

第3章 最简单的C程序设计

3.1 顺序程序设计举例

3.2 数据的表现形式及其运算

3.3 C语句

3.4 数据的输入输出

3.1 顺序程序设计举例

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

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

f代表华氏温度，**c**代表摄氏温度



3.1 顺序程序设计举例

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：

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

N-S图



3.1 顺序程序设计举例

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

```
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 顺序程序设计举例

例**3.2** 计算存款利息。有**1000**元，想存一年。有三种方法可选：

(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三种方法所得到的本息和。



3.1 顺序程序设计举例

➤ 解题思路：确定计算本息和的公式。

从数学知识可知：若存款额为**p0**，则：

活期存款一年后本息和为：

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

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：

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



3.1 顺序程序设计举例

➤ 算法:

输入p0,r1,r2,r3的值
计算 $p1=p0(1+r1)$
计算 $p2=p0(1+r2)$
计算 $p3=p0(1+\frac{r3}{2})(1+\frac{r3}{2})$
输出p1,p2,p3



3.1 顺序程序设计举例

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

定义变量同时赋予初值

```
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("%f\n%f\n%f\n", p1, p2, p3);
```

```
return 0;
```

```
}
```

```
1003.599976  
1022.500000  
1019.898010
```



3.2 数据的表现形式及其运算

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：如**1000**, **12345**, **0**, **-345**

➤ 实型常量

◆ 十进制小数形式：如**0.34** **-56.79** **0.0**

◆ 指数形式：如**12.34e3** (代表 **12.34×10^3**)

➤ 字符常量：如**'?'**

◆ 转义字符：如**'\n'**

➤ 字符串常量：如**"boy"**

➤ 符号常量：**#define** **PI** **3.1416** (不加**;**)

(特别注意，改错！)



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. 常变量: **const int a=3;**

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**

注意: C语言中大小写是有区别的, 改错!



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



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
- 短整型
- 长整型
- 双长整型
- 字符型
- 布尔型

◆ 浮点类型

- 单精度浮点型
- 双精度浮点型
- 复数浮点型



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

算术类型

纯量类型

- ◆ 指针类型
- ◆ 数组类型
- ◆ 结构体类型
- ◆ 共用体类型
- ◆ 函数类型



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**型): 占**2**个或**4**个字节
- ◆ 短整型(**short int**): **CodeBlocks**中占**2**个字节
- ◆ 长整型(**long int**): **CodeBlocks**中占**4**个字节
- ◆ 双长整型(**long long int**): **C99**新增的



3.2.3 整型数据

1. 整型数据的分类

2. 整型变量的符号属性

- ◆ 整型变量的值的范围包括负数到正数
- ◆ 可以将变量定义为“无符号”类型
- ◆ 扩充的整形类型：



3.2.3 整型数据

扩充的整型类型:

- 有符号基本整型 **[signed] int;**
- 无符号基本整型 **unsigned int;**
- 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]** 

不同类型取值范围

short a //有符号短整型 -32768~32767

unsigned short b 0~65535

详见表3.2

不能给无符号整型赋值负数

unsigned short price = -1;

printf("%d\n",price); 65535

- **C**标准并未规定各种类型占用的长度
- 注意：不同标准的要求可能不同
详见 **P45** 说明

3.2.4 字符型数据

- 字符是按其代码(整数)形式存储的
- **C99**把字符型数据作为整数类型的一种
- 字符型数据在使用上有自己的特点

以' ' 符号表示字符

注意：仅存放一个字符



3.2.4 字符型数据

1. 字符与字符代码

大多数系统采用**ASCII**字符集

- ◆字母: **A ~ Z, a ~ z**
- ◆数字: **0 ~ 9**
- ◆专门符号: **29个: ! " # & ' () * 等**
- ◆空格符: 空格、水平制表符、换行等
- ◆不能显示的字符: 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**2**个或**4**个字节

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



3.2.4 字符型数据

2. 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 ?

char 与 unsigned char 的区别

详见P49 注释



➤ P49 脚注

char 表示有符号字符，其值范围**-128~127**

unsigned char 无符号字符，其值**0~255**

在中文操作系统下，超过**127**以后部分都当做
中文字符处理，不显示扩展字符。

实际在显示时候显示 **'?'**

```
#include <stdio.h>

int main()
{
    unsigned char c1,c2;
    c1=65;
    c2=129;
    printf("c1=%c,c2=%c\n",c1,c2);
    printf("c1=%d,c2=%d\n",c1,c2);
    return 0;
}
```

c1=A ,c2=?

c1=65,c2=129

3.2.5 浮点型数据

浮点型数据是用来表示具有小数点的实数

➤ **float**型(单精度浮点型)

- ◆ 编译系统为**float**型变量分配**4**个字节
- ◆ 数值以规范化的二进制数指数形式存放

参见主教材图**3.11**



3.2.5 浮点型数据

浮点型数据是用来表示具有小数点的实数

➤ **float**型(单精度浮点型)

➤ **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



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**int**型还是**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



➤ 注意常量与变量的区别

float a = 3.14159; 与

3.14159 的区别

前者为**float**类型 **a**为变量

后者为**double**类型 为常量

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（特别注意，改错！）
 - ◆ 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
 - ◆ **CodeBlocks**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



➤注意：**C语言中*（乘号）不可以省略**
（改错）

例如：

数学表达式： **$2a-3b$** 在**C语言**中写成：
 $2*a-3*b$

3.2.7 运算符和表达式

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**

例:

$i=3$;

$y=i++$; (此时 **y** 的值为**3**)

$y=++i$; (此时 **y** 的值为**4**)



编程的规范性

1、编程语句的可读性：

不要写出自己不易理解别人也不看不懂的语句

例如：**i+++j** 是什么意思？

在需要的时候可以使用括号提高程序的可读性

2、语句的整洁

配对的花括号在同一列上对齐

3、增加必要的注释

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



3.2.7 运算符和表达式

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



假设 **i**为**int** **f**为**float** **d**为**double**

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

类型转换为系统自动完成

若**int j** (值为**7**)

以下表达式的值为多少？

$10 + 'A' + j / 2$

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**



3.2.7 运算符和表达式

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

```
int main ( )
```

```
{
```

```
    char c1,c2;
```

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

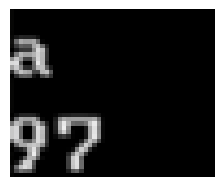
```
    c2=c1+32; 将65+32的结果放到c2中
```

```
    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;
```

```
}
```



a
97



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) 注意：有括号

◆ **(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



➤ 注意强制类型转换运算符的优先级

如: **float x=3.5, y = 4.1**

(int) (x+y) = ?

(int) x+y = ?

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 | (? :) |



3.2.7 运算符和表达式

6. C 运算符

- | | |
|----------------|-------------------|
| (7) 逗号运算符 | (,) |
| (8) 指针运算符 | (* 和 &) |
| (9) 求字节数运算符 | (sizeof) |
| (10) 强制类型转换运算符 | ((类型)) |
| (11) 成员运算符 | (. ->) |
| (12) 下标运算符 | ([]) |
| (13) 其他 | (如函数调用运算符 ()) |



3.3 C语句

3.3.1 C语句的作用和分类

3.3.2 最基本的语句-----赋值语句



3.3.1 C语句的作用和分类

C 语句分为以下**5**类：

- (1) 控制语句：**if、switch、for、while、do...while、continue、break、return、goto**等
- (2) 函数调用语句
- (3) 表达式语句
- (4) 空语句
- (5) 复合语句



3.3.2 最基本的语句----赋值语句

➤ 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 最基本的语句----赋值语句

例**3.4** 给出三角形的三边长，求三角形面积。



3.3.2 最基本的语句----赋值语句

- 解题思路：假设给定的三个边符合构成三角形的条件
- 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;
  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);
  return 0;
}
```

对边长a、b、c赋值

计算s

计算area



```
#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\t%f\n",a,b,c);
  printf("area=%f\n",area);
  return 0;
}
```

数学函数，计算平方根



```
#include <stdio.h>
#include <math.h> 调用数学函数加此行(改错)
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);
```

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



赋值过程中类型转换

- 浮点数赋值给整数（取整，舍弃小数）
- 整数赋值给浮点数，值不变
- **double**赋值给**float**，精度丢失
- 字符型赋值给整数，存放**ASCII**码
- 多字节赋值给少字节变量时，**截断**
（低字节原封不动赋值）

赋值过程中类型转换

- 浮点数赋值给整数（取整，舍弃小数）

```
double d=7.567; 7.467
```

```
int i;
```

```
i=d;
```

```
printf("i=%d",i);
```

四舍五入

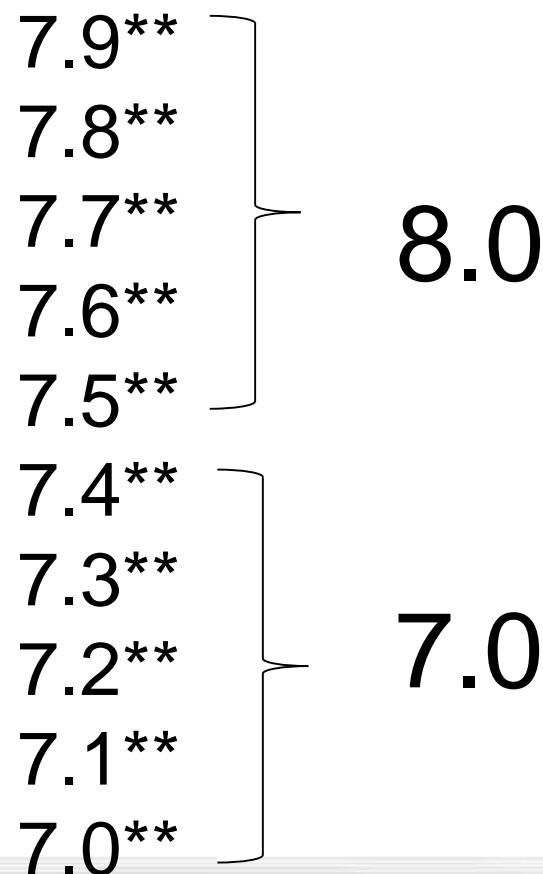
➤ 浮点数赋值给整数（取整，舍弃小数）

```
double d=7.567;
```

```
int i;
```

```
i=d;
```

```
printf("i=%d",i);
```



四舍五入

➤ 浮点数赋值给整数（取整，舍弃小数）

```
double d=7.567;
```

```
int i;
```

```
i=int(d+0.5);
```

```
printf("i=%d",i);
```

思考：若要保留小数点后两位，怎么处理？

提示：将需要处理的数变成“十分位”（小数点后一位）

注意：负数怎么处理？

c=33, c='!'

```
int a = 32767;
```

```
short b;
```

```
b=a+1;
```

```
printf("b=%d\n",b);
```

b= -32768

详见课本

➤ 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

注意：改错

int a=3,b=3,c; 但是不能 **int a=b=c=3;**
int a=3; 相当于 **int a; a=3;**



int a=3,b=3,c=3; 对

int a=b=c=3; 错

**int a,b,c;
a=b=c=3; 对**

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

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,disc,x1,x2,p,q;
```

这是初学者最容
易犯的错误！

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

 (注意：改错)

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

输入a,b,c的值

```
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);
```

```
return 0;
```

```
}
```



```
#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);
return 0;
}
```

输入的是双精度
型实数，改错！



```
#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);
```

```
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,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);
```

```
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



3.4.2 有关数据输入输出的概念

(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
- **putchar**、**getchar**、**puts**、**gets**



3.4.2 有关数据输入输出的概念

(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>

或

#include "stdio.h"



3.4.3 用printf函数输出数据

- 在C程序中用来实现输出和输入的，主要是**printf**函数和**scanf**函数
- 这两个函数是格式输入输出函数
- 用这两个函数时，必须指定格式



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，只能显示 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); 1.0/3 或 1/3.0

A black rectangular box containing the text "0.3333333333333333" in white, representing the output of the first printf statement.

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

A black rectangular box containing the text "3333" in white, representing the output of the second printf statement.

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**，**VC++**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（改错题）

对于普通字符，需要原样输入

`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✓



➤ 在输入数值数据时，数据结束的标记为空格、回车、**Tab**键或其他的非数字符（即不属于数值的字符） 见课本**P77**

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

scanf("%d\n",&d1); 错!

scanf中不出现\n



scanf函数易错点

➤ **scanf("%f", a);**

&符号遗漏

➤ **int a;**

scanf("%f", &a);

控制符前后不一致 尤其注意double → %lf

➤ **scanf("a=%lf", &a);** 输入 **18.8(回车)**

普通字符要原样输入, 输入 a=18.8(回车)

➤ **scanf("%d\n",&c);**

不能出现 \n

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

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

```
int main ( )
```

```
{
```

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

```
    putchar(`a');
```

```
    putchar(`b');
```

```
    putchar(`c');
```

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

```
    return 0;
```

```
}
```

输出: abc



3.4.5 字符数据的输入输出

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

```
int main ( )
```

```
{
```

```
    putchar(65);
```

```
    putchar(66);
```

```
    putchar(67);
```

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

```
    return 0;
```

```
}
```

输出： ABC



➤ 给一个字符变量赋值字符常量与字符的
ASCII码是等效的。

如 **char ch='a';** 与
char ch=97; 是等效的

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
B
O



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;
```

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

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

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

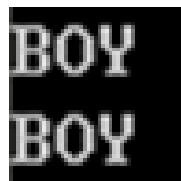
```
    putchar(getchar());
```

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

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

```
    return 0;
```

```
}
```



BOY
BOY

