[整型数据求平均数方法的探讨](http://blog.csdn.net/anchor89/article/details/5978227)

今天在网上看到一个很有意思的题目,求两个int类型数字的平均值,并作为int类型返回.

下面这种代码是显然错误的

**[cpp]** [view plaincopy](http://blog.csdn.net/anchor89/article/details/5978227)

1. **int** average(**int** a, **int** b)
2. {
3. **return** (a+b)/2;
4. }

道理也很显然,这个代码没有考虑到溢出的情况,如果a,b太大或太小,就会返回一个错误的结果,比如当a=2147483647 b=1时,

这段代码会返回-1073741824而不是正确的结果1073741824.

另外一个容易犯的错误是这样(我荣幸的犯了这个错误...),为了保证不溢出,可以事先比较a,b的大小,使得a<b,然后用a加上两数之差的一半,可以得到如下改进的代码

**[c-sharp]** [view plaincopy](http://blog.csdn.net/anchor89/article/details/5978227)

1. **int** average2(**int** a, **int** b)
2. {
3. **if** (a > ｂ)
4. swap(a,b)
5. **return** a + ((b-a) >> 1);
6. }

因为b肯定是不小于a的,所以b-a不可能溢出(如果去掉前面的if比较话则不能保证这点了,这就是事先比较的必要性),同时两者的平均数显然也在表示范围内,故这个函数中不会出现溢出的情况.

错误之处在于当b为正数,a为负数时,b-a可能超出了int的表示范围,于是就hll的溢出了.

熟悉c/c++的同学会立马想到,在double类型中,底数的有效位数大于int,可以首先将int类型的数据强制转换成double类型求得平均数,然后再转换会int类型并返回.这种方法确实可以避免溢出的问题,但是int和double类型之间的强制转换是有时间开销的,在求平均数这样基本的运算上浪费这种时间太可惜了.如果碰到需要多次求平均数的程序,这种时间开销直接导致整个程序的时间增长,因此也不是很完美的方法.

从上面的方法来看,貌似为了完成求平均数这样简单的数学运算,让计算机来做还真有点为难!

可是给定的a,b都是int可表示的,所求的平均数也是int可表示的,冥冥之中总有一种感觉会有一个简单有效的方法来完成这个功能,不需要做罗哩罗嗦的条件判断,也不需要在更大的范围内做类型转换.

确实,有一个很简单优美的式子就能完成这个功能,请看下面的代码

**[cpp]** [view plaincopy](http://blog.csdn.net/anchor89/article/details/5978227)

1. **int** averagePerfect(**int** a, **int** b)
2. {
3. **return** (a&b) + ((a^b) >>　1);
4. }

这段代码中只涉及加法和位运算,因此具有很高的效率.但打眼看上去,除了一个右移能看出除2的影子来,实在看不出跟求平均数有什么关系.

下面就来证明一下这段代码的正确性,证明之后你会发现从计算机的角度来看这段代码是显然正确的.

设a的二进制表示为a[31]a[30]a[29]...a[0],类似的b的二进制表示为b[31]b[30]...b[0],结果用c变量来存,同样他的二进制表示为c[31]c[30]..c[0].

那么c=(a+b)>>1用二进制表示就是c[n]=((a[n+1]+b[n+1])的低位+(a[n]+b[n])的高位)=((a[n+1]^b[n+1]))+(a[n]&b[n]),换成a,b,c来看就是c=(a&b) + ((a^b)>>1)了,如果还不清楚,可以将n取的小一些,然后自己列列竖式就行了.

但是证明上述**return** (a&b) + ((a^b) >>　1); 是错的

a = 2147483648 b = 1

结果为-1073741824

a = 2147483648 b = 2

结果为-1073741823