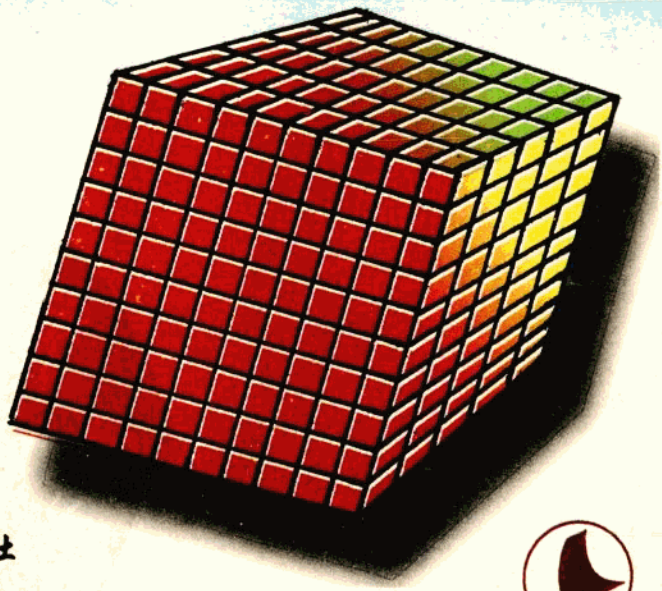
计算机实践与提高丛书之二

最新C高级实用编程技术

●白 明/编著



吉林大学出版社

前　言

C语言是当今世界广为流行的一种新型的通用程序设计语言，其功能完善，反映了当前计算机的能力，被广泛认为是一种高效、实用、灵活的软件开发语言，在短短数年内便赢得了广大计算机开发人员的广泛注目。

C语言是一种理想的结构化语言。它既继承了高级语言的优点，又兼顾了低级语言的许多特点；它既适合编写紧凑、高效的系统程序，又适合编写各类风格的应用程序。因此，计算机程序人员、各类非专业的程序人员、以及大学生、研究生都应该学会使用C语言。

为了帮助读者尽快学会C语言，更好地掌握C语言的编程风格和编制技巧，本书从C语言的基本知识入手，介绍了C语言程序的基本编制方法和调试操作；为了进一步掌握有关操作系统、编译、数据结构、系统资源利用、图形绘制及菜单设计等方面的程序设计技术，本书提供了大量的相关算法和应用程序示例；并在此基础上介绍了有关人工智能等方面的高级C语言程序编制技术和应用举例以及C语言的超集C++和如何使用Borland C++编制Windows应用程序的基本方法。本书中给出的所有程序实例都已在PC类微机上调试通过，调试环境为Turbo C2.0或Borland C++。

全书分为两部分共十四章。

第一部分为快速走进C语言。它从C语言的特点、基本概念入手，介绍了C语言丰富的运算符和优先顺序，讲述了结构化程序设计之顺序结构、选择结构和循环结构，以及函数构成和预处理，讲述了有关数组、结构体、共用体程序的编制方法，并较详细地阐述了在各种应用场合中指针的使用方法，还介绍了命令行参数和文件处理方法等。

第二部分为高级实用编程技术。它主要是在第一部分的基础上，介绍重要的通用算法及应用，包括在数据结构、编译原理和操作系统中的若干常用算法以及一些有关图形绘制、界面设计、系统资源利用等实用函数。

除此之外，第二部分还兼顾到已具有一定基础的读者所要求学习的高级编程技术，介绍了一些较高级的C编程技术，包括图像处理等人工智能求解技术问题、Borland C++和Windows 应用程序基础。

本书的所有源程序均存在一张软盘中，该软盘附于书后。

本书是从实用的角度讲述C语言的，由于计算机科学技术发展迅速，加之编者水平所限，疏漏与不足之处难免，诚恳希望专家同行和广大读者批评指正。

编者

1995年5月

目　　录

第一部分　快速走进C语言

第二部分　实用编程技术

第一部分
快速走进C语言

本书的第一部分共六章，以提纲和总结的形式全面论述了程序设计语言C的最基本内容。

第一章讲述C语言程序结构，基本的运行操作。

第二章讨论基本数据和运算。

第三章讲述结构化程序设计，包括函数和编译预处理等。

第四章讨论构造数据类型，包括数组、结构体、共用体以及位段。

第五章讲述指针。

第六章讲述文件操作和系统函数。

第一章　C语言程序设计概述

1.1　C语言程序的基本结构

1. C程序的函数结构

```
编译预处理语句(如:#include <stdio.h>)    /* 简单的C程序的结构 */
main()
{
 函数声明        /* 见说明(3) */
 变量定义
 语句
}

类型 fun1(形式参数)
  函数声明          /* 见说明(3) */
  变量定义
  语句
  return(返值)    /* 当无返值的情况下缺省 */
}
  ⋮
类型 funn(形式参数)
  函数声明          /* 见说明(3) */
  变量定义
  语句
  return(返值)    /* 当无返值的情况下缺省 */
}
```

说明:

(1) 任何函数间均为并行结构关系;

(2) 函数的类型为函数返值的类型,当函数无返值时,其类型应规定为 void 类型;

(3) 主调函数应对被调函数进行说明,当主调函数与被调函数符合向前引用关系或被调函数的返值为 int 型(或 char 型)时,声明可以省略;

(4) return 语句只能返回一个值;

(5) 形式参数应体现参数的个数、类型、顺序;当无形式参数时,应在形式参数位置填充 void。

2. 几种分支结构

```
if(a==b)     /* 不平衡结构的简单 if 语句 */
```

语句

```
if(a<b)        /*不平衡结构的复合 if 语句*/
  {
   语句 1
   ⋮
   语句 n
  }
if(x! =y)       /*if...else 结构*/
   语句 1
else
   语句 2

if(a>0)        /*嵌套 if...else*/
  if(b>0)     /*内嵌的 if...else*/
    语句 1
  else       /*可缺省,当缺省时,将改变配对关系。即下面的 else 与内嵌的 if 配对*/
    语句 2
else
  if(b! =a)              /*内嵌的 if...else,可全部缺省或缺省 else 部分*/
    语句 3
  else
    语句 4

switch(i)    /*开关分支,i 为整型或字符量*/
{
  case 情况 1:
    语句组 1
    break;
  case 情况 2:
    语句组 2
    break;
  ......
  case 情况 n:
    语句组 n
    break;
  default:  /*可缺省*/
    语句组
  }
```

3. 几种循环结构

```
while(循环条件)    /*先判断,后执行。当循环条件成立时执行循环*/
   循环体          /*为单一语句,当为多个语句时,应使用{ }包围*/

do                 /*为单一语句,当为多个语句时,应使用{ }包围*/

while(循环条件)    /*先执行,后判断。当循环条件成立时再次执行循环*/

for(e1;e2;e3)
   循环体          /*为单一语句,当为多个语句时,应使用{ }包围*/
```

说明:

(1) e1 是初值表达式,允许为逗号表达式,如:i=0,sum=0;

(2) e2 是条件表达式,先判断,后执行;

(3) e3 是修正表达式,允许为逗号表达式,如:i++,j--。

1.2 C 语言的语句

高级语言源程序的基本组成单位是语句。程序语句可分为非执行语句(如:变量的定义语句、函数的声明语句、注释等)与执行语句。执行语句又分为操作运算语句(用于描述计算机所执行的操作)和流程控制语句(控制操作运算语句的执行顺序)。

1. 表达式语句

C 语言是一种表达式语言。操作运算类语句就由表达式语句组成。由表达式后接一个分号构成表达式语句。表达式语句通常分为以下几种形式:

(1) 赋值语句(赋值表达式加分号)。例如:

```
x=0;
l=len(str);
```

(2) 函数调用语句(函数调用表达式加分号)。例如:

```
scanf("%d%f%c",&da,&fa,&ca);
```

(3) 空语句(只有一个分号)。例如:

```
;          /*形式语句,不产生任何操作*/
```

(4) 复合表达式语句。用花括号包围的一组语句。例如:

```
{
  i=1;
  sum=0;
  c=getchar();
}
```

2. 控制语句

(1) 条件分支语句。例如:

```
if(x>y)
```

```
    a=x;
  else
    a=y;
```

(2) 开关分支语句。例如：

```
switch(i)
{
  case  1: printf("A");break;
  case  2: printf("B");break;
  case  3: printf("C");break;
  default: printf("D");
}
```

(3) 循环语句。例如：

```
for(i=1,sum=0;i<10;i++)
  sum=sum+i;
```

(4) exit 函数。它是一个标准的库函数。它的作用是立即停止当前正在执行的程序，并回到操作系统状态。调用此函数需用一个 int 型实参，当使用 0 作为参数时，说明属正常停止；而使用其它数值作为参数时，则表示错误停止，该数值通常表示造成停止的错误类型。

3. 辅助控制语句

(1) break 语句：

break 语句只能用在两种语句结构中。一种是 switch 分支结构，用于将控制从 switch 语句中转出；另一种是循环结构，用于将控制从包含该 break 语句的循环层中转出，即结束本层循环。例如：

```
while(...)
{ ...
   break;
  ...
}
语句 A
```

该循环中的 break 语句使控制从此语句处转到语句 A 处继续执行。

(2) continue 语句：

continue 语句只能出现在循环语句的循环体中。它的功能是使本次循环的执行提前结束，即不再执行 continue 下面本层循环的其它语句，控制转到重新判断循环条件处，再根据循环条件是否满足决定是否进入下次循环。例如：

```
while(...)
{ ...
   continue;
  ...
  循环体末语句
}
```

该循环中的 continue 语句使控制从此语句处转到 while 语句(重新判断循环条件)处继续执

行。

(3) goto 语句：

一般形式：

goto 标号

goto 语句是无条件转移语句，即将控制强行转到由标号指定的语句处。多用 goto 语句是有害无益的，但在某些情况下还有其特殊的功效。goto 语句限制在以下情况中使用是比较安全的：

● 限定在包含该 goto 语句的最小范围内转移(一个复合语句内部或一个函数内部)。

● 置于深层循环中，使控制从多重循环中转出。这是 goto 语句最常使用的情况。它可以提高程序的效率，而此控制功效是非其它语句所能替代的。

(4) return 语句：

return 语句的功能是使控制流程从被调函数返回主调函数。在返回的同时可以通过 return(返回值)带回一个函数值到主调函数，而当无需带回值到主调函数时函数中可以缺省 return 语句。

1.3 C 语言程序的编辑、编译、连接与运行

C 语言以编译方式将源程序转换为二进制目标程序代码，要执行一个 C 程序，一般需经以下步骤。

(1)利用编辑程序编写 C 语言源程序，如：Turbo c 提供的 TC.EXE 或 MS—DOS 下的 EDLIN.EXE 等。

(2)程序编译，主要完成程序的语法检查，编译通过则产生后缀为.OBJ 的目标文件。

(3)程序连接，将编译好的二进制目标代码与系统标准模块进行连接的处理。连接后生成后缀为.EXE 的执行文件。

(4)程序执行，在操作系统状态下，对经编译、连接后的具有.EXE 扩展名的执行文件的运行过程。

C 允许将一个程序分解为若干块，装入若干文件，并进行独立编译，将它们用一种扩展名为.PRJ 的文件联系起来。

集成化的工具环境(例如：Turbo C 的集成开发环境)将编辑、编译、连接、调试工具集于一体，用户在窗口状态下可连续进行编辑、编译、连接、调试、运行的全过程。

1.4 C 语言程序的函数组装结构

C 语言本身语句较少，有许多功能都是通过函数来完成的。这是 C 语言的一个重要特色。它使 C 语言有更大的灵活性和多方面的功能。

1. 利用 C 库函数

C 语言提供了极为丰富的库函数，在编写 C 程序时应尽量利用 C 的库函数所提供的函数功能，以完成自己的工作。即首先采用 C 库函数来组装用户的 C 程序。但是，使用 C 库函数务必注意包含所使用函数必备的头文件。例如：在使用输入输出函数时必须包含名为"stdio.h"的头文件；在使用数学函数时必须包含名为"math.h"的头文件；在使用字符处理函数时必须

包含名为"string.h"的头文件。

2. 用户设计C函数

用C库函数组装程序是在C程序设计中的一条捷径和非常重要的方法。但是当在函数库中找不到所需功能的函数时，还必须自己动手设计所需的函数。

必须由用户设计的一个函数是main()主函数。main是函数名，函数名后面的一对括号不能缺省，根据需要，括号内可以有参数，也可以缺省参数。一个完整的程序必须有且只能有一个main函数，它被称为主函数。每一个C程序都是从main函数开始执行的。

用户所设计的所有函数都为并行关系，其函数定义按传统风格的一般形式为：

```
函数类型  函数名(函数形式参数)
形参类型定义
{ 数据定义部分
  函数执行部分
}
```

ANSI建议采用现代风格方式，即函数原形方式，它把函数的形参类型定义写在函数名后面的括号之内，以增加程序的安全性。

(1) 将传统风格的函数定义形式：

```
float sum(a,b,c)
float a,b,c;
```

改写成现代风格形式为：

```
float sum(float a,float b,float c);
```

(2) 将传统风格的函数声明形式：

```
float sum();
```

改写成现代风格形式为：

```
float sum(float a,float b,float c);
```

以下情况可以缺省函数声明：

(1) 调用是在定义的有效区域内，即在同文件中，定义在前，调用在后；

(2) 函数的返回值为int或char类型。

第二章　数据及其基本操作

2.1　常量、变量、标识符

2.1.1　常量

常量是指在程序执行期间其值始终保持不变的量。在C语言中，常量分为两大类共五种。一类是直接常量，即日常所说的常量；另一类是符号常量，即用一个符号来表示一个常量。

1. 整型常量

C语言允许使用十进制数、八进制数、十六进制数来表示一个整型常量。凡以0开头的数字序列，表示八进制数。如：0111为八进制数，等价于十进制数73；而以0X开头的数字序列，表示十六进制数。如：0XA3为十六进制数，等价于十进制数163。在C语言中，可以在整型数后加字母L(或l)表示长整型数。

2. 实型常量

实型常量只允许用十进制的形式表示，而且不分单精度型和双精度型。实型常量可以采用小数形式表示，也可以采用指数形式表示。如：3.14159，108.10，3e2，5e－3等均为合法的实型常量。

3. 字符常量

字符常量是用一对单撇号括起来的单一字符。如：'a'，'R'，'8'，'#'。注意，这里单撇号是字符常量的定界符，所以当单撇号本身作为字符常量时，应采用'\''的形式；当反斜线本身作为字符常量时，应采用'\\'的形式。

4. 字符串常量

字符串常量是一种有别于字符常量的常量。它采用双撇号作为定界符。即用一对双撇号括起来的零个或多个字符序列。如："C PROGRAM"，"a"，" "等。

字符串常量在计算机内存储时，系统自动在其末尾添加一个字符"\0"，作为结束标志，字符串中的字符数称为该字符串的长度。

注意，当双撇号本身作为字符串中的字符时，应采用\"的形式；当反斜线本身作为字符串中的字符时，应采用\\的形式。

字符串常量与字符常量不仅其表示形式不同，更重要的是存储上的区别。如：'A' 是一个字符常量，而"A" 是一个字符串常量。它们的存储形式分别为：

'A'——仅占一个字节，并以其ASCII码值65存储；

"A"——则占两个字节，第一个字节是以其ASCII码值65存储，而第二个字节是系统添加的结束标志\0。

5. 符号常量

(1) 反斜线字符常量：

在计算机中，有些控制字符是无法直接用字符常量的形式表示的，但在C中允许采用下

面几种形式表示控制字符：

●用反斜线开头后面跟一个字母，如：

\a 报警　　\b 退格　　\f 走纸换页　　\n 换行

\r 回车　　\t 水平制表　　\v 垂直制表

●用反斜线开头后面跟1到3个八进制数来代表 ASCII 码为该八进制数的字符。如：\010表示\b，\012表示\n。

●用\x 开头后面跟2个十六进制数来代表 ASCII 码为该十六进制数的字符。如：\x08表示\b，\x0A 表示\n。

(2) 用#define 定义符号常量：

C 语言提供了一种方法，常在程序的开头处，用#define 将一般常量用大写符号来表示，以区别于程序中的一般变量。此后的程序中即可用此符号来代替等价的常量了。如：

```
#define PI 3.1415926
```

表示 PI 是与3.1415926等价的符号常量。

2.1.2 变量

变量是指在程序执行期间其值可以改变的量。在程序中，数据连同其存储空间被抽象为变量。每个变量都有一个名字，称为变量名。它代表了某个存储空间及其所存储的数据。因此在引用变量之前必须首先定义变量的类型，编译程序则可根据所定义的类型分配一定的存储空间，并决定数据的存储方式和所允许的操作。

对变量定义的形式如下：

```
int     /* 定义 ia,ib 为整型变量 */
float   /* 定义 fa,fb 为实型单精度变量 */
char    /* 定义 ca,cb 为字符型变量 */
```

变量可以通过赋值获得数据，也可以通过初始化确定其值。把从运算器向变量所代表的存储单元传送数据的操作称为赋值。在 C 语言中用"="表示赋值操作。其一般形式为：

变量=表达式

以下是合法的表示形式：

```
a=5+9
x=a=b=c=4*6
```

C 语言允许在定义变量的同时对变量赋初值，，称之为变量的初始化。如：

```
int ia=1,ib=1;
float fa=3.5,fb;
char ca,cb='x';
```

2.1.3 标识符

标识符是对变量、函数、标号和其它各种用户定义对象的命名。C 语言规定，标识符的第一个字符必须是字母或下划线，其随后的字符则必须是字母、数字或下划线。C 语言中，字母的大小写是敏感的，即字母的大小写是不同的字符，代表不同的意义。如：ABC，Abc，abc 是三个不同的标识符。注意，标识符不能与 C 语言的关键字相同，也不能与用户自定义的函数及 C 语言库函数同名。

关键字也称保留字，是C编译程序预先登录的标识符，共计28个，均由小写字母构成，它们代表了固定的意义。它们是：

char int float double short long struct union auto register static

unsigend typedef extern goto return break continue (asm) entry

sizeof if else for do while switch case default (fortran)

系统除了定义的关键字外，还指定了一些具有特定含义的标识符，被称为特定字。主要用于预处理程序中。它们是：

define undef ifdef ifndef endif line include

2.2 基本数据类型及其转换

2.2.1 基本数据类型

C语言有以下几种基本数据类型：

类型标识	类 型	字节数	数的范围	无符号数范围
char	字符型	1字节	−128～127	0～255
short	短整型	2字节	−32768～32767	0～65535
int	整型	机器字长(16位或32位)		
long	长整型	4字节	−2147483648～2147483647	0～4294967295
float	单精度浮点型	4字节	±1.7E±38	无
double	双精度浮点型	8字节	±3.4E±308	无

前面所指出的数据类型都是指有符号的数据类型，此外，C 语言还允许使用无符号整数，它可使整数最大值的范围扩大一倍，适用于表示只有正值的数据。在使用无符号数据类型时，必须在原类型标识符的前面加保留字 unsigend，而有符号整数类型的标识符前的 signed 可以缺省。

在C语言中，字符数据(char)也分为 signed 和 unsigned。字符数据和整型数据兼容，即可以把字符型数据当做一字节整型数据使用，字符型数据以其 ASCII 码值参与运算。

2.2.2 类型转换

C语言允许数据值从一种类型转换成另一种类型。数据的转换主要分为隐式转换和显式转换两大类，共有以下几种形式。

1. 必然转换(隐式转换)

这种转换是将短的数据扩展成机器处理的长度，即使运算符两端运算量的类型相同。转换原则是将表达式中的 char 或 short 全部自动转换为相应的 int 型；将 float 全部自动转换为 double 型等，具体情况如下表所示：

运算量类型	运算数据类型
char short	int
unsigned char unsigned short	unsigned int
float	double
long	不变
unsigned long	不变

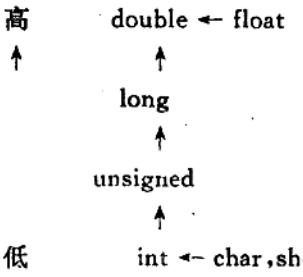
2. 运算转换(隐式转换)

当一个运算符的两端的运算量类型不同时,必须使运算符两端具有共同的类型。转换原则是"向高看齐,一次到位"。

(1) 不同长的整型或字符型数据,向其中最长的整型转换;

(2) 整型或字符型与实型数据混合,向 double 型转换;

3. 数据转换图示



其中由右至左为必然转换,由下至上为运算转换。

4. 赋值转换(隐式转换)

赋值转换具有强制性,即对赋值运算符右边表达式的值强制转换为赋值运算符左边变量的相同类型。因此,当类型不同时可能会产生误差(或错误)。如:

```
int    a=0.93;
```

结果则为0。

5. 强制类型转换(显式转换)

C 语言提供了强制类型转换的运算,以使数据转换为人们所期望的任何已存在的数据类型。

强制类型转换运算是一种单目运算,它的一般形式为:

(类型标识符)表达式

其中(类型标识符)是运算符,运算功能是将该运算符右边表达式的值强制转换为括号中类型表识符所指定的数据类型。如:

```
(int)((float)i+j)        /*得到一个整型数*/
(float)(x=1)             /*得到一个单精度实型数*/
(char)(3.14*i+6)         /*得到一个字符型数*/
```

注意:(int)5.5+8.2只对5.5转换,而不对8.2转换,表达式值为13.2的实型数。

6. 输出转换(显式转换)

在使用 printf 函数输出时，可以通过指定显式的格式，使输出的数据按指定的类型输出。例如：一个 long 型数在 printf 函数中指定用%i(或%d)格式输出，处理过程是将 long 型数转换为 int 型数据后再输出。需要提醒注意的是：

(1)较长类型转换为较短类型，当数据超出目标类型的范围时，将得到不可思议的结果；

(2)有符号数转换为无符号数，符号位将作为数值的一部分处理，而不再作为符号。因此，负数将导致错误；

(3)实型数转换为整型数，小数的截断将导致误差，当实数值超过整数类型的最大值范围时将产生错误的结果。

在众多的数据类型转换中，其转换都是临时性的。即源数据的数据类型并未改变，转换仅仅是为了满足运算需要，转换只对结果(目的数据)产生影响。

2.3 基本运算

C 语言的运算种类丰富，与此相应，它的运算符种类繁多。讨论各种不同运算组成一个运算表达式时，应考虑以下四个问题。

1. 运算符的功能

包括运算符要求的运算量的数目、运算量的数据类型。如：表达式 a%b，运算符%为求模运算(即表达式的值为 a 除以 b 的余数)，同时要求两个运算量 a 和 b 均为整型数据。下面给出基本运算及功能一览表。

(1) 算术运算：

运算类别	运算量数目	运算符	名称	例子	运算功能
算术加	2	+	加	a+b	求 a 与 b 的和
算术减	2	−	减	a−b	求 a 与 b 的差
算术乘	2	*	乘	a*b	求 a 与 b 的乘积
算术除	2	/	除	a/b	求 a 除以 b 的商
算术求模	2	%	模	a%b	求 a 除以 b 的余数
自反加赋值	2	+=	加赋值	a+=b	a=a+b
自反减赋值	2	−=	减赋值	a−=b	a=a−b
自反乘赋值	2	*=	乘赋值	a*=b	a=a*b
自反除赋值	2	/=	除赋值	a/=b	a=a/b
算术自加	1	++	自加	a++(或++a)	a=a+1
算术自减	1	−−	自减	a−−(或−−a)	a=a−1
符号运算	1	+(或−)	符号	+a(或−a)	0+a(或0−a)

(2) 关系运算：

运算量数目	运算符	名称	例子	结　果
2	>	大于	m>n	T:当 m 大于 n 时,F:否则
2	>=	大于等于	m>=n	T:当 m 大于或等于 n 时,F:否则
2	<	小于	m<n	T:当 m 小于 n 时,F:否则
2	<=	小于等于	m<=n	T:当 m 小于或等于 n 时,F:否则
2	=	等于	m=n	T:当 m 等于 n 时,F:否则
2	!=	不等于	m!=n	T:当 m 不等于 n 时,F:否则

(3) 逻辑运算：

运算量数目	运算符	名称	例子	结　果
1	!	逻辑非	!a	T:当 a 为 F,F:否则
2	&&	逻辑与	a&&b	T:当 a,b 都为 T,F:否则
2	||	逻辑或	a||b	F:当 a,b 都为 F,T:否则

(4)位运算：

运算量数目	运算符	名称	例子	运算功能
1	~	按位取反	~a	对变量 a 按位取反
2	&	按位与	a&b	变量 a 和 b 按位进行与运算
2	|	按位或	a|b	变量 a 和 b 按位进行或运算
2	^	按位异或	a^b	变量 a 和 b 按位进行异或运算
2	<<	左移	a<<2	变量 a 的各位全部左移2位
2	>>	右移	a>>2	变量 a 的各位全部右移2位
2	&=	位与赋值	a&=b	a=a&b
2	|=	位或赋值	a|=b	a=a|b
2	^=	位异或赋值	a^=b	a=a^b
2	<<=	左移赋值	a<<=2	a=a<<2
2	>>=	右移赋值	a>>=2	a=a>>2