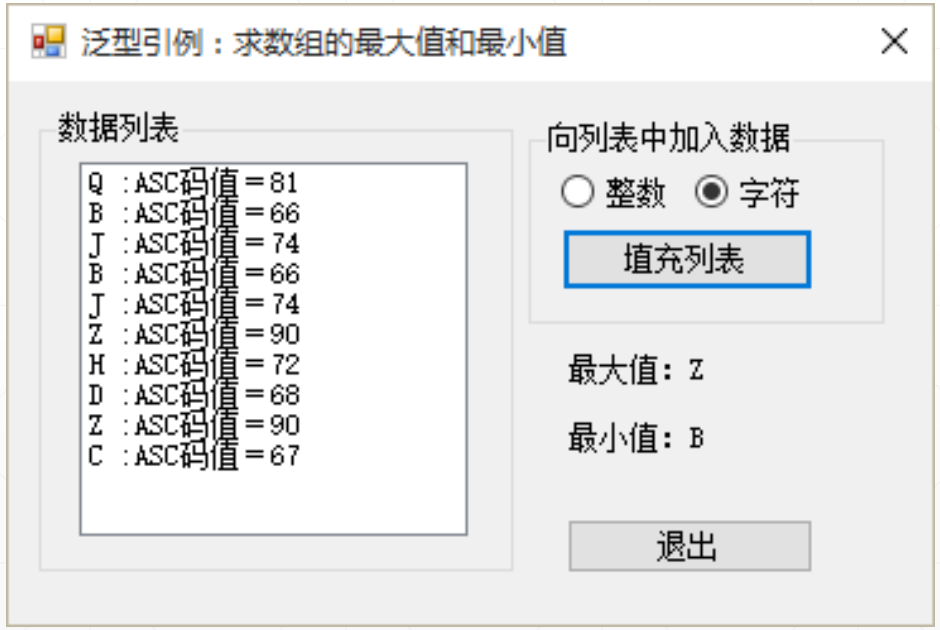
泛型编程技术

北京理工大学计算机学院
金旭亮

1 为什么引入“泛型”？

典型开发场景

我们需要写个程序求出某数组元素中的最大值与最小值



示例： MaxMinValueForGP1

示例程序中存在的问题

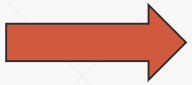
需要为不同类型的数组各提供一个代码类似的方法。

```
1.  private void GetMaxMinValueFromIntArray( int[ ] datas)

2.  private  void GetMaxMinValueFromCharArray(char [ ] datas  )
```

当增加一种新的类型时，需要再写一个类似的方法……

这些函数完成的工作都是一样的，不同之处仅仅在于处理的数据类型不一样，能否将这些非常类似的函数“合为一个”？



泛型可用于解决这个问题！

定义泛型方法：

泛型类型参数

```
private void GetMaxMinVauleFromArray<T>(
        T[] datas ,ref T Max ,ref T Min)
        where T:IComparable
```

泛型类型约束

方法参数前的**ref**关键字，表明方法将会修改传入变量的原始值，这就是说，方法执行完毕之后，传入的实参值将会被永久改变，方法调用语句后面的代码将得到一个被修改后的新值。

使用泛型方法：

```
GetMaxMinVauleFromArray<int>(
        IntArray, ref MaxInt , ref MinInt);
GetMaxMinVauleFromArray<char>(
        CharArray, ref MaxChar, ref MinChar);
```

使用时动态替换的类型

2 泛型编程基础技巧

泛型类的编写方法

只需给类指定一个（或多个）“类型参数”，然后，将原先具体的数据类型用定义的类型参数替换即可。

```
public class GenericList<T>
{
    void Add(T input) { }
}
```

设定泛型类型参数的约束

如果希望限制泛型类所能适用的数据类型，可以给类型参数添加约束

```
class MyClass<T,U>
        where T:struct
        where U:IComparable,new()
    {
        T value;
        void fun(U obj)
        {
        }
    }
```

类型参数的约束：

1. 类型T必须为值类型
2. 类型U必须实现IComparable接口并且必须具有一个无参数的构造函数

几种常用的泛型参数约束

T：struct	类型参数必须是值类型。可以指定除 以外的任何值类型。
T：class	类型参数必须是引用类型；这一点也适用于任何类、接口、委托或数组类型。
T：new()	类型参数必须具有无参数的公共构造函数。当与其他约束一起使用时，new() 约束必须最后指定。
T：<基类名>	类型参数必须是指定的基类或派生自指定的基类。
T：<接口名称>	类型参数必须是指定的接口或实现指定的接口。可以指定多个接口约束。约束接口也可以是泛型的。

编写泛型方法

当类中的某个方法使用了类型参数，这种方法称为“**泛型方法**”。

泛型方法多出现在泛型类中，但在普通的类中也可以直接定义泛型方法：

```
class MyClass
{
        static void Swap<T>(ref T lhs, ref T rhs)
        {
            T temp;
            temp = lhs;
            lhs = rhs;
            rhs = temp;
        }

}
```

泛型类与泛型方法的示例

示例： CSMaxMinValueForGP2

泛型接口

将泛型类的class关键字换为Interface，去掉所有实现代码，就定义了一个泛型接口。

.NET Framework基类库中定义了多个泛型接口，如IEnumerable<T>、IComparable<T>等，一个示例如下：

```
public interface IComparable<T>
{
    int CompareTo (T other);
}
```

泛型接口的使用

1. 一个泛型类的类型参数可添加多个泛型接口约束，其含义为：**此类型参数必须实现所有的这些接口**。

```
class Stack<T>
        where T : IComparable<T>,
                  IEnumerable<T>
{
    //...
}
```

2. 一个泛型接口也可以定义多个泛型类型参数：

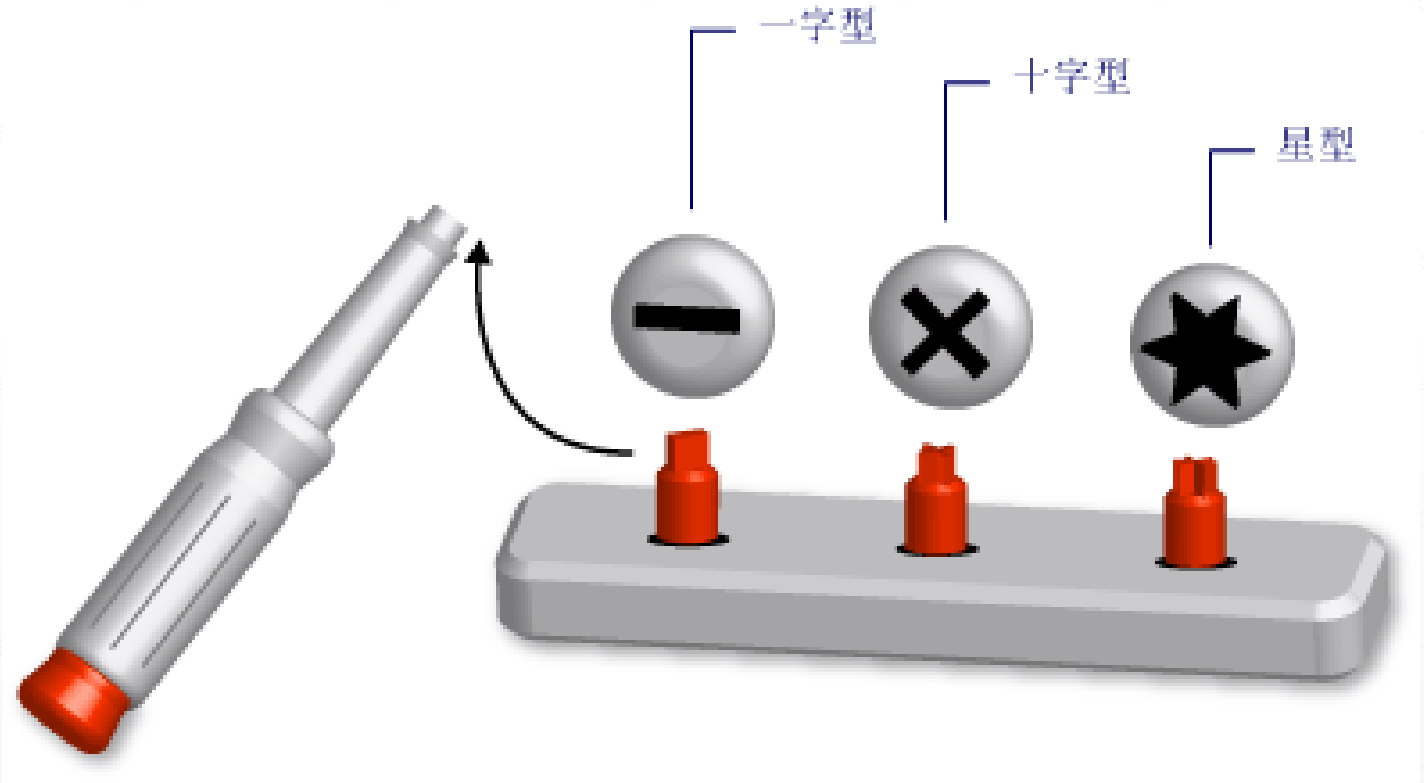
```
interface IDictionary<K, V>
{
    //...
}
```

检验你的学习成果

请编写一个泛型方法，能对一个数组（或集合）中的元素进行排序。排序所用的算法自选。

泛型编程小结

一张图，帮助你理解泛型编程技术！

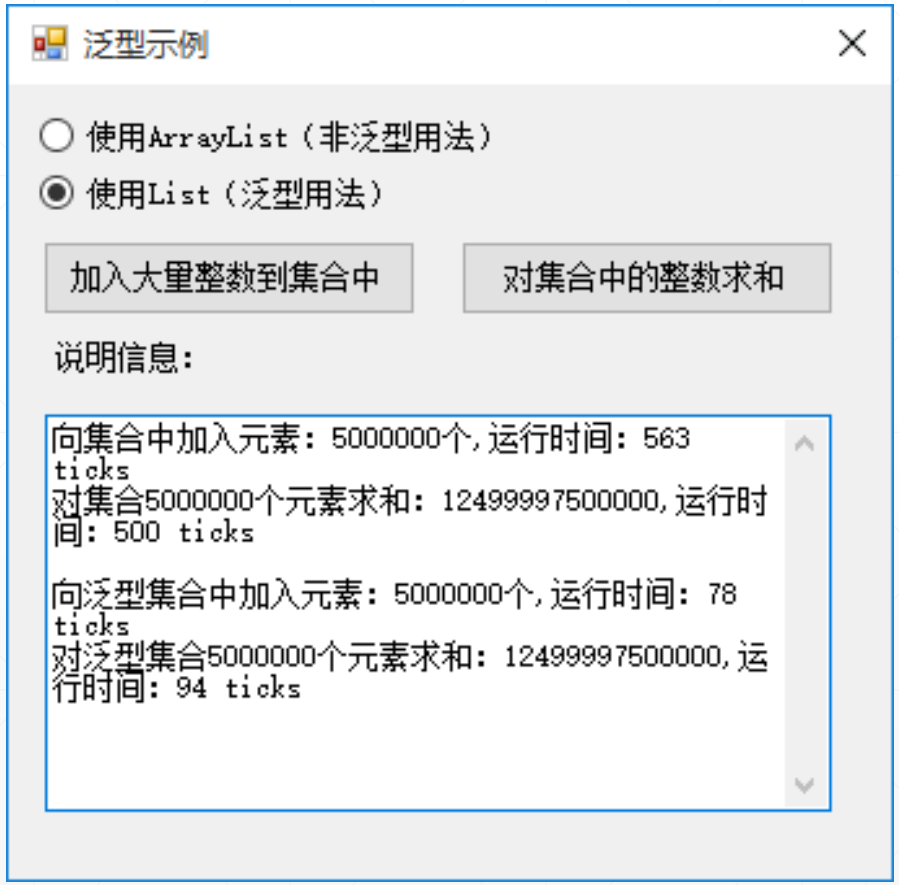


使用泛型编程有哪些好处？

减少类的数目	• 结构与功能高度类似的类，可以被“合并”为一个
剖离数据结构与算法	• 可以编写一些“通用”的算法，这些算法可以应用于多种数据类型
减少编码错误	• 可以在编译期间通过检查数据类型而发现许多编码错误，可以在编程时直接使用集成开发环境提供的下拉列表功能
提升程序运行效率	• 可以减少装箱和拆箱以及数据类型转换等操作带来的性能损失

事实胜于雄辩！

现场体验泛型集合对性能的提升.....



示例：HowFastIsGeneric