```
{ /* 使用 param1 和 param2 的代码放在这里 */ ... });
```

习惯之后，会发现 Lambda 表达式提供的语法比匿名方法更简洁和自然。另外，正如本书后面要讲到的，C#许多比较高级的领域会大量使用 Lambda 表达式。总的来说，应该尽可能在代码中使用 Lambda 表达式而不是匿名方法。

18.4　比较数组和集合

数组和集合的重要差异总结如下。

- 数组实例具有固定大小，不能增大或缩小。集合则可根据需要动态改变大小。
- 数组可以多维，集合则是线性。但集合中的项可以是集合自身，所以可用集合的集合来模拟多维数组。
- 数组中的项通过索引来存储和获取。并非所有集合都支持这种语法。例如，要用 `Add` 或 `Insert` 方法在 `List<T>`集合中存储项，用 `Find` 方法获取项。
- 许多集合类都提供了 `ToArray` 方法，能创建数组并用集合中的项来填充。复制到数组的项不从集合中删除。另外，这些集合还提供了直接从数组填充集合的构造器。

使用集合类来玩牌

以下练习修改第 10 章的扑克牌游戏来使用集合而不是数组。

➢ 用集合实现扑克牌游戏

1. 如 Microsoft Visual Studio 2017 尚未启动，请启动。
2. 打开 Cards 解决方案，它位于“文档”文件夹下的\Microsoft Press\VCSBS\Chapter 18\Cards 子文件夹。

   该项目更新了第 10 章使用数组实现的版本。修改了 `PlayingCard` 类，牌的点数和花色作为只读属性公开。
3. 在“代码和文本编辑器”窗口中显示 Pack.cs。在文件顶部添加以下 `using` 指令：

   ```
   using System.Collections.Generic;
   ```
4. 将 `Pack` 类中的二维数组 `cardPack` 改成 `Dictionary<Suit, List< PlayingCard>>` 对象，如加粗的代码所示：

   ```
   class Pack
   {
      ...
      private Dictionary<Suit, List<PlayingCard>> cardPack;
   ```