

位运算

1 引言

2 基本位运算

3 位段

4 应用实例

引言

❖ C语言既是一种高级语言，广泛应用于应用软件的开发和程序设计，同时又是一种低级语言，可以用于系统软件的开发和程序设计，如自动控制系统中的过程控制、参数检测、数据通讯等控制程序，都可以综合利用C语言中的指针操作、位运算和位段技术来实现。

❖ 本章介绍位运算的基本形式和常用运算符,并简要介绍位段的概念。位运算的深入学习，应该在《计算机原理》和《汇编语言程序设计》课程中进行。

基本位运算

❖ 位运算概述

- 所谓“位运算”，是指按二进制位进行运算。

运算符	含义	示例	说明
&	按位与	b&c	求b和c的位与
	按位或	b c	求b和c的位或
^	按位异或	b^c	求b和c的位异或
~	取反	~b	求b的位反
<<	左移	c<<2	c左移2位
>>	右移	b>>3	b右移3位

注：参与运算的数是以补码形式出现的，并且参与运算的量应为整型或字符型，
不能为实型数据

按位与 (均为1时结果为1)

```
#include<stdio.h>
int main( )
{
    int a,b;
    printf("Enter a :");
    scanf("%d",&a);
    printf("Enter b:");
    scanf("%d",&b);
    printf("a&b=%d\n",a&b);
    return 0;
}
```

$$\begin{array}{r} 0001\ 0100\ (a) \\ \&\ 0001\ 1110\ (b) \\ \hline 0001\ 0100 \end{array}$$
$$\begin{array}{r} 0000\ 1010\ (a) \\ \&\ 0001\ 0100\ (b) \\ \hline 0000\ 0000 \end{array}$$

Enter a and b: 20,30
a&b=20

Enter a and b: 10,20
a&b=0

按位或 (均为0时结果为0)

```
#include<stdio.h>
int main( )
{
    int a,b;
    printf("Enter a :");
    scanf("%d",&a);
    printf("Enter b:");
    scanf("%d",&b);
    printf("a | b=%d\n",a | b);
    return 0;
}
```

$$\begin{array}{r} 0001\ 0100\ (a) \\ | \ 0001\ 1110\ (b) \\ \hline 0001\ 1110 \end{array}$$
$$\begin{array}{r} 0000\ 1011\ (a) \\ | \ 0001\ 0100\ (b) \\ \hline 0001\ 1111 \end{array}$$

Enter a and b: 20,30
a | b=30

Enter a and b: 11,20
a & b=31

按位异或 (相等时为0, 不等时为1)

```
#include <stdio.h>
int main()
{
    unsigned char a=0x39, b=0x0F;
    a=a^b;
    printf("%#x\n", a);
    return 0;
}
```

$$\begin{array}{r} 0011\ 1001\ (a) \\ \wedge\ 0000\ 1111\ (b) \\ \hline 0011\ 0110 \end{array}$$

0x36

按位取反

```
#include <stdio.h>
int main( )
{
    char a=3;
    int b=10;
    printf("~a=%d,~b=%d\n",~a,~b);
    return 0;
}
```

~ a:

补码: 11111100

原码: 10000100

~ b:

补码: 11110101

原码: 10001011

~a=-4, ~b=-11

左移 $a \ll n$ 将a中各位向左移n位，右端补0，高位溢出丢弃。

```
#include <stdio.h>
int main()
{
    unsigned int a=0x3ef,b;
    b=a<<2;
    printf("%x,%x\n",a,b);
    return 0;
}
```

3ef,fbc

右移运算

$a \gg n$ 将a中各位右移n位，溢出则舍弃。

左端补位

a为正数时（符号位为0），填0；

a为负数时（符号位为1），填0或填1与系统有关

填0（逻辑右移）

填1（算术右移） TC使用算术右移

右移

```
#include <stdio.h>
int main()
{
    int a=9,b=-9;
    printf("%d,%d",a>>2,b>>2);
}
```

结果： 2, -3

-9的补码： 1111111111110111

右移后为： 1111111111111101

位段

- ❖ C语言允许在一个结构体中以位为单位来指定其成员所占内存长度。这种以位为单位的成员称为“位段”。

```
int main()
{
    struct packed_date{
        unsigned int a:2;
        unsigned int b:6;
        unsigned int c:4;
        unsigned int d:4;
        int i;
    }data;
    data.a=3;data.b=4;
    data.c=data.a+data.b;
    data.d=data.b-data.a;
    printf("data.c=%d,data.d=%d\n",data.c,data.d);
    data.i=data.a+data.b+data.c+data.d;
    printf("%d\n",data.i);
}
```

- ❖ 位段的类型只能是int, unsigned int, signed int三种类型, 不能是char型或者浮点型;
- ❖ 位段占的二进制位数不能超过该基本类型所能表示的最大位数, 比如在VC中int是占4个字节, 那么最多只能是32位;
- ❖ 无名位段不能被访问, 但是会占据空间;
- ❖ 不能对位段进行取地址操作;
- ❖ 若位段占的二进制位数为0, 则这个位段必须是无名位段, 下一个位段从下一个位段存储单元(VC环境下是4个字节)开始存放;
- ❖ 若位段出现在表达式中, 则会自动进行整型升级, 自动转换为int型或者unsigned int。
- ❖ 对位段赋值时, 最好不要超过位段所能表示的最大范围, 否则可能会造成意想不到的结果。
- ❖ 位段不能出现数组的形式。

应用实例

从整数a最右端第m个位置开始取该位开始右面n位

算法如下：

$$b = a \gg (m - n + 1)$$

$$c = \sim (\sim 0 < \sim n)$$

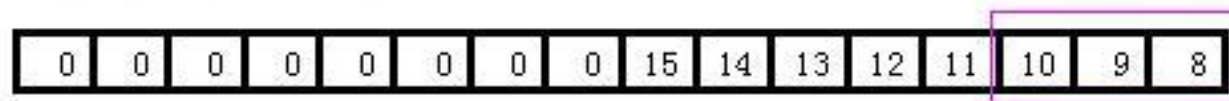
d=b&c

注：位自右向左从0
开始编号

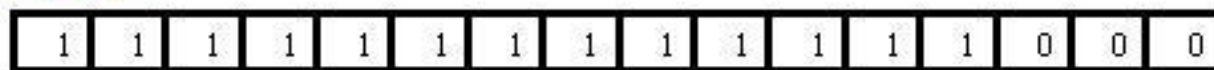
a $m=10$ $n=3$



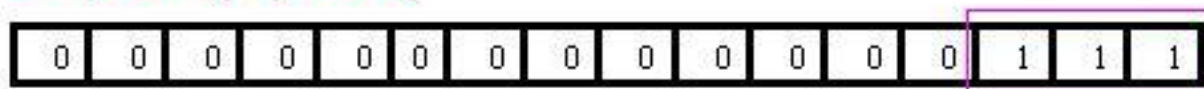
$$b = a \gg (10 - 3 + 1) = a \gg 8$$



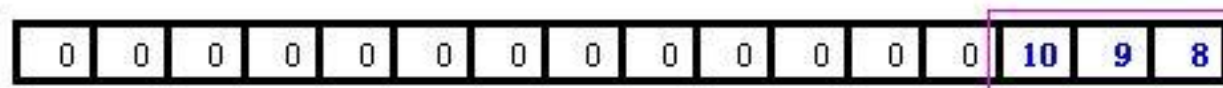
$-0 \leq \leq 3$



$$c = \sim(\sim 0 \leq n) = \sim(\sim 0 \leq 3)$$



d=b&c



将一个整数a循环右移n位。

算法如下：

$b = a \ll (16 - n)$

$c = a \gg n$

$c = c | b$

a n=5

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---

$b = a \ll (16 - n)$

5	4	3	2	1	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

$c = a \gg n$

0	0	0	0	0	16	15	14	13	12	11	10	9	8	7	6
---	---	---	---	---	----	----	----	----	----	----	----	---	---	---	---

$c = c | b$

5	4	3	2	1	16	15	14	13	12	11	10	9	8	7	6
---	---	---	---	---	----	----	----	----	----	----	----	---	---	---	---