**【实验2-1】**

有红，黄，黑，白 4 色球各一个，放置在编号为1、2、3、4 的4个盒子中，每个盒子放一只球，他们的顺序不知。

甲乙丙三人猜测放置顺序如下：

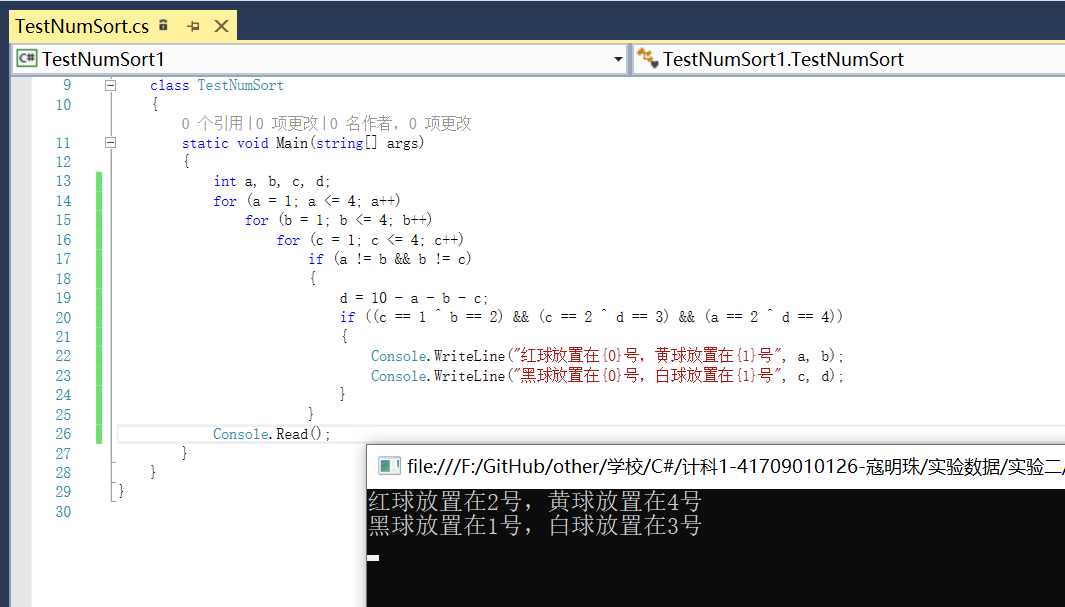
甲：黑球在 1 号盒子，黄球在 2 号盒子；

乙：黑球在 2 号盒子，白球在 3 号盒子；

丙：红球在 2 号盒子，白球在 4 号盒子；

结果证明，甲、乙、丙三人各猜中了一半，下面给出的程序就是找出 4 色球在盒子中的放置情况。

**A跟着学习：**运行结果：



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TestNumSort1

{

class TestNumSort

{

static void Main(string[] args)

{

int a, b, c, d;

for (a = 1; a <= 4; a++)

for (b = 1; b <= 4; b++)

for (c = 1; c <= 4; c++)

if (a != b && b != c)

{

d = 10 - a - b - c;

if ((c == 1 ^ b == 2) && (c == 2 ^ d == 3) && (a == 2 ^ d == 4))

{

Console.WriteLine("红球放置在{0}号，黄球放置在{1}号", a, b);

Console.WriteLine("黑球放置在{0}号，白球放置在{1}号", c, d);

}

}

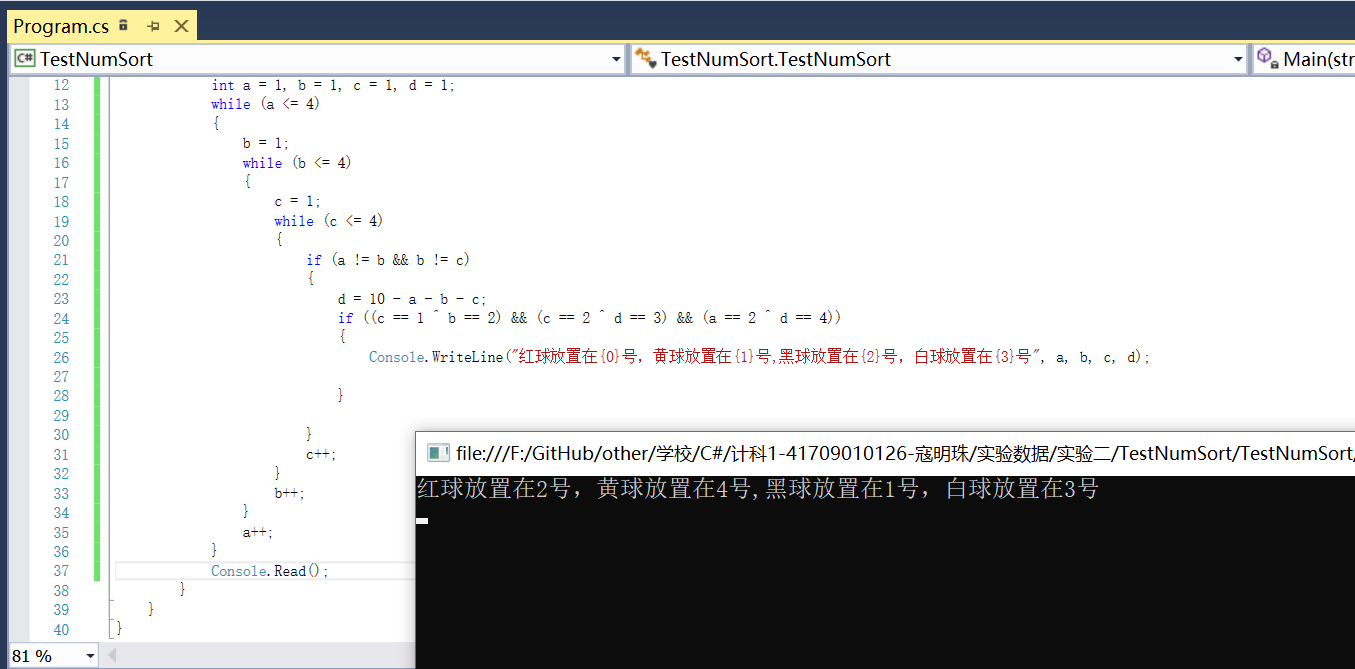
Console.Read();

}

}

}

**B自己练习：**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TestNumSort

{

class TestNumSort

{

static void Main(string[] args)

{

int a = 1, b = 1, c = 1, d = 1;

while (a <= 4)

{

b = 1;

while (b <= 4)

{

c = 1;

while (c <= 4)

{

if (a != b && b != c)

{

d = 10 - a - b - c;

if ((c == 1 ^ b == 2) && (c == 2 ^ d == 3) && (a == 2 ^ d == 4))

{

Console.WriteLine("红球放置在{0}号，黄球放置在{1}号,黑球放置在{2}号，白球放置在{3}号", a, b, c, d);

}

}

c++;

}

b++;

}

a++;

}

Console.Read();

}

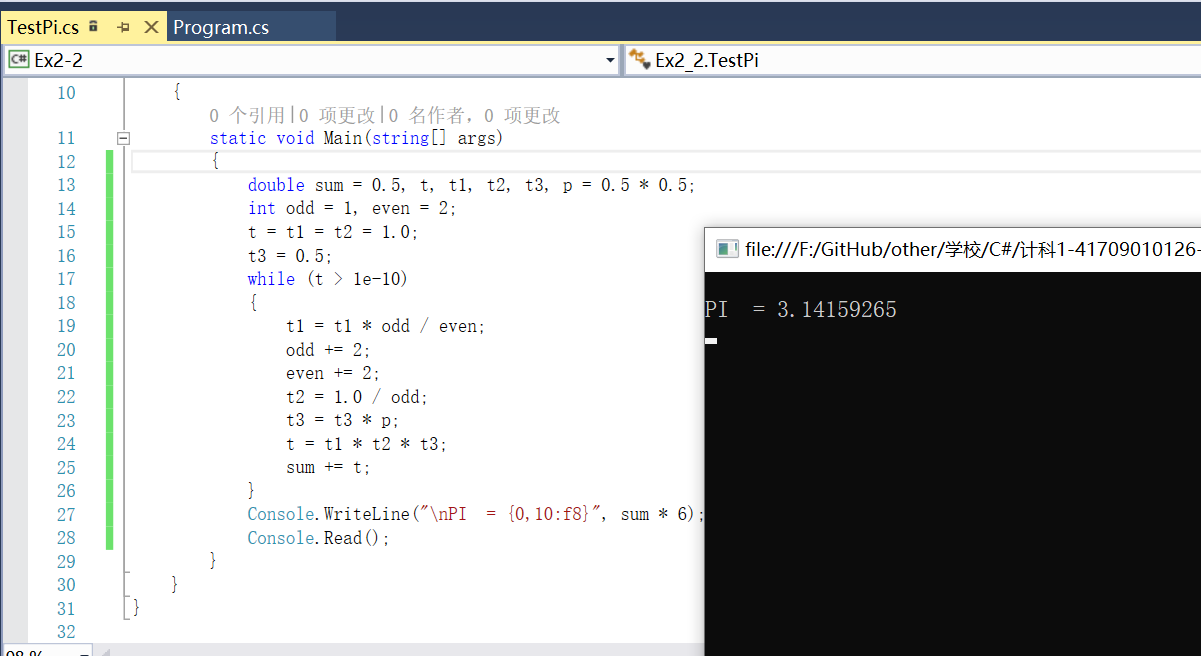
}

}

**【实验2-2】**

**A.自己练习**

根据给出的公式编程计算pi 的值。



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Ex2\_2

{

class TestPi

{

static void Main(string[] args)

{

double sum = 0.5, t, t1, t2, t3, p = 0.5 \* 0.5;

int odd = 1, even = 2;

t = t1 = t2 = 1.0;

t3 = 0.5;

while (t > 1e-10)

{

t1 = t1 \* odd / even;

odd += 2;

even += 2;

t2 = 1.0 / odd;

t3 = t3 \* p;

t = t1 \* t2 \* t3;

sum += t;

}

Console.WriteLine("\nPI = {0,10:f8}", sum \* 6);

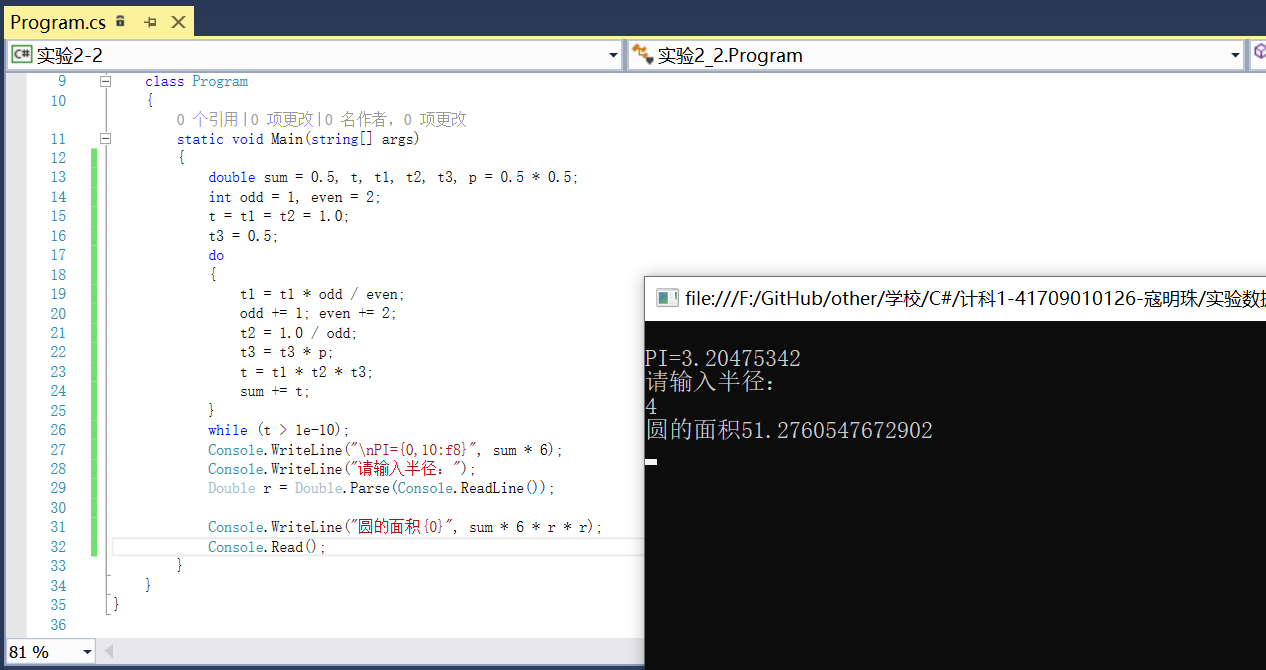
Console.Read();

}

}

}

**B.自己练习：**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace 实验2\_2

{

class Program

{

static void Main(string[] args)

{

double sum = 0.5, t, t1, t2, t3, p = 0.5 \* 0.5;

int odd = 1, even = 2;

t = t1 = t2 = 1.0;

t3 = 0.5;

do

{

t1 = t1 \* odd / even;

odd += 1; even += 2;

t2 = 1.0 / odd;

t3 = t3 \* p;

t = t1 \* t2 \* t3;

sum += t;

}

while (t > 1e-10);

Console.WriteLine("\nPI={0,10:f8}", sum \* 6);

Console.WriteLine("请输入半径：");

Double r = Double.Parse(Console.ReadLine());

Console.WriteLine("圆的面积{0}", sum \* 6 \* r \* r);

Console.Read();

}

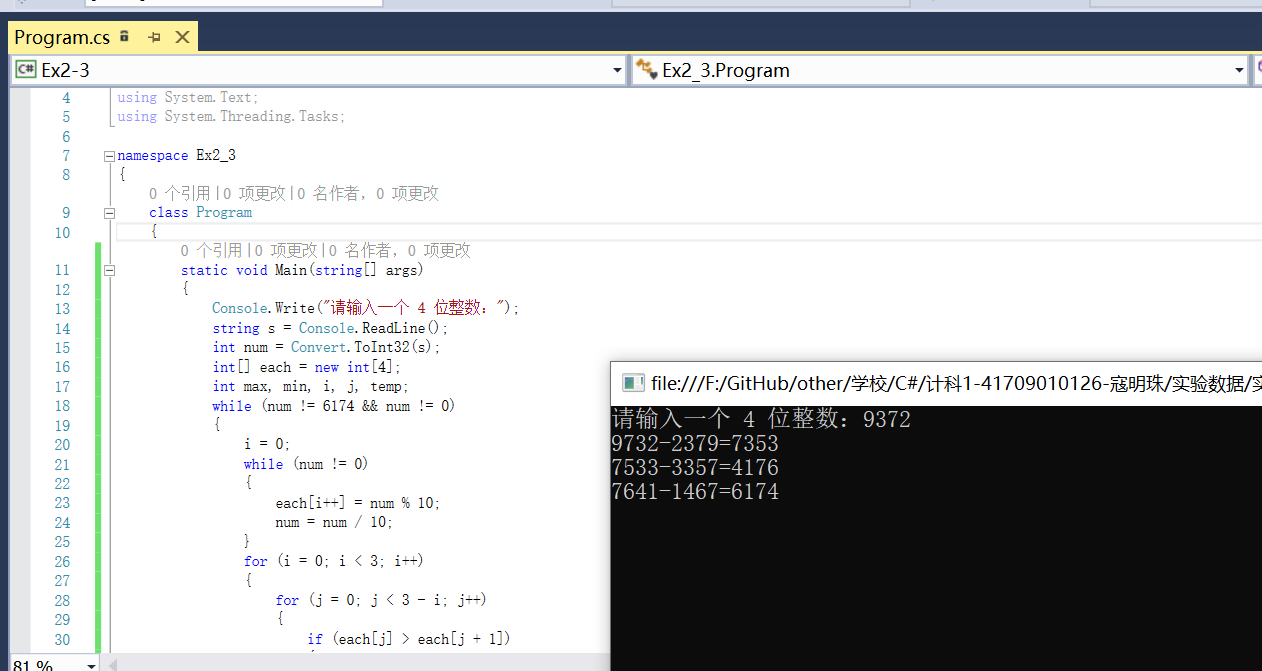
}

}

**【实验2-3】：**

编程进行卡布列克运算。所谓卡布列克运算，是指任意一个4位数，只要它们各个位上的数字不全相同，就有这样的规律:  
(1)把组成这个4位数的4个数字由大到小排列，形成由这4个数字构成的最大的4位数;  
(2)把组成这个4位数的4个数字由小到大排列，形成由这4个数字构成的最小的4位数(如果4个数字中含有0,则此数不足4位):  
(3)求出以上两数之差，得到一个新的4位数。重复以上过程，最后的结果总是6174。

**A.跟着学习**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Ex2\_3

{

class Program

{

static void Main(string[] args)

{

Console.Write("请输入一个 4 位整数：");

string s = Console.ReadLine();

int num = Convert.ToInt32(s);

int[] each = new int[4];

int max, min, i, j, temp;

while (num != 6174 && num != 0)

{

i = 0;

while (num != 0)

{

each[i++] = num % 10;

num = num / 10;

}

for (i = 0; i < 3; i++)

{

for (j = 0; j < 3 - i; j++)

{

if (each[j] > each[j + 1])

{

temp = each[j];

each[j] = each[j + 1];

each[j + 1] = temp;

}

}

}

min = each[0] \* 1000 + each[1] \* 100 + each[2] \* 10 + each[3];

max = each[3] \* 1000 + each[2] \* 100 + each[1] \* 10 + each[0];

num = max - min;

Console.WriteLine("{0}-{1}={2}", max, min, num);

}

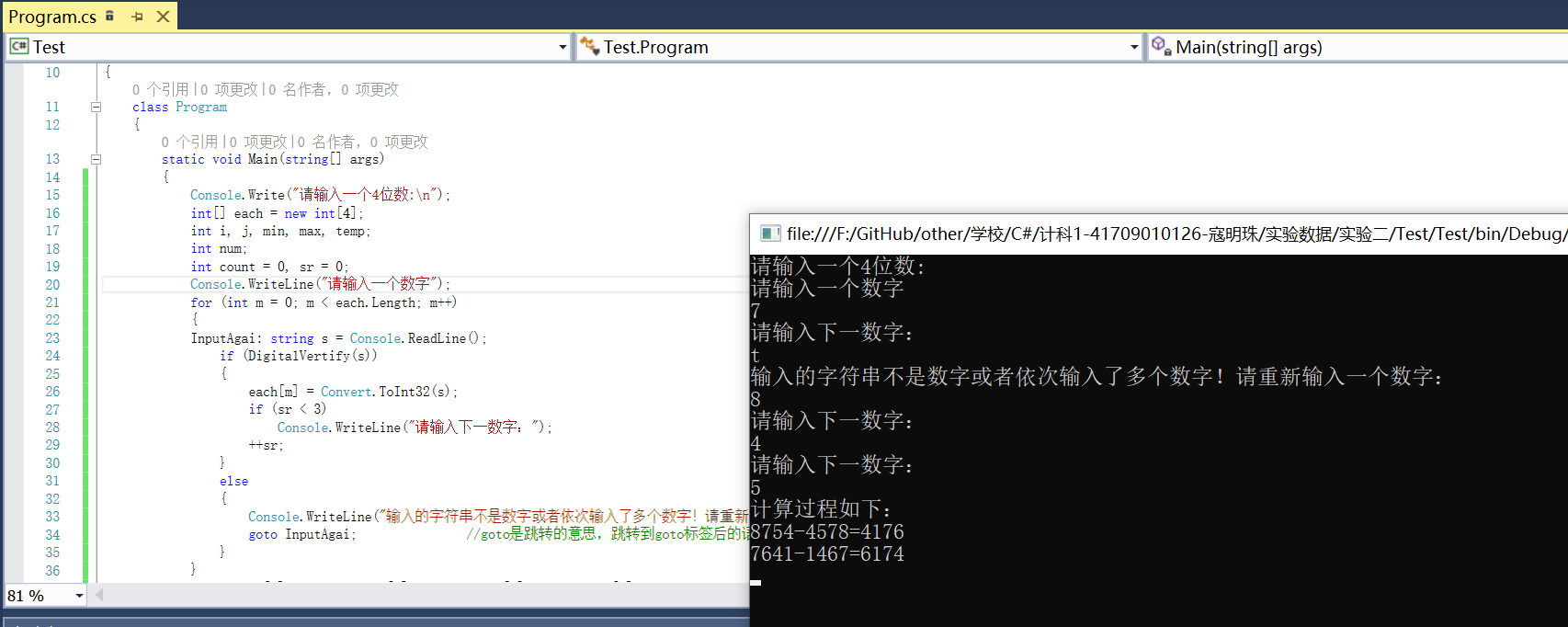
Console.Read();

}

}

}

**B.自己思考：**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions; //引入正则表达式程序集。

using System.Threading.Tasks;

namespace Test

{

class Program

{

static void Main(string[] args)

{

Console.Write("请输入一个4位数:\n");

int[] each = new int[4];

int i, j, min, max, temp;

int num;

int count = 0, sr = 0;

Console.WriteLine("请输入一个数字");

for (int m = 0; m < each.Length; m++)

{

InputAgai: string s = Console.ReadLine();

if (DigitalVertify(s))

{

each[m] = Convert.ToInt32(s);

if (sr < 3)

Console.WriteLine("请输入下一数字：");

++sr;

}

else

{

Console.WriteLine("输入的字符串不是数字或者依次输入了多个数字！请重新输入一个数字：");

goto InputAgai;

}

}

num = each[0] \* 1000 + each[1] \* 100 + each[2] \* 10 + each[3];

Console.WriteLine("计算过程如下：");

while (num != 6174 && num != 0)

{

i = 0;

while (num != 0)

{

each[i++] = num % 10;

num = num / 10;

}

for (i = 0; i < 3; i++)

{

for (j = 0; j < 3; j++)

{

if (each[j] > each[j + 1])

{

temp = each[j];

each[j] = each[j + 1];

each[j + 1] = temp;

}

}

}

min = each[0] \* 1000 + each[1] \* 100 + each[2] \* 10 + each[3];

max = each[0] + each[1] \* 10 + each[2] \* 100 + each[3] \* 1000;

num = max - min;

++count;

Console.WriteLine("{0}-{1}={2}", max, min, num);

}

Console.Read();

}

private static bool DigitalVertify(string s)

{

string pattern = "^[0-9]$";

Regex rx = new Regex(pattern);

return rx.IsMatch(s);

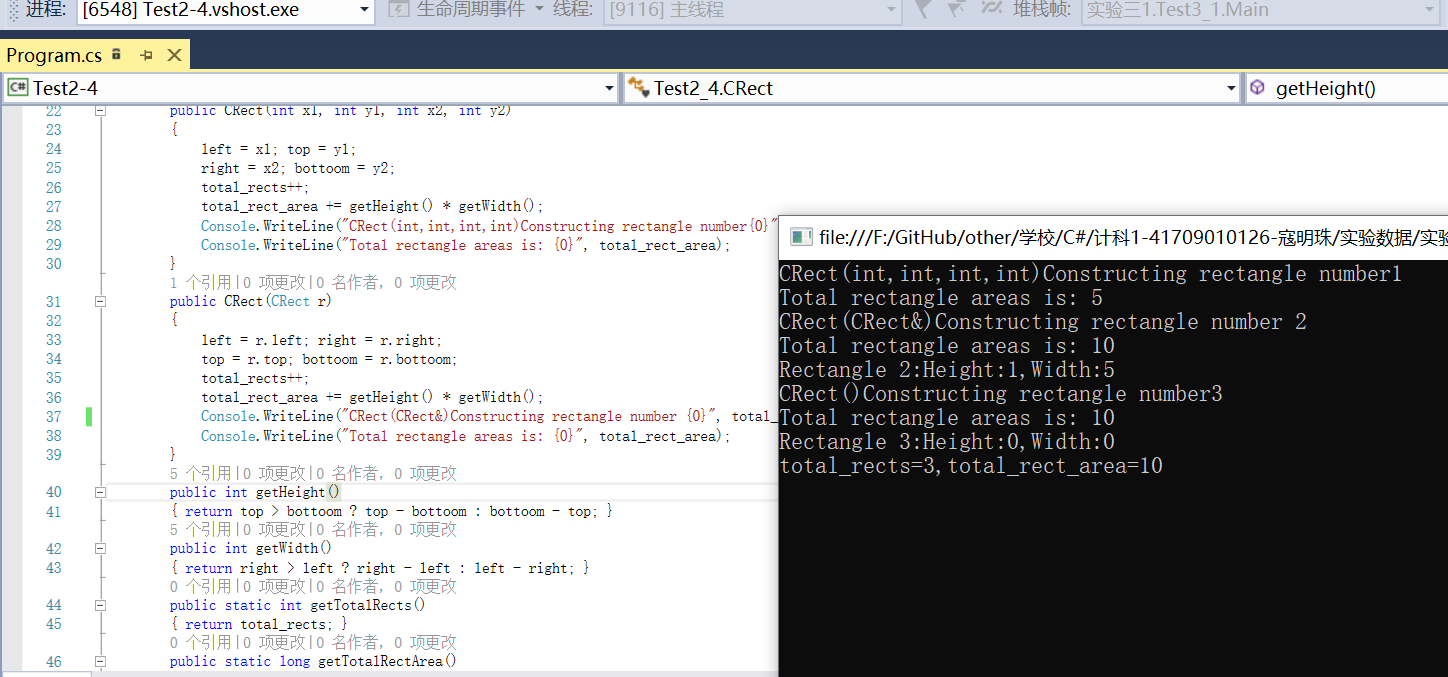
}

}

}

**【实验2-4】**

**A.跟着学习**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Test2\_4

{

class CRect

{

private int top, bottoom, left, right;

public static int total\_rects = 0;

public static long total\_rect\_area = 0;

public CRect()

{

left = top = right = bottoom = 0;

total\_rects++;

total\_rect\_area += getHeight() \* getWidth();

Console.WriteLine("CRect()Constructing rectangle number{0}", total\_rects);

Console.WriteLine("Total rectangle areas is: {0}", total\_rect\_area);

}

public CRect(int x1, int y1, int x2, int y2)

{

left = x1; top = y1;

right = x2; bottoom = y2;

total\_rects++;

total\_rect\_area += getHeight() \* getWidth();

Console.WriteLine("CRect(int,int,int,int)Constructing rectangle number{0}", total\_rects);

Console.WriteLine("Total rectangle areas is: {0}", total\_rect\_area);

}

public CRect(CRect r)

{

left = r.left; right = r.right;

top = r.top; bottoom = r.bottoom;

total\_rects++;

total\_rect\_area += getHeight() \* getWidth();

Console.WriteLine("CRect(CRect&)Constructing rectangle number{0}", total\_rects);

Console.WriteLine("Total rectangle areas is: {0}", total\_rect\_area);

}

public int getHeight()

{ return top > bottoom ? top - bottoom : bottoom - top; }

public int getWidth()

{ return right > left ? right - left : left - right; }

public static int getTotalRects()

{ return total\_rects; }

public static long getTotalRectArea()

{ return total\_rect\_area; }

}

class Test2\_4

{

static void Main(string[] args)

{

CRect rect1 = new CRect(1, 3, 6, 4), rect2 = new CRect(rect1);

Console.Write("Rectangle 2:Height:{0}", rect2.getHeight());

Console.WriteLine(",Width:{0}", rect2.getWidth());

{

CRect rect3 = new CRect();

Console.Write("Rectangle 3:Height:{0}", rect3.getHeight());

Console.WriteLine(",Width:{0}", rect3.getWidth());

}

Console.Write("total\_rects={0},", CRect.total\_rects);

Console.WriteLine("total\_rect\_area={0}", CRect.total\_rect\_area);

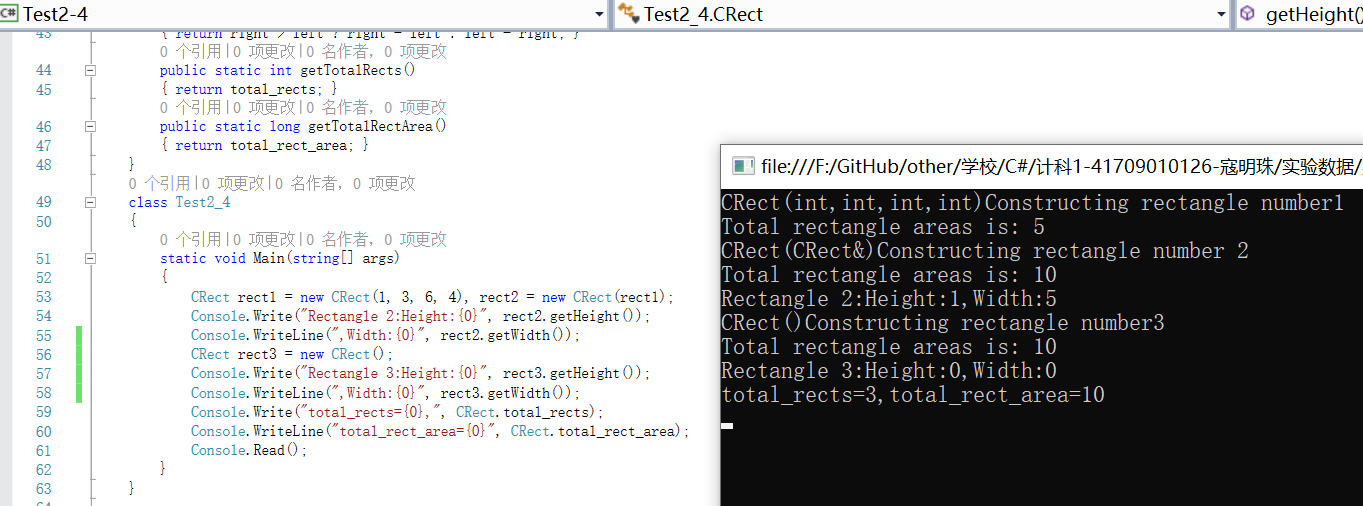
Console.Read();

}

}

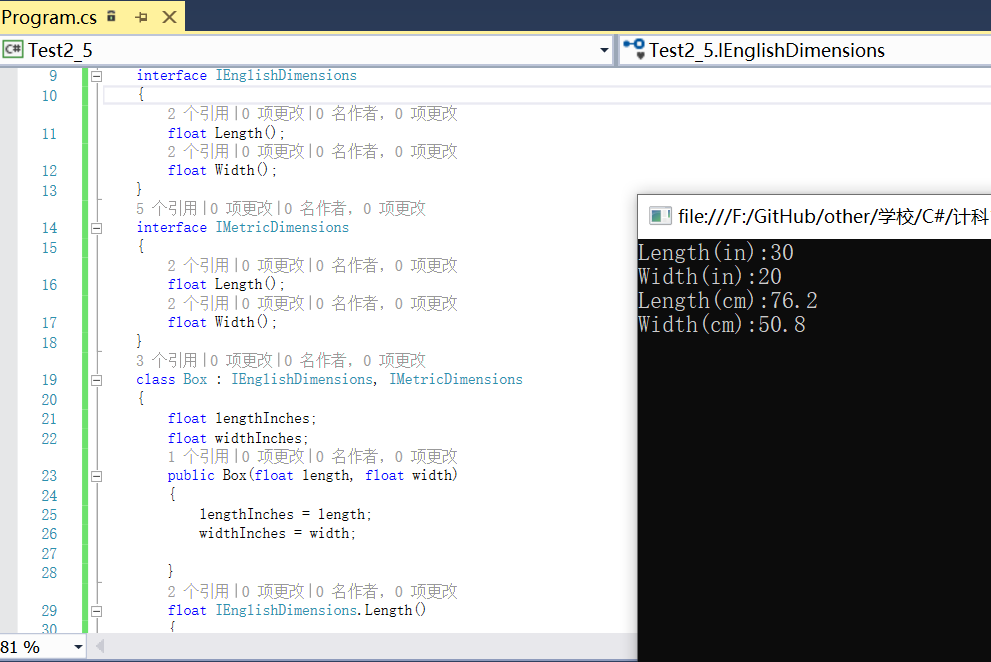
}

**B.自己思考：**注释去掉后没有变化。



**【实验2-5】**

编写IEnglishDimensions和IMetricDimensions 两个接口，同时分别以公制单位和英制单位显示框的尺寸。Box 类继承IEnglishDimensions和IMetricDimensions两个接口，它们表示不同的度量衡系统。两个接口有相同的成员名Length 和Width。



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Test2\_5

{

interface IEnglishDimensions

{

float Length();

float Width();

}

interface IMetricDimensions

{

float Length();

float Width();

}

class Box : IEnglishDimensions, IMetricDimensions

{

float lengthInches;

float widthInches;

public Box(float length, float width)

{

lengthInches = length;

widthInches = width;

}

float IEnglishDimensions.Length()

{

return lengthInches;

}

float IEnglishDimensions.Width()

{

return widthInches;

}

float IMetricDimensions.Length()

{

return lengthInches \* 2.54f;

}

float IMetricDimensions.Width()

{

return widthInches \* 2.54f;

}

}

class Program

{

static void Main(string[] args)

{

Box myBox = new Box(30.0f, 20.0f);

IEnglishDimensions eDimensions = (IEnglishDimensions)myBox;

IMetricDimensions mDimensions = (IMetricDimensions)myBox;

Console.WriteLine("Length(in):{0}", eDimensions.Length());

Console.WriteLine("Width(in):{0}", eDimensions.Width());

Console.WriteLine("Length(cm):{0}", mDimensions.Length());

Console.WriteLine("Width(cm):{0}", mDimensions.Width());

Console.Read();

}

}

}

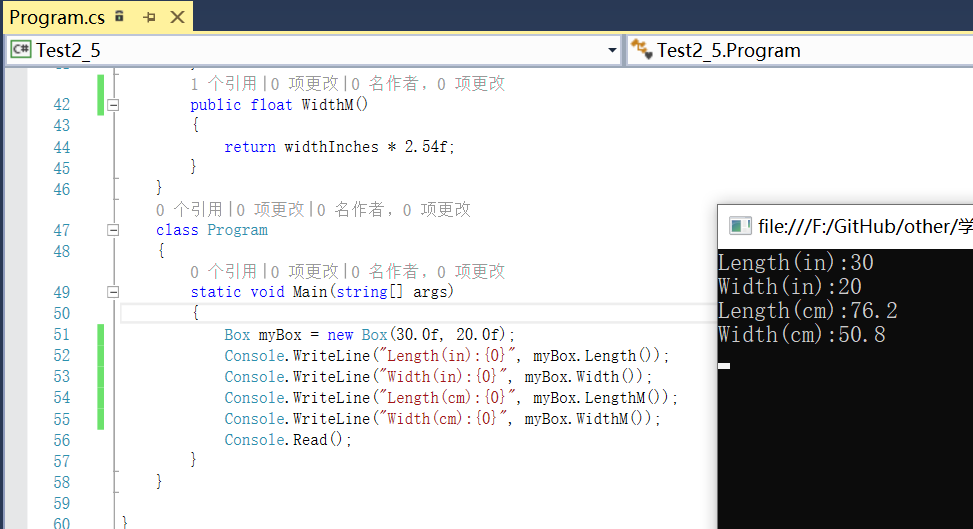
**B.自己思考：**

**比较显式接口实现和隐式接口实现的异同：**

显式：每个函数前会加上函数名，没有且不能添加权限关键字。只能通过接口访问，不能通过类的实例访问。自动添加的函数不是类的成员函数，类的实现中不能调用该函数。

隐式：每个函数与接口名称一样，没有接口名，必须添加权限关键字 public 。可以通过接口访问也可以通过类实例访问。自动添加的函数是类的程序函数，类的实现中可以调用。

隐式实现：



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Test2\_5

{

public interface IEnglishDimensions

{

float Length();

float Width();

}

public interface IMetricDimensions

{

float Length();

float Width();

}

public class Box : IEnglishDimensions, IMetricDimensions

{

float lengthInches;

float widthInches;

public Box(float length, float width)

{

lengthInches = length;

widthInches = width;

}

public float Length()//实现接口IenglishDimensions的方法

{

return lengthInches;

}

public float Width()//实现接口IenglishDimensions的方法

{

return widthInches;

}

public float LengthM()//实现接口IMetricDimensions的方法

{

return lengthInches \* 2.54f;

}

public float WidthM()//实现接口IMetricDimensions的方法

{

return widthInches \* 2.54f;

}

}

class Program

{

static void Main(string[] args)

{

Box myBox = new Box(30.0f, 20.0f);

Console.WriteLine("Length(in):{0}", myBox.Length());

Console.WriteLine("Width(in):{0}", myBox.Width());

Console.WriteLine("Length(cm):{0}", myBox.LengthM());

Console.WriteLine("Width(cm):{0}", myBox.WidthM());

Console.Read();

}

}

}