# 实验四 操作符重载编程练习

## 【实验目的】

1. 了解操作符重载的意义，掌握操作符重载的一般方法。
2. 理解操作符对的重载实现，编写相应的程序。

## 【实验要求】

1. 进一步了解操作符重载的实现。

2. 掌握operator关键字的使用方法。

3. 掌握常用的操作符重载：数学操作符、逻辑操作符、转换操作符重载实现。

4.写出程序，并调试程序，要给出测试数据和实验结果。

5. 整理上机步骤，总结经验和体会。

6.完成实验日志和上交程序。

## 【实验内容】

* 1. 数学操作符重载程序
  2. 逻辑操作符重载程序
  3. 转换操作符重载程序

### 一、定义一个TimeSpan类，用TimeSpan.Add方法实现类中对象的加法

程序具体功能要求如下：

TimeSpan类含一个总耗费秒数变量，每小时秒数3600常量，每分钟秒数60常量；

构造方法实现无参数时总耗秒为初设为0，具有小时、分钟和秒参数时总耗秒为小时和分钟及秒总含秒数；

打印出总共消耗小时数、分钟数和秒数；

定义TimeSpan Add方法，实现两个TimeSpan对象的加和；

参考代码如下：

using System;

class TimeSpan

{

private uint totalSeconds;

private const uint SecondsInHour = 3600;

private const uint SecondsInMinute = 60;

public TimeSpan()

{

totalSeconds = 0;

}

public TimeSpan(uint initialHours, uint initialMinutes,

uint initialSeconds)

{

totalSeconds = initialHours \* SecondsInHour +

initialMinutes \* SecondsInMinute + initialSeconds;

}

public uint Seconds

{

get

{

return totalSeconds;

}

set

{

totalSeconds = value;

}

}

public void PrintHourMinSec()

{

uint hours;

uint minutes;

uint seconds;

hours = totalSeconds / SecondsInHour;

minutes = (totalSeconds % SecondsInHour) / SecondsInMinute;

seconds = (totalSeconds % SecondsInHour) % SecondsInMinute;

Console.WriteLine("{0} Hours {1} Minutes {2} Seconds",

hours, minutes, seconds);

}

public static TimeSpan Add(TimeSpan timeSpan1, TimeSpan timeSpan2)

{

TimeSpan sumTimeSpan = new TimeSpan();

sumTimeSpan.Seconds = timeSpan1.Seconds + timeSpan2.Seconds;

return sumTimeSpan;

}

}

class TimeSpanTest

{

public static void Main()

{

TimeSpan totalTime;

TimeSpan myTime = new TimeSpan(1,20,30);

TimeSpan yourTime = new TimeSpan(2,40,45);

totalTime = TimeSpan.Add(myTime, yourTime);

Console.Write("My time: "); myTime.PrintHourMinSec();

Console.Write("Your time: "); yourTime.PrintHourMinSec();

Console.Write("Total time: "); totalTime.PrintHourMinSec();

}

}

将以上程序的TimeSpanAdd方法改为加号操作符重载的实现，并回答有何异同。

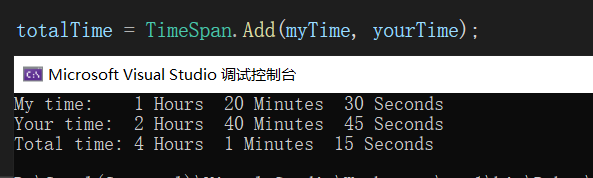


图1 使用Add函数进行相加

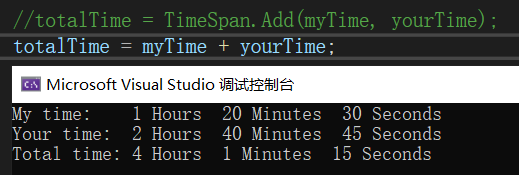


图2 使用重载的加法运算

**相同点：输入输出的参数和内在的逻辑是一样的。**

**不同点：Add函数需要调用函数执行，运算符重载只需要使用原本的“+”号就可以实现功能；从外在来看，运算符重载后的代码更简洁，更具有可读性。**

### 二、Explicit convertion显式转换练习

程序功能要求如下：

定义TimeSpan类型，其中定义显式操作符ushort实现将TimeSpan类对象的显式转换成ushort类型；

TimeSpan具有一个构造方法，内容可以由你任意书写；

参考代码如下：

using System;

class TimeSpan

{

private uint totalSeconds;

public TimeSpan(uint initialTotalSeconds)

{

totalSeconds = initialTotalSeconds;

}

public uint TotalSeconds

{

get

{

return totalSeconds;

}

}

public static explicit operator ushort(TimeSpan convertFrom)

{

Console.WriteLine("Converting from TimeSpan to ushort");

return (ushort)convertFrom.TotalSeconds;

}

}

class ExplicitConversionTester

{

public static void Main()

{

ushort simpleTimeSeconds;

TimeSpan myTime = new TimeSpan(130);

simpleTimeSeconds = (ushort) myTime;

Console.WriteLine("Value of simpleTimeSeconds: {0}", simpleTimeSeconds);

}

}

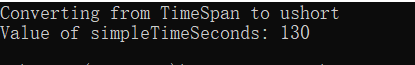


图3

### 三、Implicit convertion隐式转换练习

程序功能要求如下：

TimeSpan类含构造函数和打印方法并实现TimeSpan类型隐式转换为uint类型；

using System;

class TimeSpan

{

private uint totalSeconds;

public TimeSpan(uint initialTotalSeconds)

{

totalSeconds = initialTotalSeconds;

}

public void PrintSeconds()

{

Console.WriteLine("Total seconds: {0}", totalSeconds);

}

public static implicit operator TimeSpan(uint convertFrom)

{

TimeSpan newTimeSpan;

newTimeSpan = new TimeSpan(convertFrom);

Console.WriteLine("Converting from uint to TimeSpan");

return newTimeSpan;

}

}

class ImplicitConversionTester

{

public static void Main()

{

uint simpleTimeSeconds = 30;

TimeSpan myTime;

myTime = simpleTimeSeconds;

myTime.PrintSeconds();

}

}

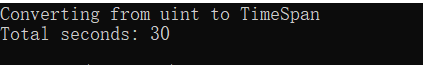


图4