位与数据

C Reference

位、二进制与位运算符

位 (bit) 是单个的 0 或 1; 数字以二进制 (binary) 存储。按位 (bitwise) 运算符直接作用于这些位:`&` 与、`|` 或、`^` 异或、`<<` 左移 (每次 ×2)、`>>` 右移 (÷2)。

```c
#include <stdio.h>

int main(void) {
    int a = 12;               // 1100
    int b = 10;               // 1010
    printf("%d\n", a & b);    // 8   与   -> 1000
    printf("%d\n", a | b);    // 14  或   -> 1110
    printf("%d\n", a ^ b);    // 6   异或 -> 0110
    printf("%d\n", a << 1);   // 24  左移
    printf("%d\n", a >> 1);   // 6   右移
    return 0;
}
```

十六进制 (hexadecimal,base 16) 是二进制的紧凑写法: 一个十六进制位恰好等于四个二进制位。字面量用 `0x` 开头; 打印用 `%x`:

```c
#include <stdio.h>

int main(void) {
    printf("%d\n", 0xFF);   // 255  (0x 表示十六进制字面量)
    printf("%x\n", 255);    // ff
    printf("%x\n", 12);     // c
    return 0;
}
```



Exam tip: bitwise ops work on bits of integers; << multiplies by 2, >> divides by 2

Bitwise operators work on the bits of integers

游程编码

游程编码 (run-length encoding,RLE) 是一种简单的压缩 (compression) 方法: 把每一段重复字符的游程 (run) 替换成一个计数加上那个字符。它是无损 (lossless) 的 — 原始数据可以完全还原。

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

int main(void) {
    char *s = "aaabbc";
    int n = strlen(s);
    for (int i = 0; i < n; ) {
        char c = s[i];
        int run = 0;
        while (i < n && s[i] == c) { run++; i++; }
        printf("%d%c", run, c);      // 3a2b1c
    }
    printf("\n");
    return 0;
}
```

常见错误

- `&` 是按位与,`&&` 是逻辑与 — 别混淆。
- 左移 `<< 1` 让值翻倍; 右移 `>> 1` 让值减半。
- 第 `n` 位的值是 `1 << n`(C 没有 `**` 幂运算符); 用 `(x >> n) & 1` 检查某一位。