**以下两个片段执行结果差异的原因是什么？**

片段一：

```
short  s=1;
s=s+1;
```

片段二：

```
short  s=1;
s+=1;
```

**可以自己组织一下答案，最后看结论**

**结论分析：**

片段一自然是编译不通过的 ，提示损失精度 。

**那么片段二为什么能编译通过呢?**

隐式类型转换可以从小到大自动转，即 byte->short->int->long，如果反过来会丢失精度，必须进行显示类型转换。

回到这一题来看，s+=1 的意思与 s = s+1 不同，s=s+1 这句先执行 s+1 然后把结果赋给 s，由于 1 为 int 类型，所以 s+1 的返回值是 int，编译器自动进行了隐式类型转换，所以将一个 int 类型赋给 short 就会出错。

而 s+=1 不同，由于他是使用+=操作符，在解析的时候 s+=1 就等价于 s = (short)(s+1)，也就是 s+=1 <=> s = (s 的类型)(s+1)。

（最后结论引自百度知道，略有删改。
解答出处：https://zhidao.baidu.com/question/250880637.html ）

**扩展：**

基本类型数据及所占字节

| 数据类型 | 所占字节 |
| --- | --- |
| boolean | 未定 |
| byte | 1 字节 |

| | |
|---|---|
| char | 2 字节 |
| short | 2 字节 |
| int | 4 字节 |
| long | 8 字节 |
| float | 4 字节 |
| double | 8 字节 |

## 隐式转换与显示转换概念

### 隐式类型转换

隐式转换也叫作自动类型转换，由系统自动完成.

从存储范围小的类型到存储范围大的类型.

byte ->short(char)->int->long->float->double

### 显示类型转换

显示类型转换也叫作强制类型转换，是从存储范围大的类型到存储范围小的类型.

当我们需要将数值范围较大的数值类型赋给数值范围较小的数值类型变量时，由于**此时可能会丢失精度**，因此，需要人为进行转换。我们称之为强制类型转换。
double→float→long→int→short(char)→byte

## 基本数据类型之间的转换规则

1.在一个双操作数以及位运算等算术运算式中，会根据操作数的类型将低级的数据类型自动转换为高级的数据类型，分为以下几种情况：

1）只要两个操作数中有一个是 double 类型的，另一个将会被转换成 double 类型，并且结果也是 double 类型；

2）只要两个操作数中有一个是 float 类型的，另一个将会被转换成 float 类型，并且结果也是 float 类型；

3）只要两个操作数中有一个是 long 类型的，另一个将会被转换成 long 类型，并且结果也是 long 类型；

4）两个操作数（包括 byte、short、int、char）都将会被转换成 int 类型，并且结果也是 int 类型。

2. 如果低级类型为 char 型，向高级类型（整型）转换时，会转换为对应 ASCII 码值，再做其它类型的自动转换。

3. 对于 byte,short,char 三种类型而言，他们是平级的，因此不能相互自动转换，可以使用下述的强制类型转换。 如：

```
short  i=99  ;
char  c=(char)i;
System.out.println("output:"+c);
```

4. 不能在布尔值和任何数字类型间强制类型转换；

5. 不同级别数据类型间的强制转换，可能会导致溢出或精度的下降。

6. 当字节类型变量参与运算，java 作自动数据运算类型的提升，将其转换为 int 类型。

例如：

```
byte  b;
b=3;
b=(byte)(b*3);//必须声明 byte。
```