# 题目

不使用运算符+和-，计算两整数a 、b之和。

**示例 1:**

输入: a = 1, b = 2

输出: 3

**示例 2:**

输入: a = -2, b = 3

输出: 1

# 分析

## 方法一：位操作

**思路：**

二进制的加法无外乎就以下几种情况，

1+1 = 0 （有进位）

1+0 = 1 （无进位）

0+0 = 0 （无进位）

0+1 = 1 （无进位）

仔细一看，在不考虑进位的情况下，就是二进制的异或操作。所以，我们就可以把加法分成无进位的异或结果，a^b与保存的进位相加，循环直到没有进位为止，就可以得到结果了。

再分析二进制加法中进位怎么能保存，因为只有1+1的时候会产生进位，这就是与操作，a&b但是进位需要在更高的一位，所以我们就左移一位就搞定了，所以，进位就可以这样保存

c = (a&b)<<1;

这样就可以用a = (a^b) ^ c 得到一轮结果，但是有可能还会有进位，所以需要将这个放在循环里面就可以了。

**代码：**

class Solution {

public:

int getSum(int a, int b) {

while(b)

{

auto carry = ((unsigned int)(a&b))<<1;

// 记录a+b的进位，直到进位为0是退出

a = a^b; //结果相加

b = carry; //循环

}

return a;

}

};