

# 后端

## Java03

前言：各位学长好呀，在学习过程中我个人更喜欢试着理解之后再把自己理解的敲下来，基本没有复制粘贴，所以文本体量可能没有那么大，用语没有那么专业，也可能会犯错就是了~

### task1

#### Q1&Q2:

因为没啥好说的我就直接摘抄了(^\_^) (总会忘记一字节对应几个二进制位数)

int: 4字节, 范围是 $[-2^{31}, 2^{31}-1]$

long: 8字节, 范围是 $[-2^{63}, 2^{63}-1]$

---

(到这里突然想起来一个问题, 为什么c++里long与int都是整型, 范围一样, 还要定义一个long类型呢?)

搜索了一下: 是c的历史遗留问题呀, 那就没问题了

---

short: 2字节, 范围是 $[-2^{15}, 2^{15}-1]$

byte: (因为不知道了解了一下, 八位, 有符号, 以二进制补码表示); (原来没有long long)

char: 1字节, 范围十进制等效应该是[0, 65535]

float: 4字节, 这个范围要看数的位数

double: 8字节, 范围看数的位数

boolean: 1字节, {true, false}

#### Q3:

涉及到的时自动类型转换, b的值应为52。

原因的话, '0'的ascii码为48运算时'0'转为48运算。

(说起来, 初见类型转换的时候还觉得很神奇)

不过会有学长会去背一些重要符号的ascii码吗 (为了节约一点点的时间...)

#### Q4:

从上倒下, 应该是false, true, false

Integer应该默认范围实在 $[-128, 127]$ , 那它爆了之后会返回啥? 我去查了下Integer源码, 发现我还看不懂Java, 反正应该是false没错

在一开始, 对于new函数我一直认为的是它定义的是一个新指针, 本来就不是指向同一个空间, 不一样很自然。不知道java里兴不兴这样说。 (或者我一开始就错了)

具体查了之后, 是这样:

new Integer() 每次都会新建一个对象

Integer.valueOf() 会使用缓存池中的对象，多次调用会取得同一个对象的引用  
自然如此。

---

然后对于这种类型的可以把基本类型和string转换（这算是高精度算法吗？）

**看到这里有个疑问,原文是这样:**

在Java中，可能会使用到int类型的数据，但可能会有所要求：比如只能使用引用数据类型，但是由于int类型是基本数据类型，无法直接使用，所以需要进行包装，这就引入了Integer类，其他基本数据类型的包装类也是这样

有点没看懂，这和c++是不一样吗，查了一会没有明白，留作历史遗留问题吧。

再说有区别的话，integer是指针指向封装int的箱子，int就是int

**Q5:**

（我们采用一排排直接翻译的解释方法吧）

声明int类型变量a，并赋值为5；

声明int类型变量b，并赋值为7；

声明int类型变量c，并赋值为[ (a+1) +b ],并使a，b自增

此时c的值为13

最终输出：

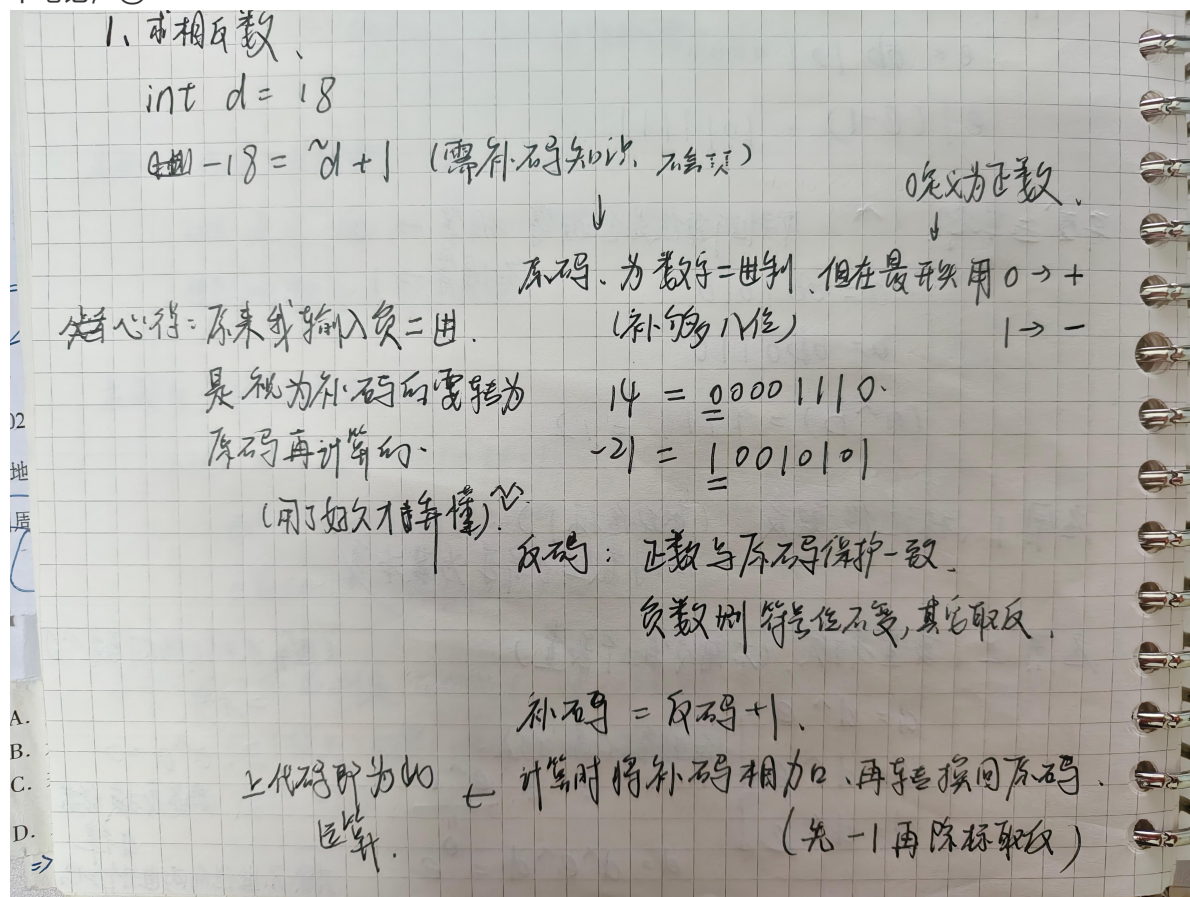
13

6 8

所以a++是先赋值后加，++a是先加后赋值，在很多带flag的算法里有用到

**Q6:**

我记得在遥远的过去，我学c++的时候老师叫我，有兴趣自己了解补码是啥，现在有点忘了，等我去翻下笔记，😞



没记错的话&位与运算符，应该是对于二进制每一位，两数均为1才得1。

但是在了解了之后我有点搞不清楚，计算机运算是把补码位与还是把原码位与了，有点难绷啊。

我去敲代码试试看。等我一下！

验证了一下应该没问题~~(应该?)~~:

如果是补码位与:

[illegible]

然后是lowbit:

$n \& (-n)$  代表留下 $n$ 二进制最右边的第一位其他归零所得到的数, 至少算了几个是这样

要怎么证明我不太明白，但源代码应该还是知道的

```
int lowbit(int x){
return x&(~n+1);
}
```

