用于处理数组的语法比较复杂。OOP 技术可以创建在内部执行大多数此类处理的类，因此简化了使用项列表或数组的代码。

C#中的数组实现为 System.Array 类的实例，它们只是集合类(Collection Class)中的一种类型。集合类一般用于处理对象列表，其功能比简单数组要多，功能大多是通过实现 System.Collections 名称空间中的接口而获得的，因此集合的语法已经标准化了。这个名称空间还包含其他一些有趣的东西，例如，以不同于 System.Array 的方式实现这些接口的类。

集合的功能(包括基本功能，例如，用[index]语法访问集合中的项)可以通过接口来实现，所以不仅可以使用基本集合类，例如 System.Array，还可以创建自己的定制集合类。这些集合可以专用于要枚举的对象(即要从中建立集合的对象)。这么做的一个优点是定制的集合类可以是强类型化的。也就是说，从集合中提取项时，不需要把它们转换为正确类型。另一个优点是提供专用的方法，例如，可以提供获得项子集的快捷方法。在扑克牌示例中，可以添加一个方法，来获得特定花色中的所有 Card 项。

System.Collections 名称空间中的以下几个接口提供了基本的集合功能：

- IEnumerable——可以迭代集合中的项。
- ICollection——继承于 IEnumerable。可以获取集合中项的个数，并能把项复制到一个简单的数组类型中。
- IList——继承于 IEnumerable 和 ICollection。提供了集合的项列表，允许访问这些项，并提供其他一些与项列表相关的基本功能。
- IDictionary——继承于 IEnumerable 和 ICollection。类似于 IList，但提供了可通过键值(而不是索引)访问的项列表。

System.Array 类实现了 IList、ICollection 和 IEnumerable，但不支持 IList 的一些更高级功能，它表示大小固定的项列表。

11.1.1 使用集合

Systems.Collections 名称空间中的类 System.Collections.ArrayList 也实现了 IList、ICollection 和 IEnumerable 接口，但实现方式比 System.Array 更复杂。数组的大小是固定不变的(不能添加或删除元素)，而这个类可以用于表示大小可变的项列表。为了更准确地理解这个高级集合的功能，下面列举一个使用这个类和一个简单数组的示例。

试一试 数组和高级集合：Ch11Ex01

(1) 在 C:\BeginningCSharp7\Chapter11 目录中创建一个新的控制台应用程序 Ch11Ex01。

(2) 在 Solution Explorer 窗口中右击项目，选择 Add | Class 选项，给项目添加 3 个新类：Animal、Cow 和 Chicken。

(3) 修改 Animal.cs 中的代码，如下所示：

```
namespace Ch11Ex01
{
   public abstract class Animal
   {
      protected string name;
      public string Name
      {
         get { return name; }
         set { name = value; }
      }
      public Animal() => name = "The animal with no name";

      public Animal(string newName)=> name = newName;

      public void Feed() => WriteLine($"{name} has been fed.");
   }
}
```

(4) 修改 Cow.cs 中的代码，如下所示：