的位将丢失。把[illegible]移入 0。对于有符号值而言，左侧移入 1 还是 0 取决于被移动数字的符号，还取决于该操作在计算机上的实现方式。如果符号位是 0（表示该值是正的），不管是哪种机器，都将移入 0。然而，如果符号位是 1，那么在一些计算机上将移入 1，而其他计算机上则移入 0。前一类型的运算符通常称为算术右移，而后者通常称为逻辑右移。

警告

对于系统使用算术右移还是逻辑右移，千万不要进行猜测。如果进行此类的假设，那么在一个系统中可正确进行有符号右移运算的程序，有可能在其他系统上运行失败。

如果 w1 是 unsigned int，用 32 位表示它，并且 w1 等于十六进制数的 F777EE22，那么使用语句

```
w1 >>= 1;
```

将 w1 右移一位后，w1 等于十六进制数的 7BBBF711，表示如下：

```
w1       1111 0111 0111 0111 1110 1110 0010 0010    0xF777EE22
w1 >> 1  0111 1011 1011 1011 1111 0111 0001 0001    0x7BBBF711
```

如果将 w1 声明为（有符号的）short int，在某些计算机上会得到相同的结果，而在其他计算机上，如果将该运算作为算术右移来执行，结果将会是 FBBBF711。

应该注意到，如果试图用大于或等于该数据项的位数将值向左或向右移位，那么该 Objective-C 语言对结果没有规定。因此，如果计算机用 64 位表示整数，那么把一个整数向左或向右移动 64 位或更多位时，不能保证在你的程序中得到确定的结果。还应注意到，如果使用负数对值移位，结果将同样是未定义的。

10.7 练习

1. 使用第 8 章“继承”中的 Rectangle 类，根据下面的声明增加一个初始化方法：

（注意：一定要使用这个初始化器重载 init。）

```
-(id) initWithWidth: (int) w andHeight: (int) h;
```

2. 假设将练习 1 中的初始化方法标记为 Rectangle 类的指定初始化方法，根据第 8 章定义的 Square 和 Rectangle 类，结合下面的声明，为 Square 类增加一个初始化方法：

```
-(id) initWithSide: (int) side;
```

3. 为 Fraction 类的 add:方法增加一个计数器来计算它的调用次数。如何获取这个变量的值？
4. 使用 typedef 定义一个名为 Day 的类型，可能的值为 Sunday、Monday、Tuesday、Wednesday、Thursday、Friday 和 Saturday。
5. 使用 typedef 和枚举数据类型定义名为 FractionObj 的类型，该类型允许编写如下语句：

```
FractionObj f1 = [[Fraction alloc] init],
            f2 = [[Fraction alloc] init];
```

6. 根据下面的定义：

```
float     f = 1.00;
short int i = 100;
long int  l = 500L;
double    d = 15.00;
```

和本章讲解表达式中操作数类型转换时列举的 7 个步骤，确定以下表达式的类型和值：

```
f + i
l / d
i / l + f
l * i
f / 2
i / (d + f)
l / (i * 2.0)
l + i / (double) l
```