```
    } else {
        num2
    }
    return value
}
```

注意这里的代码变化，if 语句使用每个条件的最后一行代码作为返回值，并将返回值赋值给了 value 变量。由于现在没有重新赋值的情况了，因此可以使用 val 关键字来声明 value 变量，最终将 value 变量返回。

仔细观察上述代码，你会发现 value 其实也是一个多余的变量，我们可以直接将 if 语句返回，这样代码将会变得更加精简，如下所示：

```
fun largerNumber(num1: Int, num2: Int): Int {
    return if (num1 > num2) {
        num1
    } else {
        num2
    }
}
```

到这里为止，你觉得代码足够精简了吗？确实还不错，但是我们还可以做得更好。回顾一下刚刚在上一节里学过的语法糖，当一个函数只有一行代码时，可以省略函数体部分，直接将这一行代码使用等号串连在函数定义的尾部。虽然上述代码中的 largerNumber()函数不止只有一行代码，但是它和只有一行代码的作用是相同的，只是返回了一下 if 语句的返回值而已，符合该语法糖的使用条件。那么我们就可以将代码进一步精简：

```
fun largerNumber(num1: Int, num2: Int) = if (num1 > num2) {
    num1
} else {
    num2
}
```

前面我之所以说这个语法糖非常重要，就是因为它除了可以应用于函数只有一行代码的情况，还可以结合 Kotlin 的很多语法来使用，所以它的应用场景非常广泛。

当然，如果你愿意，还可以将上述代码再精简一下，直接压缩成一行代码：

```
fun largerNumber(num1: Int, num2: Int) = if (num1 > num2) num1 else num2
```

怎么样？通过一个简单的 if 语句，我们挖掘出了 Kotlin 这么多好玩的语法特性，现在你应该能逐渐体会到 Kotlin 的魅力了吧？

2.4.2　when 条件语句

接下来我们开始学习 when。Kotlin 中的 when 语句有点类似于 Java 中的 switch 语句，但它又远比 switch 语句强大得多。