.4.2 有关数据输入输出的概念

3.4.3 用printf函数输出数据

3.4.4 用scanf函数输入数据

3.4.5 字符数据的输入输出

3.4.1 输入输出举例

例**3.5** 求 $ax^2 + bx + c = 0$ 方程的根。

a、b、c由键盘输入

假设 $b^2 - 4ac > 0$

3.4.1 输入输出举例

- 解题思路：首先要知道求方程式的根的方法。
- 由数学知识已知：如果 $b^2 - 4ac \geq 0$ ，则一元二次方程有两个实根：

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

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

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

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

```
#include <math.h>  程序中调用数学函数sqrt
```

```
int main ( )
```

```
{double a,b,c,delta,x1,x2,p,q;
```

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

```
delta=b*b-4*a*c;  输入a,b,c的值
```

```
p=-b/(2.0*a);
```

```
q=sqrt(delta)/(2.0*a);
```

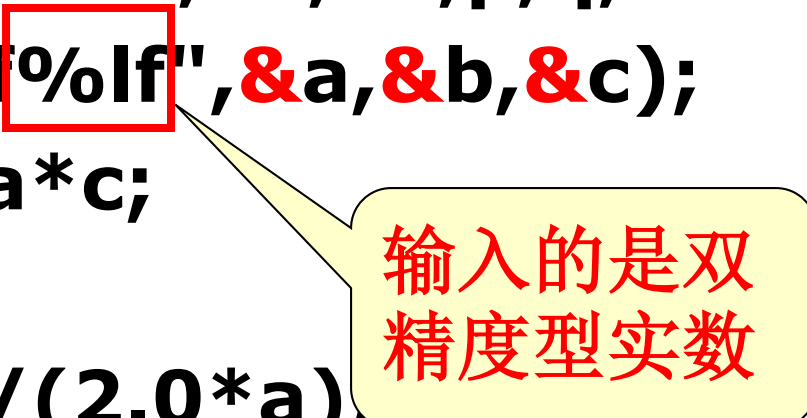
```
x1=p+q;  x2=p-q;
```

```
printf("x1=%7.2f\nx2=%7.2f\n",x1,x2);
```

```
return 0;
```

```
}
```

```
#include <stdio.h>
#include <math.h>
int main ( )
{double a,b,c, delta,x1,x2,p,q;
  scanf("%lf%lf%lf",&a,&b,&c);
  delta=b*b-4*a*c;
  p=-b/(2.0*a);
  q=sqrt(delta)/(2.0*a);
  x1=p+q;  x2=p-q;
  printf("x1=%7.2f\nx2=%7.2f\n",x1,x2);
  return 0;
}
```



输入的是双精度型实数

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

```
int main ( )
```

```
{double a,b,c, delta,x1,x2,p,q;
```

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

```
delta=b*b-4*a*c;
```

```
p=-b/(2.0*a);
```

```
q=sqrt(delta)/(2.0*a);
```

```
x1=p+q; x2=p-q;
```

```
printf("x1=%7.2f\nx2=%7.2f\n",x1,x2);
```

```
return 0;
```

```
}
```

1 3 2

自动转成实数
后赋给a,b,c

要求输入3个实数

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

```
int main ( )
```

```
{double a,b,c, delta,x1,x2,p,q;
```

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

```
delta=b*b-4*a*c;
```

```
p=-b/(2.0*a);
```

```
q=sqrt(delta)/(2.0*a);
```

```
x1=p+q; x2=p-q;
```

```
printf("x1=%7.2f\nx2=%7.2f\n",x1,x2);
```

```
return 0;
```

```
}
```

```
1 3 2
x1= -1.00
x2= -2.00
```

输出数据占7列，其中小数占2列

3.4.2 有关数据输入输出的概念

- 几乎每一个**C**程序都包含输入输出
- 输入输出是程序中最基本的操作之一

(1) 所谓输入输出是以计算机主机为主体而言的

- ◆ 从计算机向输出设备(如显示器、打印机等)输出数据称为输出
- ◆ 从输入设备(如键盘、磁盘、光盘、扫描仪等)向计算机输入数据称为输入

3.4.2 有关数据输入输出的概念

(2) 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

- ◆**printf**和**scanf**不是C语言的关键字，而只是库函数的名字；这两个函数是格式输入输出函数，在用时，必须指定格式

- ◆**putchar**、**getchar**、**puts**、**gets**

(3) 在使用输入输出函数时，要在程序文件的开头用预编译指令

- ◆**#include <stdio.h>**

3.4.3 用printf函数输出数据

1.printf函数的一般格式

printf(格式控制, 输出表列);

例如:

printf("i=%d,c=%c\n",i,c);

格式声明

3.4.3 用printf函数输出数据

1.printf函数的一般格式

printf(格式控制, 输出表列);

例如:

printf("i=%d,c=%c\n",i,c);

普通字符

3.4.3 用printf函数输出数据

1.printf函数的一般格式

printf(格式控制，输出表列)

例如：

printf("i=%d,c=%c\n", **i,c**);

可以是常量、变量或表达式

3.4.3 用printf函数输出数据

2. 常用格式字符

◆ d 格式符。用来输出一个有符号的十进制整数

●可以在格式声明中指定输出数据的域宽

```
printf("%5d%5d\n",12,-345);
```

●**%d**输出**int**型数据

●**%ld**输出**long**型数据

3.4.3 用printf函数输出数据

2. 常用格式字符

◆ c 格式符。用来输出一个字符

```
char ch='a';
```

```
printf("%c",ch); 或
```

```
printf("%5c",ch);
```

输出字符： a

3.4.3 用printf函数输出数据

2. 常用格式字符

◆ s 格式符。用来输出一个字符串

```
printf("%s","CHINA");
```

输出字符串：**CHINA**

3.4.3 用printf函数输出数据

2. 常用格式字符

◆**f**格式符。用来输出实数，以小数形式输出

①不指定数据宽度和小数位数，用**%f**

例**3.6** 用**%f**输出实数，只能得到 6 位小数。

```
double a=1.0;
```

```
printf("%f\n",a/3);
```

```
0.333333
```

3.4.3 用printf函数输出数据

2. 常用格式字符

◆**f**格式符。用来输出实数，以小数形式输出

② 指定数据宽度和小数位数。用**%m.nf**

```
printf("%20.15f\n",1/3);
```

```
0.333333333333333
```

```
printf("%.0f\n",10000/3.0);
```

```
3333
```


3.4.3 用printf函数输出数据

2. 常用格式字符

◆**f**格式符。用来输出实数，以小数形式输出

② 指定数据宽度和小数位数。用**%m.nf**

```
float a;
```

```
a=10000/3.0;
```

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

```
3333.333333
```

3.4.3 用printf函数输出数据

2. 常用格式字符

◆**f**格式符。用来输出实数，以小数形式输出

③ 输出的数据向左对齐，用**%-m.nf**

3.4.3 用printf函数输出数据

2. 常用格式字符

- ◆**f**格式符。用来输出实数，以小数形式输出
 - float**型数据只能保证**6**位有效数字
 - double**型数据能保证**15**位有效数字
 - 计算机输出的数字不都是绝对精确有效的

3.4.3 用printf函数输出数据

2. 常用格式字符

◆**e**格式符。指定以指数形式输出实数

●**%e**，**VS**中给出小数位数为 6 位

指数部分占**5**列

小数点前必须有而且只有**1**位非零数字

```
printf("%e",123.456);
```

输出: **1.234560 e+002**

3.4.3 用printf函数输出数据

2. 常用格式字符

◆**e**格式符。指定以指数形式输出实数

●**%m.ne**

```
printf("%13.2e",123.456);
```

输出: **1.23e+002** (前面有**4**个空格)

3.4.4 用scanf函数输入数据

1. scanf 函数的一般形式

scanf(格式控制, 地址表列)

含义同printf函数

3.4.4 用scanf函数输入数据

1. scanf 函数的一般形式

scanf(格式控制, 地址表列)

可以是变量的地址，或字符串的首地址

3.4.4 用scanf函数输入数据

2. scanf函数中的格式声明

- 与**printf**函数中的格式声明相似
- 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);
```


3.4.4 用scanf函数输入数据

3.使用scanf函数时应注意的问题

`scanf("%f%f%f",a,b,c);` 错

`scanf("%f%f%f",&a,&b,&c);` 对

对于

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

1 3 2 ☒

错

a=1,b=3,c=2 ☒

对

a=1 b=3 c=2 ☒

错

3.4.4 用scanf函数输入数据

3.使用scanf函数时应注意的问题

对于scanf("%c%c%c",&c1,&c2,&c3);

abc ☒

对

a b c ☒

错

对于scanf("%d%c%f",&a,&b,&c);

若输入

1234a123o.26 ☒

3.4.4 用scanf函数输入数据

3.使用scanf函数时应注意的问题

对于scanf("%c%c%c",&c1,&c2,&c3);

abc ☒

对

a b c ☒

错

对于scanf("%d%c%f",&a,&b,&c);

若输入

1234a123o.26 ☒

3.4.4 用scanf函数输入数据

3.使用scanf函数时应注意的问题

对于scanf("%c%c%c",&c1,&c2,&c3);

abc ☒

对

a b c ☒

错

对于scanf("%d%c%f",&a,&b,&c);

若输入

1234a 123 o.26 ☒

3.4.5 字符数据的输入输出

1. 用**putchar**函数输出一个字符

- 从计算机向显示器输出一个字符
- **putchar**函数的一般形式为：

putchar(c)

3.4.5 字符数据的输入输出

例**3.8** 先后输出**BOY**三个字符。

➤ 解题思路：

- ◆ 定义**3**个字符变量，分别赋以初值**B**、**O**、**Y**
- ◆ 用**putchar**函数输出这**3**个字符变量的值

3.4.5 字符数据的输入输出

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

```
int main ( )
```

```
{
```

```
    char a='B',b='O',c='Y';
```

```
    putchar(a);    向显示器输出字符B
```

```
    putchar(b);
```

```
    putchar(c);
```

```
    putchar ('\n'); 向显示器输出换行符
```

```
    return 0;
```

```
}
```



BOY

3.4.5 字符数据的输入输出

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

```
int main ( )
```

改为int a=66,b=79,c=89;

```
{
```

```
    char a='B',b='O',c='Y';
```

```
    putchar(a);
```

```
    putchar(b);
```

```
    putchar(c);
```

```
    putchar ('\n');
```

```
    return 0;
```

```
}
```

BOY

3.4.5 字符数据的输入输出

putchar('\101') (输出字符 A)

putchar('\') (输出单撇号字符')

3.4.5 字符数据的输入输出

2. 用**getchar**函数输入一个字符

- 向计算机输入一个字符
- **getchar**函数的一般形式为：

getchar()

3.4.5 字符数据的输入输出

例**3.9** 从键盘输入**BOY**三个字符，然后把它们输出到屏幕。

➤ 解题思路：

◆ 用**3**个**getchar**函数先后从键盘向计算机输入**BOY**三个字符

◆ 用**putchar**函数输出

3.4.5 字符数据的输入输出

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

```
int main ( )
```

```
{ char a,b,c;
```

```
    a=getchar();    输入一个字符，送给变量a
```

```
    b=getchar();
```

```
    c=getchar();
```

```
    putchar(a); putchar(b); putchar(c);
```


```
    putchar('\n');
```

```
    return 0;
```

```
}
```



BOY
BOY



B
O
Y

3.4.5 字符数据的输入输出

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

```
int main ( )
```

```
{ char a,b,c;
```

```
    a=getchar(); putchar(getchar());
```

```
    b=getchar();
```

```
    c=getchar();
```

```
    putchar(a); putchar(b); putchar(c);
```

```
    putchar('\n');
```

```
    return 0;
```

```
}
```

3.4.5 字符数据的输入输出

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

```
int main ( )
```

```
{ char a,b,c;
```

```
    putchar(getchar());
```

```
    b=getchar();
```

```
    putchar(getchar());
```

```
    c=getchar();
```

```
    putchar(b); putchar(c);
```

```
    putchar('\n');
```

```
    return 0;
```

```
}
```

3.4.5 字符数据的输入输出

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

```
int main ( )
```

```
{ char a,b,c;
```

```
    putchar(getchar());
```

```
    putchar(getchar());
```

```
    c=getchar();
```

```
    putchar(getchar());
```

```
    putchar(c);
```

```
    putchar('\n');
```

```
    return 0;
```

```
}
```

3.4.5 字符数据的输入输出

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

```
int main ( )
```

```
{ char a,b,c;
```

```
    putchar(getchar());
```

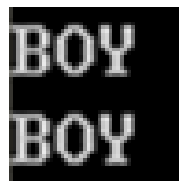
```
    putchar(getchar());
```

```
    putchar(getchar());
```

```
    putchar('\n');
```

```
    return 0;
```

```
}
```



BOY
BOY