万水创作效果百例丛书


# C语言精彩编程百例

温海 张友 童伟 等编著

中国水利水电出版社
www.waterpub.com.cn

万水创作效果百例丛书

# C 语言精彩编程百例

温海　张友　童伟　等编著

中国水利水电出版社

## 内 容 提 要

C 是一种通用的程序设计语言，它包含了紧凑的表达式、丰富的运算符集合、现代控制流以及数据结构等四个部分。C 语言功能丰富，表达能力强，使用起来灵活方便；它应用面广，可移植性强，同时具有高级语言和低级语言的优点，因此，在工程计算及应用程序开发中得到了广泛的应用。

众所周知，学习新的程序设计语言的最佳途径是编写程序，而本书正是通过了对 100 个典型实例的分析和讲解，来帮助读者掌握这门语言并积累大量经验，从而可以熟练地进行 C 程序设计。

全文共分为四篇，全面、系统地讲述了 C 语言各个方面的知识点和程序设计的基本方法，以及编写程序过程中值得注意的地方，内容深入浅出，通俗易懂。对于 C 语言的初学者来说，这是一本绝对好的入门教材，对于有经验的专业人员，也会发现本书很有价值。

**图书在版编目（CIP）数据**

C 语言精彩编程百例/温海等编著. —北京：中国水利水电出版社，2003
（万水创作效果百例丛书）
ISBN 7-5084-1818-2

Ⅰ. C… Ⅱ. 温… Ⅲ. C 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2003）第 102473 号

| | |
|---|---|
| 书　　名 | C 语言精彩编程百例 |
| 作　　者 | 温海　张友　童伟　等编著 |
| 出版、发行 | 中国水利水电出版社（北京市三里河路 6 号　100044） |
| | 网址：www.waterpub.com.cn |
| | E-mail：mchannel@public3.bta.net.cn（万水） |
| | sale@waterpub.com.cn |
| | 电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水） |
| 经　　售 | 全国各地新华书店和相关出版物销售网点 |
| 排　　版 | 北京万水电子信息有限公司 |
| 印　　刷 | 北京北医印刷厂 |
| 规　　格 | 787×1000 毫米　16 开本　　22.75 印张　　499 千字 |
| 版　　次 | 2004 年 1 月第一版　2004 年 1 月北京第一次印刷 |
| 印　　数 | 0001—5000 册 |
| 定　　价 | 34.00 元 |

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

# 前　言

C 是一种通用的程序设计语言，它包含了紧凑的表达式、丰富的运算符集合、现代控制流以及数据结构等四个部分。C 语言功能丰富，表达能力强，使用起来灵活方便。它应用面广，可移植性强，同时具有高级语言和低级语言的优点，因此，在工程计算及应用程序开发中得到了广泛的应用。

经验告诉我们，学习新的程序语言的最佳途径就是亲手实践，而本书正是通过对 100 个典型实例的分析和讲解，来帮助读者掌握这门语言并积累大量经验，从而可以熟练地进行 C 程序设计。

全书分为四篇，按照由浅及深、循序渐进的原则，全面、系统地讲解了 C 语言各个方面的知识点和常用的程序设计基本技巧，以及编写程序过程中值得注意的地方，内容深入浅出，通俗易懂。

第一篇由 44 个实例组成，主要侧重于对 C 语言基础知识的介绍，集中大量实例来讲解指针、数组以及位操作等对于 C 语言初学者而言较难理解的知识点。

第二篇由 26 个实例组成，覆盖自定义结构类型、I / O 操作及部分常用函数，至此读者已拥有独立完成一般程序设计的能力。

第三篇由 23 个实例组成，以 C 语言的常用算法为主,重点向读者演示了数据结构的应用以及常用的数值算法的 C 语言实现方法。

第四篇由 7 个实例组成，这一篇的作用是验证、提升读者的编程能力，作为前三篇的总结应用，涵盖了本书中所介绍的大部分知识点。

我们希望读者通过对这本书的学习 ，能够具有较强的 C 语言编程技能，具备一定的独立编程能力，本书所附带的程序源代码都已经过编译，读者可验证程序运行的结果。

在这里感谢中国水利水电出版社给我们这次机会，使我们能够把自己的经验传授给广大读者；同时感谢出版社的编辑们，是他们的认真监督和辛勤工作才能使该书顺利完成；还要感谢参与本书编写的李伯苓、王晓霞、刘逸飞、段广仁、谭友利、辛知庆、李冠、任宝山、吴仪委等人，是他们的辛勤劳动使本书顺利的与大家见面。

由于编者水平有限，书中缺点和错误在所难免，恳请广大读者批评指正，并提出宝贵的意见和建议。

编者

2003 年 10 月

# 目　　录

## 第一篇　基础知识篇

## 第二篇　深入提高篇

## 第三篇　常用算法篇

# 第四篇　综合应用篇

# 第一篇
# 基础知识篇

想成为一个优秀的C程序员，首先要做的就是熟练掌握C语言的各基本要素，并会正确使用它们，因为C语言的基本要素是所有C程序的根本和基石。基础篇作为全书的第一篇，其目的就是通过程序实例向读者介绍C语言的基本要素，让读者能够尽可能快地掌握它们。

对于成功的C程序设计而言，正确理解并熟练使用指针是至关重要的，所以指针这部分内容是本篇的重点。由于指针的概念比较抽象，对于初学者而言很难一下子理解，因此在这里列举了大量的实例，并作了全面、详尽的分析，希望读者通过对这些程序的学习，能够加深对指针的理解；希望读者通过本篇的学习，能够模仿书中提供的实例编写一些小程序，以便对C程序设计有初步的了解。

# 实例1 数据类型转换

## 实例说明

本例旨在介绍数据的类型转换。C语言规定，不同类型的数据需要转换成同一类型后才可进行计算，在整型、实型和字符型数据之间通过类型转换便可以进行混合运算。除了上述内容，我们还要介绍一些常用算术运算符的优先级与结合性，希望读者能够熟记。

注意点：并非所有类型的数据之间都可以进行转换，例如，指针和上述三种类型数据之间不能够进行类型换算。

## 知识要点

当混合不同类型的变量进行计算时，便可能会发生类型转换。

相同类型的数据在转换时有规则可循，如字符必定先转换为整数（C语言规定字符类型数据和整数数据之间可以通用），short型转为int型（同属于整型），float型数据在运算时一律转换为双精度（double）型，以提高运算精度（同属于实型）。

不同类型的数据发生转换时，遵循低级类型向高级类型转换的原则，例如int型数据与double型数据进行运算时，是先将int型数据转换成double类型，然后再进行运算，结果为double类型。

此外，在一个赋值语句中，若发生类型转换，则是赋值语句右部（表达式一侧）的值转换成左部（目标一侧）的类型。

## 程序源码

该应用程序的源代码如下：

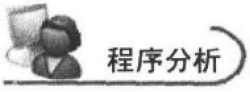
```
# include <stdio.h>

void main()
{
    // 定义变量并赋初值
    int    a = 5;
    char   c = 'a';
    float  f = 5.3;
    double m = 12.65;
  double result;
```

```
    // 同类型数据间进行运算并输出结果
    printf("a + c = %d\n", a + c);
    printf("a + c = %c\n", a + c);
    printf("f + m = %f\n", f + m);

    // 不同类型数据间进行运算并输出结果
    printf("a + m = %f\n", a + m);
    printf("c + f = %f\n", c + f);

    // 将上述四个变量进行混合运算，并输出结果
    result = a + c * (f + m);
    printf("double = %f\n", result);
}
```




程序中分别定义了一个整型数据 a，一个字符型数据 c，以及两个实型数据 f 和 m。

当整型数据和字符型数据进行运算时，结果会随输出格式说明的不同而不同，当结果以整型输出格式“%d”输出时，结果为整数，若以字符型输出格式“%c”输出时，结果为字符。

当整型数据和双精度型数据进行运算时，C 先将整型数据转换成双精度型数据，再进行运算，结果为双精度类型的数据。同样，当字符型数据和实型数据进行运算时，C 先将字符型数据转换成实型数据，然后进行计算，结果为实型数据。

在表达式求解时，按运算符的优先级别的高低次序执行，例如先乘除后加减。若在一个运算对象两侧的运算符的优先级别相同，那么按照“自左向右”的方向进行结合，但若在表达式中存在括号，则括号中运算的优先级别最高，最先被执行，所以程序中算式 a + c * (f + m)的运算次序为，先执行(f + m)中的运算，然后将其结果与 c 相乘，最后同 a 相加。

请注意，代码行中的“=”是赋值运算符，不属于算术运算符。赋值运算符的结合性是按照“自右向左”的规则执行的。因此，在代码行 result = a + c * (f + m)中，是先得出算式 a + c * (f + m)的结果，而后再将此结果赋给双精度变量 result。

# 实例2 转义字符

## 实例说明

C中的字符常量是用单引号括起来的一个字符。此外，C还允许一种特殊形式的字符常量，就是以一个"\"开头的字符序列，通常称它们为转义字符。

鉴于转义字符的特殊性，在此有必要作出介绍。在实例当中会看到一些常用的转义字符，如换行符、回车符等。通过对例题的学习，希望读者能够理解这些常用转义字符的含义，并能够在今后的编程中熟练使用它们。

## 知识要点

转义字符是C语言中表示字符的一种特殊形式。

通常使用转义字符表示 ASCII 码字符集中不可打印的控制字符和特定功能的字符，如用于表示字符常量的单撇号（'），用于表示字符串常量的双撇号（"）以及反斜杠（\）等。转义字符用反斜杠（\）后面跟一个字符或一个八进制或十六进制数表示。

字符常量中使用单引号和反斜杠以及字符常量中使用双引号和反斜杠时，都必须使用转义字符表示，即在这些字符前加上反斜杠。

使用转义字符时需要注意以下三点问题：

（1）转义字符中只能使用小写字母，每个转义字符只能看作一个字符。

（2）\v 垂直制表和\f 换页符对屏幕没有任何影响，但会影响打印机执行响应操作。

（3）在C程序中，使用不可打印字符时，通常用转义字符表示。

## 程序源码

该应用程序的源代码如下：

```
# include <stdio.h>3

void main()
{
    // 换行符'\n'，用于输出换行
    printf("How are you?\n");
    printf("I am fine.\n\n");

    // 横向跳格符'\t'，使跳到下一个输出区
```

```
    printf("How are you?\t");
    printf("I am fine.\n\n");

    // 退格符'\b'，使当前的输出位置退一格，即输出的起始位置左移一位
    printf(" How are you?\n");
    printf(" \bI am fine.\n\n");

    // 回车符'\r'，使当前输出位置回到本行开头
    printf("                I am fine.");  // I 前面共有 16 个空格
    printf("\rHow are you?\n\n");

    // 多个转义字符的混合运用
    printf("note:\n  a s\ti\b\bk\rp\n");
}
```

## 程序分析

程序中共有五个输出模块，前四个输出模块都只用到了一个转义字符，不难得出结果。在第五个输出模块中，综合用到了前四个输出模块中的转义字符，在此，我们将重点分析第五个输出模块。

printf 函数先在当前行输出“note:”，然后换行。程序的输出位置跳到第二行后，首先从左端开始输出 “ a s”（注意，在字符 a 的左右两边各有一个空格），然后遇到“\t”,它的作用是跳格，即跳到下一个输出位置，在我们所用的输出系统中，一个输出区占 8 列（即 8 个空格）。则下一输出位置从第 9 列开始，所以在第 9 列上输出“i”。下面遇到两个“\b”，“\b”的作用是退一格，因此，“\b\b”的作用是使当前输出位置（第十列）退回到第 8 列输出“k”。最后，遇到“\r”，它代表回车（不换行），此时输出返回到本行的最左端（第一列），输出字符“p”。

所以第五个输出模块的最终输出是：

```
note:
p a s  ki
```

# 实例 3 关系和逻辑运算

## 实例说明

这是一个介绍关系运算和逻辑运算的 C 程序实例。程序向读者介绍了所有的六种关系运算符和三种逻辑运算符，以及它们的优先级次序，旨在使读者通过此例能够熟练掌握它们，并能够对它们进行简单运用。

此外，本例还要让读者确立这样一种概念，那就是真（true）和假（false）的思维是关系和逻辑操作符概念的基础。在 C 中，true 代表非零值，false 代表零值。使用逻辑或关系操作符的表达式返回零作为假值，返回 1 作为真值。

## 知识要点

关系运算符中的“关系”二字指的是一个值与另一个值之间的关系，逻辑运算符中的“逻辑”二字指的是连接关系的方式。因为关系和逻辑运算符常在一起使用，所以在此将它们放在一起讨论。

下面列出的是关系和逻辑操作符的相对优先级：

最高　　!

　　>　　>=　　<　　<=

　　==　　!=

　　&&

最低　　||

需要注意的是，除运算符“!”之外，所有关系和逻辑操作符的优先级都低于算术操作符，也就是说，当算式中同时有算术操作符、关系和逻辑操作符（“!”除外）时，是在执行完所有的算术运算后，才开始执行关系和逻辑运算。

此外，同算术表达式一样，在关系或逻辑表达式中也可使用括号来修改原来的计算顺序。

## 程序源码

该应用程序的源代码如下：

```
# include <stdio.h>

void main()
{
```

```
    // 定义一个整数类型的变量，用来存放后面算式的值
    int logic;

    int a = 1;
    int b = 2;
    int c = 3;

    logic = a+b>c&&b<=c;
    printf("logic = %d\n", logic);

    logic = a>=b+c||b==c;
    printf("logic = %d\n", logic);

    logic = !(a<c)+b!=1&&(a+c)/2;
    printf("logic = %d\n", logic);
}
```

## 程序分析

程序中的三个输出是 0、0 和 1，即分别为假、假、真。下面分析一下程序中三个算式的运算顺序。

算式一：a+b>c&&b<=c，实际上可表示成((a+b)>c)&&(b<=c)。C 首先进行算术运算 a+b，其值为 3（真），然后才根据关系和逻辑运算符的优先级进行运算，即分别运算 3>c 和 b<=c，它们的值非别为 0（假）和 1（真），最后将 0 和 1 相与（&&），得出最终结果为 0（假）。

算式二：a>=b+c&&b==c，在程序中亦可写成(a>=(b+c))&&(b==c)。首先得出 b+c 的值为 5（真），再分别计算出 a>=5 和 b==c，值分别为 0（假）和 0（假），将它们相或后，可得输出为 0（假）。

算式三：!(a<c)+b!=1&&(a+c)/2，可写成((!(a<c)+b)!=1)&&((a+c)/2)。由于嵌套括号的计算顺序是由里向外，所以算式的计算顺序可表示如下：

((!(a<c)+b)!=1)&&((a+c)/2) → ((!1+b)!=1)&&(4/2) → ((0+b)!=1)&&(4/2)

→ 1&&2 → 1

输出结果为 1。

# 实例 4 自增自减

## 实例说明

C 语言中有两个很有用的运算符，通常在其它计算机语言中是找不到它们的，那就是自增和自减运算符：++和--。自增运算符++对操作数增加一个单位，而自减运算符--对操作数减小一个单位。

本例正是要向读者介绍这两个运算符，使读者能够熟悉这两个运算符的一般用法。此外，自增和自减运算符是既能放在操作数之前，也能放在操作数之后的，但在具体的运算当中，这两种方法是有区别的，这也是本例所要讲述的重点，希望读者能够注意。

## 知识要点

在表达式当中，自增和自减运算符在操作数前或后是有区别的。增/减运算符位于操作数之前时，C 先实施增/减操作，然后才使用操作数的值；若增/减运算符是在操作数的后边，那么，C 是先使用操作数的值，而后再相应增/减操作数的内容。

注意点：

（1）自增运算符（++）和自减运算符（--），只能用于变量，而不能用于常量或表达式。

（2）++和--的结合方向是“自右向左”。

## 程序源码

该应用程序的源代码如下：

```
# include <stdio.h>

void main()
{
    int i, j, k;
    int m, n, p;

    i = 8;
    j = 10;
    k = 12;

    // 自增在操作数之前
    m = ++i;
```

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

     // 自减在操作数之后
     n = j--;
     printf("j = %d\n", j);
     printf("n = %d\n", n);

     // 自增、自减的混合运算
     p = (++m)*(n++)+(--k);
     printf("k = %d\n", k);
     printf("p = %d\n", p);
}
```

## 程序分析

在算式 m = ++i 中，对整形变量 i 进行了自增运算。由于自增运算符是置于 i 之前，所以是先对 i 进行加 1 操作，此时 i 的值已不再是 8，而是 9，然后再将自增后的 i 赋给变量 m，所以得到的输出为 9。

算式 n = j--是对变量 j 进行的自减操作，自减运算符位于操作数 j 之后，因此，赋给变量 n 的值就是 j 的原值 10，变量 n 的输出为 10，然后才进行自减操作，这时 j 的值减 1 变为 9。

最后进行的运算是同时包含自增和自减的混合运算。对于操作数 m 和 k 而言，自增和自减运算符位于它们之前，所以它们在算式中的值是经过自加和自减的，而对于变量 n 而言，自增运算符是位于它之后，因此它在算式中是以原值进行计算的。此时，我们再分析算式 p = (++m)*(n++)+(--k)，它实际上可以写成 p=10*10+11，所以变量 p 的输出结果为 111。