位运算

1. 与、或、异或和取反

&

|

^

~

2. 移位运算

左移 << 算术移位和逻辑移位是相同的

高位丢弃，低位补0

右移 >> 算术移位 高位补符号位，低位舍弃

逻辑移位 高位补0，低位舍弃

算术移位 带符号

逻辑移位 不带符号

3. 位运算的性质

取反运算性质：−a=∼(a−1)

a & (a−1)  将 a的二进制表示的最后一个 1变成 0；

a & (-a) ⇔ a & (~(a-1)) 只保留二进制最后一个1，其余的1变成0

例题：

1. 十进制转七进制

class Solution {

public:

string convertToBase7(int num) {

if (num == 0) return "0"; // 特殊情况处理，0的七进制表示是"0"

bool isNegative = num < 0; // 判断是否为负数

num = abs(num); // 转换为正数进行处理

string s = "";

while (num > 0) {

s = to\_string(num % 7) + s; // 计算当前位的数字并加到字符串前面

num /= 7; // 去掉最后一位

}

if (isNegative) s = "-" + s; // 如果是负数，添加负号

return s;

}

};

1. 十进制转十六进制（补码方式）

class Solution {

public:

string toHex(int num) {

if (num == 0) return "0"; // 特殊情况处理，0的十六进制表示是"0"

string hex\_chars = "0123456789abcdef"; // 十六进制字符集

string s = "";

// 补码表示，负数会以32位补码形式表示

unsigned int n = num; // 将整数转换为无符号数以处理负数

while (n > 0) {

s = hex\_chars[n & 0xF] + s; // 提取最低4位，并拼接到字符串前面

n >>= 4; // 右移4位，处理下一位

}

return s;

}

};

1. 给定一个正整数 n，编写一个函数，获取一个正整数的二进制形式，并返回其二进制表达式中设置位的个数。

输入：n = 11

输出：3

解释：输入的二进制串 1011 中，共有 3 个设置位。

class Solution {

public:

    int hammingWeight(int n) {

        int count=0;

        while(n!=0){

            n=n&(n-1); //利用位运算的性质

            count++;

        }

        return count;

    }

};

4. 颠倒给定的 32 位无符号整数的二进制位。

class Solution {

public:

    uint32\_t reverseBits(uint32\_t n) {

        uint32\_t num=0;

        int x=31;

        while(n!=0){

            num+=(n&1)<<x;

            n>>=1;

            x--;

        }

        return num;

    }

};

5. 两整数之和（不用+）

加法的结果分成两个部分，分别是进位部分和非进位部分。

进位部分：只有当两个数都是1时，才会发生进位，进位部分的值是10，因此进位部分可以看成是两个数的按位与的结果左移一位。

非进位部分：当两个数相同时非进位部分的值是0，当两个数不同时非进位部分的值是1，因此非进位部分可以看成是两个数的按位异或的结果。

class Solution {

public:

    int getSum(int a, int b) {

        while(a!=0){

            int temp=a^b;

            a=(a&b)<<1;

            b=temp;

        }

        return b;

    }

};

6. 格雷编码

n 位格雷码序列 是一个由 2n 个整数组成的序列，其中：

每个整数都在范围 [0, 2n - 1] 内（含 0 和 2n - 1）

第一个整数是 0

一个整数在序列中出现 不超过一次

每对 相邻 整数的二进制表示 恰好一位不同 ，且

第一个 和 最后一个 整数的二进制表示 恰好一位不同

给你一个整数 n ，返回任一有效的 n 位格雷码序列 。