# 实验五 结构体、接口编程练习

## 【实验目的】

1. 了解结构体的概念，理解结构体和类相比的异同，掌握结构体的使用方法；

2. 了解什么是接口，接口和抽象类的异同，掌握接口的方法实现，接口方法的重定义。

## 【实验要求】

1. 写出能够使用结构体的程序；

2. 写出使用接口实现的程序；

3. 写出程序，并调试程序，要给出测试数据和实验结果。

4. 整理上机步骤，总结经验和体会。

5. 完成实验日志和上交程序。

## 【实验内容】

### 一、接口的定义与作用

接口可以看作是没有实现的方法和常量的集合。接口与抽象类相似，接口中的方法只是做了声明，而没有定义任何具体的操作方法。使用接口是为了解决C# 语言中不支持多重继承的问题。单继承可使语言本身结构简单，层次清楚，易于管理，安全可靠，避免冲突。但同时限制了语言的功能。为了在实际应用中实现多重继承的功能，C# 使用了接口技术，一个类可以实现多个接口以达到使用公用常量和一些常用的方法。

### 二、分析实现接口的程序文件

分析以下实现接口的程序文件并回答问题：

* 本程序中的接口包含方法的构成是哪些；

**答：构成为返回值，函数名，参数类型，形式参数，不包括函数体。**

* 实现接口的类包含哪些元素？

**答：非抽象类至少包含对接口中所有方法的实现。**

* 类实现接口方法的参数如何变换实现的？

**答：通过强制转换，将接口方法的参数转换成类中的参数类型。**

* 给出程序的输出结果。

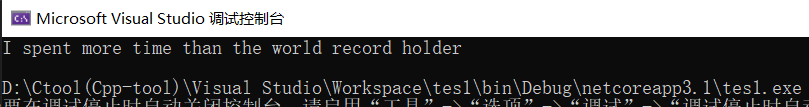


图5-1 程序输出结果

代码如下：

using System;

public interface IComparable

{

int CompareTo(IComparable comp);

}

public class TimeSpan : IComparable

{

private uint totalSeconds;

public TimeSpan()

{

totalSeconds = 0;

}

public TimeSpan(uint initialSeconds)

{

totalSeconds = initialSeconds;

}

public uint Seconds

{

get

{

return totalSeconds;

}

set

{

totalSeconds = value;

}

}

public int CompareTo(IComparable comp)

{

TimeSpan compareTime = (TimeSpan) comp;

if(totalSeconds > compareTime.Seconds)

return 1;

else if(compareTime.Seconds == totalSeconds)

return 0;

else

return -1;

}

}

class Tester

{

public static void Main()

{

TimeSpan myTime = new TimeSpan(3450);

TimeSpan worldRecord = new TimeSpan(1239);

if(myTime.CompareTo(worldRecord) < 0)

Console.WriteLine("My time is below the world record");

else if(myTime.CompareTo(worldRecord) == 0)

Console.WriteLine("My time is the same as the world record");

else

Console.WriteLine("I spent more time than the world record holder");

}

}

### 三、覆盖虚接口程序

以下程序组合了多种功能，请参考如下代码解释并回答问题。

* 该程序包含的类的个数和关系？

**答：包含了4个类，前两个是继承的关系；后面两个和其他的类没有直接关系。**

* 类对接口的实现有何区别？

**答：非抽象类实现接口，必须实现接口中的所有方法；类可以继承多个接口，但不能有多个基类。**

* 第一个类中无参数构造函数是否起作用，是否可以删除不用？

**答：这个地方可以删除，因为派生类中调用了基类的有参构造函数，所以不会调用基类的无参构造函数。**

* 类中的属性在哪里被应用到？

**答：类中的属性用于访问和修改类中的成员变量，成员变量是私有的，属性则是公有的用于访问成员变量的途径。**

* 第一个类中哪些成员被继承，列出所有？

**答：成员变量totalSeconds及其属性Seconds，方法CompareTo**

* 第二个类中构造方法成员如何实现，有何意义？可以去掉么？

**答：通过调用基类的构造函数实现；考虑程序输出的话不能去掉，因为主函数中使用了该构造函数；不考虑输出的话，可以删除，因为基类中定义了无参构造函数，不会出现变异错误。**

* 第二个类覆盖第一个类中接口虚方法，程序体现了什么功能区别？

**答：体现了多态性。**

* Sorter类有何作用？你能否根据Sorter类写一个十个数比较大小的冒泡法程序？

**答：可以将比较的数组按升序排序；代码如下：**

**bool swapped = true;**

**type array[10];**

**输入array[10];**

**for (int i = 0; swapped; i++)**

**{**

**Console.WriteLine("run outter");**

**swapped = false;**

**for (int j = 0; j < (array.Length - (i + 1)); j++)**

**{**

**if (array[j]>array[j+1])**

**{**

**Console.WriteLine("run inner");**

**Swap(j, j + 1, array);**

**swapped = true;**

**}**

**}**

**}**

* Sorter类中for (int i = 0; swapped; i++)和 //for (int i = 0; i< bubbles.Length; i++)两行是否作用相同？

**答：作用不相同；前者是改进后的冒泡排序的最外层循环，通过判断数组中是否发生交换来决定循环体是否执行。**

* 你知道Console.WriteLine("run outter");和Console.WriteLine("run inner");在程序运行过程中可以起到什么作用？

**答：“run outter”可以提示最外层循环执行的次数，同时提示数组中有多少连续的不规则片段；“run inner”可以提示内层循环执行的次数，同时提示了交换的次数。**

* 将Main方法中的TimeSpan对象语句（注释掉的5行）和TimeSpanAdvanced对象语句选择轮流注释，体验排序结果的异同。

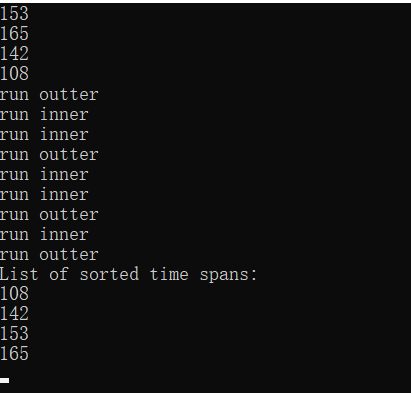


图5-2 注释掉TimeSpanAdvanced语句的输出结果

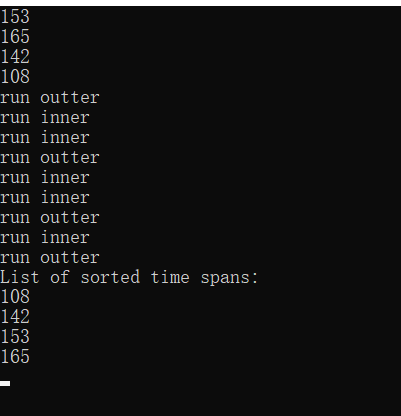


图5-3 注释掉TimeSpan对象语句后的输出结果

* 语句Sorter.BubbleSortAscending(raceTimes);前后的foreach语句功能区别。

**答：之前的foreach语句用于输出未排序的数组；之后的foreach语句用于输出排序后的数组。**

using System;

public interface IComparable

{

int CompareTo(IComparable comp);

}

public class TimeSpan : IComparable

{

private uint totalSeconds;

public TimeSpan()

{

totalSeconds = 0;

}

public TimeSpan(uint initialSeconds)

{

totalSeconds = initialSeconds;

}

public uint Seconds

{

get

{

return totalSeconds;

}

set

{

totalSeconds = value;

}

}

public virtual int CompareTo(IComparable comp)

{

TimeSpan compareTime = (TimeSpan)comp;

if (totalSeconds > compareTime.Seconds)

return 1;

else if (compareTime.Seconds == totalSeconds)

return 0;

else

return -1;

}

}

public class TimeSpanAdvanced : TimeSpan

{

public TimeSpanAdvanced(uint initialSeconds): base(initialSeconds)

{

//

}

public override int CompareTo(IComparable comp)

{

TimeSpan compareTime = (TimeSpan)comp;

if (base.Seconds > compareTime.Seconds)

{

if (base.Seconds > (compareTime.Seconds + 50))

return 2;

else

return 1;

}

else if (base.Seconds < compareTime.Seconds)

{

if (base.Seconds < (compareTime.Seconds - 50))

return -2;

else

return -1;

}

else

return 0;

}

}

class Sorter

{

// Sort the comparable elements of an array in ascending order

public static void BubbleSortAscending(IComparable[] bubbles)

{

bool swapped = true;

for (int i = 0; swapped; i++)

//for (int i = 0; i< bubbles.Length; i++)

{

Console.WriteLine("run outter");

swapped = false;

for (int j = 0; j < (bubbles.Length - (i + 1)); j++)

{

if (bubbles[j].CompareTo(bubbles[j + 1]) > 0)

{

Console.WriteLine("run inner");

Swap(j, j + 1, bubbles);

swapped = true;

}

}

}

}

//Swap two elements of an array

public static void Swap(int first, int second, IComparable[] arr)

{

IComparable temp;

temp = arr[first];

arr[first] = arr[second];

arr[second] = temp;

}

}

class Tester

{

public static void Main()

{

//TimeSpan[] raceTimes = new TimeSpan[4];

//raceTimes[0] = new TimeSpan(153);

//raceTimes[1] = new TimeSpan(165);

//raceTimes[2] = new TimeSpan(142);

//raceTimes[3] = new TimeSpan(108);

TimeSpanAdvanced[] raceTimes = new TimeSpanAdvanced[4];

raceTimes[0] = new TimeSpanAdvanced(153);

raceTimes[1] = new TimeSpanAdvanced(165);

raceTimes[2] = new TimeSpanAdvanced(142);

raceTimes[3] = new TimeSpanAdvanced(108);

foreach (TimeSpan time in raceTimes)

{

Console.WriteLine(time.Seconds);

}

Sorter.BubbleSortAscending(raceTimes);

Console.WriteLine("List of sorted time spans:");

foreach (TimeSpan time in raceTimes)

{

Console.WriteLine(time.Seconds);

}

Console.ReadLine();

}

}

### 四、结构体的使用

1.程序功能要求，创建三个结构体，MyCircle,MyCylinder,MyCone分别表示圆形、圆柱体和圆锥体，MyCircle包含一个int类型的成员r表示半径，MyCylinder和MyCone各自包含一个MyCircel类型的成员表示圆柱体和圆锥体的底面，成员h和volumn（都为整型）分别表示圆柱体和圆锥体的高和体积。写出结构体和程序的主方法求圆柱体和圆锥体的体积。

struct MyCircle

{

public int r;

}

struct MyCylinder

{

public MyCircle c;

public int h;

public int volumn;

}

struct MyCone

{

public MyCircle c;

public int h;

public int volumn;

}

public class Tester

{

public static void Main()

{

Console.Write("请输入底面半径：");

MyCircle c = new MyCircle();

c.r = int.Parse(Console.ReadLine());

Console.Write("请输入圆柱体高度：");

MyCylinder cy = new MyCylinder();

cy.h = int.Parse(Console.ReadLine());

cy.c = c;

Console.Write("请输入圆锥体高度：");

MyCone co = new MyCone();

co.h = int.Parse(Console.ReadLine());

co.c = c;

//计算圆柱体体积

double x = Math.PI \* cy.c.r \* cy.c.r;

double y = x \* cy.h;

cy.volumn = (int)y;

//计算圆锥体体积

double x2 = Math.PI \* co.c.r \* co.c.r;

double y2 = x2 \* co.h/3;

co.volumn = (int)y2;

//输出结果

// Console.WriteLine("圆柱体的体积为：", cy.volumn);

Console.Write("圆柱体的体积为：");

Console.Write(cy.volumn);

Console.Write("圆锥体的体积为：");

Console.Write(co.volumn);

Console.ReadLine();

}

}

问题：

1. 若取消// Console.WriteLine("圆柱体的体积为：", cy.volumn);前的注释替代分开输出的写法，看输出结果何变化，分析原因。

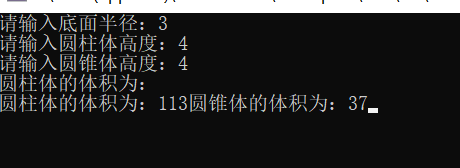


图5-4 去掉注释符的输出

**原因：没有将cy.volumn填充至语句中。**

1. 若将结构体中的变量不加public编译能通过么？为什么？

**答：不能通过编译；因为默认是私有字段。**

1. public的变量在Main中的使用方法。

**答：MyCircle c = new MyCircle();**

**c.r = int.Parse(Console.ReadLine());**

**可以用“.”访问结构体中的public字段。**

1. 不用new创建结构体将MyCircle c = new MyCircle();MyCylinder cy = new MyCylinder();MyCone co = new MyCone();改成什么样的语句起到同样的效果。

**答：可以改成：**

**MyCircle c;MyCylinder cy;MyCone co;**

1. 结构体中内嵌结构体实例的应用总结。

**答：内嵌结构体的访问需以级联方式层层向下访问，表现在代码上就是从最外层开始依次“.”至最底层。**

1. 和类相比结构体适合什么样的情况。

**答：结构体是值类型，存储于栈上，访问很快速。用结构体的情况有：希望能像int等基本数据类型一样自定义一个类型的时候；不需要从基类继承的时候；不需要多态性的时候；当栈的空间还有富余时；当成员逻辑比较简单时；...**

### 五、结构体实现

实现一个结构体Fraction,它包含两个int类型私有数据成员numerator和demominator。让Fraction包含以下元素：

* + - 一个带两个参数的构造函数来初始化numerator和demominator；
    - 用于存取numerator和demominator的属性；
    - 一个名叫value的属性，返回一个分数值，分数由(numerator/demominator)计算而得；
    - 覆盖ToString方法，返回下面的字符串“Fraction value:xxx”，其中的xxx是以字符串表示的分数值；
    - 编写适当的代码来测试此Fraction结构。

参考代码如下：

using System;

struct Fraction

{

private int numerator;

private int denominator;

public Fraction(int initNumerator, int initDenominator)

{

numerator = initNumerator;

denominator = initDenominator;

}

public int Numerator

{

get

{

return numerator;

}

set

{

numerator = value;

}

}

public int Denominator

{

get

{

return denominator;

}

set

{

denominator = value;

}

}

public double Value

{

get

{

return (double)numerator / (double)denominator;

}

}

public override string ToString()

{

string returnString;

returnString = "Fraction value: " + Value;

return returnString;

}

}

class Tester

{

public static void Main()

{

Fraction myFraction = new Fraction(1, 3);

Console.WriteLine(myFraction);

}

}

### 分析代码结果，总结结构体和类在值类型和引用类型上的区别。

**答案：区别：**

**由于结构是值类型，并且直接存储数据，因此在一个对象的主要成员为数据且数据量不大的情况下，使用结构会带来更好的性能。**

**因为结构是值类型，因此在为结构分配内存，或者当结构超出了作用域被删除时，性能会非常好，因为他们将内联或者保存在堆栈中。当把一个结构类型的变量赋值给另一个结构时，对性能的影响取决于结构的大小，如果结构的数据成员非常多而且复杂，就会造成损失。**

using System;

public struct TimeSpan

{

private uint totalSeconds;

public TimeSpan(uint initialTotalSeconds)

{

totalSeconds = initialTotalSeconds;

}

public uint Seconds

{

get

{

return totalSeconds;

}

set

{

totalSeconds = value;

}

}

}

class Tester

{

public static void Main()

{

TimeSpan myTime = new TimeSpan(480);

UpdateTime(myTime);

Console.WriteLine("Time outside UpdateTime method: {0}", myTime.Seconds);

}

public static void UpdateTime(TimeSpan timeUpdate)

{

timeUpdate.Seconds = timeUpdate.Seconds + 50;

Console.WriteLine("Time inside UpdateTime method: {0}", timeUpdate.Seconds);

}

}