剑指65 不用加减乘除做加法

知识盲区

等于模拟加法器的原理——背题
异或(^)：求和的低位。0^1 = 1， 0^0 = 0，1^1 = 0
与(&)：求和的进位。都是1，才进一位，即高位的值为1。

本题考察对位运算的灵活使用，即使用位运算实现加法。 设两数字的二进制形式 a, b ，其求和 $s = a + b$ ， $a(i)$ 代表 a 的二进制第 i 位，则分为以下四种情况：

$a(i)$	$b(i)$	无进位和 $n(i)$	进位 $c(i+1)$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

观察发现，**无进位和** 与 **异或运算** 规律相同，**进位** 和 **与运算** 规律相同（并需左移一位）。因此，无进位和 n 与进位 c 的计算公式如下；

$$\begin{cases} n = a \oplus b & \text{非进位和：异或运算} \\ c = a\&b << 1 & \text{进位：与运算 + 左移一位} \end{cases}$$

（和 s ） = （非进位和 n ） + （进位 c ）。即可将 $s = a + b$ 转化为：

$$s = a + b \Rightarrow s = n + c$$

这样加法问题又变成了下一个加法问题，套娃加法，但是在c不断与和左移的过程中，c的0会越来越多，最后c变为全0，问题结束。

解答：
1、公式是 s = n + c，但是题目不能用加号，所以只能循环套娃直到进位为0，这样就不需要加号了，此时s = n。

相当于b的值在每次循环的过程中分了一部分给a。 2、为什么可以循环运算，因为求进位的时候左移了1位，低位补了0，所以最终结果进位一定会左移到0这个值，如果进位不能左移到0，那这题用这个思路就做不出来了。
精华：2个数的加法，转化为另外2个数的加法，不断这样，最终变为一个数+0.

```
class Solution {
public:
    int add(int a, int b) {
        int sum = a^b; //sum中每一位代表a,b每一位的异或。
        // a&b每一位表示与运算，左移让其代表低位加法后的进位值。
        int carry = (a&b)<<1;
        while(carry != 0) {
            a=sum;
            b=carry;
            sum=a^b;
            carry=(a&b)<<1;
        }
        return sum;
    }
};
```