但是在很多情况下，双端闭区间却不如单端闭区间好用。为什么这么说呢？相信你一定知道数组的下标都是从 0 开始的，一个长度为 10 的数组，它的下标区间范围是 0 到 9，因此左闭右开的区间在程序设计当中更加常用。Kotlin 中可以使用 `until` 关键字来创建一个左闭右开的区间，如下所示：

```
val range = 0 until 10
```

上述代码表示创建了一个 0 到 10 的左闭右开区间，它的数学表达方式是[0, 10)。修改 `main()` 函数中的代码，使用 `until` 替代 `..` 关键字，你就会发现最后一行 10 不会再打印出来了。

默认情况下，`for-in` 循环每次执行循环时会在区间范围内递增 1，相当于 Java `for-i` 循环中 `i++`的效果，而如果你想跳过其中的一些元素，可以使用 `step` 关键字：

```
fun main() {
    for (i in 0 until 10 step 2) {
        println(i)
    }
}
```

上述代码表示在遍历[0, 10)这个区间的时候，每次执行循环都会在区间范围内递增 2，相当于 `for-i` 循环中 `i = i + 2` 的效果。现在重新运行一下代码，结果如图 2.15 所示。

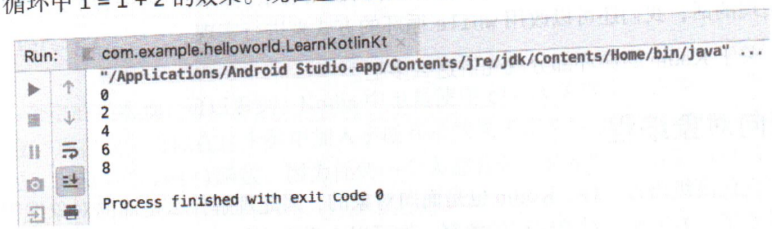

图 2.15　使用 `step` 跳过区间内的元素

可以看到，现在区间中所有奇数的元素都被跳过了。结合 `step` 关键字，我们就能够实现一些更加复杂的循环逻辑。

不过，前面我们所学习的 `..` 和 `until` 关键字都要求区间的左端必须小于等于区间的右端，也就是这两种关键字创建的都是一个升序的区间。如果你想创建一个降序的区间，可以使用 `downTo` 关键字，用法如下：

```
fun main() {
    for (i in 10 downTo 1) {
        println(i)
    }
}
```

这里我们创建了一个[10, 1]的降序区间，现在重新运行一下代码，结果如图 2.16 所示。