

Федеральное государственное бюджетное образовательное учреждение высшего
образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа № 5
по дисциплине «Современные технологии программирования»

Выполнил:
студент группы ИП-712
Алексеев Степан
Владимирович
ФИО студента

Работу проверил:
ассистент кафедры Агалаков А.А.
ФИО преподавателя

Новосибирск 2020 г.

Оглавление

ЗАДАНИЕ.....	2
ТЕСТОВЫЕ НАБОРЫ ДАННЫХ.....	3
ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ.....	4
ВЫВОД.....	5
ПРИЛОЖЕНИЕ.....	6
Листинг 1. TComplex.cs.....	6
Листинг 2. UnitTest1.cs	10

ЗАДАНИЕ

1. Реализовать абстрактный тип данных «комплексное число», используя класс C++, в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования.
3. Если необходимо, предусмотрите возбуждение исключительных ситуаций.

ТЕСТОВЫЕ НАБОРЫ ДАННЫХ

Для простейших функций типа взятия a или b можно ввести одно и то же число `TComplex tc = new TComplex(7, 2);` и запустить на них все функции, такие как `getRealString();` и т.п.

Для проверки правильного парсинга можно создать объект через строку `new TComplex("- 12 - i * 6");`

Для копирования использовать метод `Clone()` и сравнить значения a и b в скопированном объекте `TComplex tc = new TComplex(7, 56);`

```
TComplex tc2 = (TComplex)tc.Clone();
```

```
Assert.AreEqual(tc.getRealDouble(), tc2.getRealDouble());
```

Умножение $(5+2i)*(-3+12i)$: $5*(-3)-2*12 + i*(2*12 + -3*2) = -39 + i*18$ и дальше проверить правильность a или b .

Для возведения в степень сначала считаю в онлайн калькуляторе или на бумаге, потом прописываю:

```
TComplex tc = new TComplex(5, 8);
```

```
TComplex res = tc.power(4);
```

```
Assert.AreEqual(res.getImaginaryDouble(), -6239.99999899, 0.00001);
```

Последний параметр – дельта(требуемая точность при сравнении вещественных чисел).

ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ

```
C:\Users\stepa\repos2\STP_05_ComplexNumber\STP_05_ComplexNum...
tc.ToString() = -6-i*3
tc2.power(4) = -4879-i*6240
the i-th complex number is -0,809016994374947+i*0,587785252292473
```

Test Explorer UnitTest1.cs TComplex.cs

24 24 0

Test Du Group Summary

✔ UnitTestProject1 (24)

- ✔ UnitTestProject1 (24)
 - ✔ UnitTest1 (24)
 - ✔ TestMethod01ConstructorString
 - ✔ TestMethod02ConstructorDouble
 - ✔ TestMethod03Clone
 - ✔ TestMethod04Add
 - ✔ TestMethod05getRealDouble
 - ✔ TestMethod06getImaginaryDouble
 - ✔ TestMethod07multiply
 - ✔ TestMethod08square
 - ✔ TestMethod09reciprocal
 - ✔ TestMethod10subtractParameter_d
 - ✔ TestMethod11divideBy_d
 - ✔ TestMethod12minus
 - ✔ TestMethod12module
 - ✔ TestMethod12module2
 - ✔ TestMethod13angleRadians
 - ✔ TestMethod14angleDegrees
 - ✔ TestMethod15power
 - ✔ TestMethod16root
 - ✔ TestMethod17isEqualTo_d
 - ✔ TestMethod18isNotEqualTo_d
 - ✔ TestMethod19getRealString
 - ✔ TestMethod20getImaginaryString
 - ✔ TestMethod21ToString
 - ✔ TestMethod22ToString

UnitTestProject1
Tests in group: 24
⌚ Total Duration: 7 ms

Outcomes
✔ 24 Passed

ВЫВОД

Вспомнил о реализации комплексных чисел в виде объектов и способы проведения действий с ними. Научился брать корни, считать модуль и т.п. Изучил что такое i -й корень n -й степени комплексного числа. Узнал о новой команде `Assert.IsTrue(x)`; для метода тестирования. Вспомнил о соотношении градусов и радианов.

ПРИЛОЖЕНИЕ

Листинг 1. TComplex.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Remoting.Channels;
using System.Text;
using System.Threading.Tasks;

namespace STP_05_ComplexNumber
{
    public class TComplex : ICloneable
    {
        static void Main(string[] args)
        {
            TComplex tc = new TComplex("-6 - i*3");
            Console.WriteLine("tc.ToString() = " +
tc.ToString());
            TComplex tc2 = new TComplex(5, 8);
            TComplex res = tc2.power(4);
            Console.WriteLine("tc2.power(4) = " +
res.ToString());
            TComplex tc3 = new TComplex(1, 0);
            TComplex res2 = tc3.root(5, 2);
            Console.WriteLine("the i-th complex number is " +
res2.ToString());
            Console.ReadLine();
        }
        private double a;
        private double b;
        private char signOfb;

        public TComplex(double a, double b)
        {
            this.a = a;
            this.b = b;
        }
        public TComplex(string str)//Вызов возможен в виде
"6+i*3", "-5 + i*2"
        {
            str = str.Replace(" ", "");
            string[] stringsToAvoid = { "/", "+", "*", " ", "-i"
};
            string[] strSplit = str.Split(stringsToAvoid, 6,
StringSplitOptions.RemoveEmptyEntries);
            if (!Double.TryParse(strSplit[0], out a))
Console.WriteLine("first number is in bad shape");
            if (!Double.TryParse(strSplit[1], out b))
Console.WriteLine("second number is in bad shape");
```

```

        char[] strToChars = str.ToCharArray();
        if (str.Contains("-i")) b *= -1;
    }
    public object Clone()
    {
        return this.MemberwiseClone();
    }
    public TComplex add(TComplex d)
    {
        return new TComplex(a + d.getRealDouble(), b +
d.getImaginaryDouble());
    }
    public double getRealDouble()
    {
        return a;
    }
    public double getImaginaryDouble()
    {
        return b;
    }
    public TComplex multiply(TComplex d)
    {
        double a1 = a;
        double a2 = d.getRealDouble();
        double b1 = b;
        double b2 = d.getImaginaryDouble();
        return new TComplex(a1 * a2 - b1 * b2, a1 * b2 + a2
* b1);
    }
    public TComplex square()
    {
        return new TComplex(a * a - b * b, a * b + a * b);
    }
    public TComplex reciprocal()//Обратное Создает и
возвращает комплексное число (тип TComplex), полученное делением
единицы на само число
    {
        return new TComplex(a / (a * a + b * b), -(b / (a *
a + b * b)));
    }
    public TComplex subtractParameter_d(TComplex d)
    {
        double a1 = a;
        double a2 = d.getRealDouble();
        double b1 = b;
        double b2 = d.getImaginaryDouble();
        return new TComplex(a1 - a2, b1 - b2);
    }
    public TComplex divideBy_d(TComplex d)
    {
        double a1 = a;

```

```

        double a2 = d.getRealDouble();
        double b1 = b;
        double b2 = d.getImaginaryDouble();
        return new TComplex((a1 * a2 + b1 * b2) / (a2 * a2 +
b2 * b2), (a2 * b1 - a1 * b2) / (a2 * a2 + b2 * b2));
    }
    public TComplex minus()
    {
        return new TComplex(0 - a, 0 - b);
    }
    public double module()
    {
        return Math.Sqrt(a * a + b * b);
    }
    public double angleRadians()//Возвращает аргумент fi
самого комплексного числа q(в радианах).
    {
        //could be done like Math.Atan2(b, a); ?
        if (a > 0) return Math.Atan(b / a);
        else if (a == 0 && b > 0) return Math.PI / 2;
        else if (a < 0) return Math.Atan(b / a);
        else /*if (a == 0 && b < 0)*/ return -Math.PI / 2;
    }
    public double angleDegrees()
    {
        if (a > 0) return Math.Atan(b / a) *
57.29577951308;// 57.29577951308 - столько градусов в радиане
        else if (a == 0 && b > 0) return Math.PI / 2 *
57.29577951308;
        else if (a < 0) return Math.Atan(b / a) *
57.29577951308;
        else /*if (a == 0 && b < 0)*/ return -Math.PI / 2 *
57.29577951308;
    }

    public TComplex power(int n)//Степень. Возвращает целую
положительную степень n самого комплексного числа q.
        //q^n = r^n * (cos (n * fi)
+ i * sin (n * fi)).// https://math.semestr.ru/math/complex.php
    {
        double module = this.module();
        double modulePowered = Math.Pow(module, n);
        return new TComplex(modulePowered * Math.Cos(n *
angleRadians()), modulePowered * Math.Sin(n * angleRadians()));
    }
    public TComplex root(int n, int i)//Возвращает i-ый
корень целой положительной
        //степени n самого
комплексного числа q. sqrt_^n(q) = sqrt_^n(r) * (cos ((fi + 2*k*
pi)/n)+ i* sin((fi +2*k* pi)/n)).
        //При этом коэффициенту
k придается последовательно n значений: k = 0,1,2..., n - 1 и

```


корня, т.е.ровно столько, каков показатель корня.

//Корень n-й степени из всякого комплексного числа имеет
n различных значений. Все они имеют одинаковые модули
//<https://www.fxyz.ru/>

```
{
    double module = Math.Pow(this.module(), 1d /
n); // типа вычисляю таким образом корень модуля // получил корень
эной степени из модуля
    double phase = (angleRadians() + 2 * Math.PI * i) /
n;
    return new TComplex(module * Math.Cos(phase), module
* Math.Sin(phase));
}

public bool isEqualTo_d(TComplex d)
{
    if (a == d.getRealDouble() && b ==
d.getImaginaryDouble()) return true;
    else return false;
}
public bool isNotEqualTo_d(TComplex d)
{
    if (a != d.getRealDouble() || b !=
d.getImaginaryDouble()) return true;
    else return false;
}
public string getRealString()
{
    return a.ToString();
}
public string getImaginaryString()
{
    return b.ToString();
}
public string ToString()
{
    if (Math.Sign(b) == -1)
        return a + "-i*" + b * (-1);
    else if (Math.Sign(b) == 1)
        return a + "+i*" + b;
    else
        return a.ToString();
}
}
}
```

Листинг 2. UnitTest1.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using STP_05_ComplexNumber;
namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethod01ConstructorString()
        {
            TComplex tc = new TComplex("-12 - i * 6");
            Assert.AreEqual(tc.getImaginaryDouble(), -6);
        }
        [TestMethod]
        public void TestMethod02ConstructorDouble()
        {
            TComplex tc = new TComplex(7, 56);
            Assert.AreEqual(tc.getRealDouble(), 7);
        }
        [TestMethod]
        public void TestMethod03Clone()
        {
            TComplex tc = new TComplex(7, 56);
            TComplex tc2 = (TComplex)tc.Clone();
            Assert.AreEqual(tc.getRealDouble(),
tc2.getRealDouble());
        }
        [TestMethod]
        public void TestMethod04Add()
        {
            TComplex tc = new TComplex(7, 56);
            TComplex tc2 = new TComplex(-3, 12);
            TComplex tc3 = tc.add(tc2);
            Assert.AreEqual(tc3.getRealDouble(), 4);
        }
        [TestMethod]
        public void TestMethod05getRealDouble()
        {
            TComplex tc = new TComplex(45, 56);
            Assert.AreEqual(tc.getRealDouble(), 45);
        }
        [TestMethod]
        public void TestMethod06getImaginaryDouble()
        {
            TComplex tc = new TComplex(7, 56);
            Assert.AreEqual(tc.getImaginaryDouble(), 56);
        }
        [TestMethod]
```

```

public void TestMethod07multiply()
{
    TComplex tc = new TComplex(5, 2);
    TComplex tc2 = new TComplex(-3, 12);
    TComplex tc3 = tc.multiply(tc2); //  $5*(-3) - 2*12 + i*(2*12 + -3*2) = -39 + i*18$ 
    Assert.AreEqual(tc3.getRealDouble(), -39);
}
[TestMethod]
public void TestMethod08square()
{
    TComplex tc = new TComplex(7, 2);
    TComplex tcsq = tc.square(); //  $7*7 - 2*2 + i*(7*2 + 7*2) = 45 + i*28$ 
    Assert.AreEqual(tcsq.getImaginaryDouble(), 28);
}
[TestMethod]
public void TestMethod09reciprocal()
{
    TComplex tc = new TComplex(6, 3);
    TComplex tcrc = tc.reciprocal(); //  $6/(6^2+3^2) - i*3/(6^2+3^2) = 6/45 (=0.13333(3)) \dots$ 
    Assert.AreEqual(tcrc.getRealDouble(), 0.133333, 0.00001);
}
[TestMethod]
public void TestMethod10subtractParameter_d()
{
    TComplex tc = new TComplex(7, 56);
    TComplex tc2 = new TComplex(3, 57);
    TComplex tcsub = tc.subtractParameter_d(tc2); //
    Assert.AreEqual(tcsub.getImaginaryDouble(), -1);
}
[TestMethod]
public void TestMethod11divideBy_d()
{
    TComplex tc = new TComplex(7, 2);
    TComplex tc2 = new TComplex(3, 8);
    TComplex tcdiv = tc.divideBy_d(tc2); //
    Assert.AreEqual(tcdiv.getImaginaryDouble(), -0.6849315, 0.00001);
}
[TestMethod]
public void TestMethod12minus()
{
    TComplex tc = new TComplex(7, 2);
    TComplex tc2 = tc.minus();
    Assert.AreEqual(tc2.getImaginaryDouble(), -2);
}
[TestMethod]

```

```

public void TestMethod12module()
{
    TComplex tc = new TComplex(7, 2);
    double res = tc.module();
    Assert.AreEqual(res, 7.280109, 0.00001);
}
[TestMethod]
public void TestMethod13angleRadians()
{
    TComplex tc = new TComplex(1, 1);
    double res = tc.angleRadians();
    Assert.AreEqual(res, 0.78539816, 0.00001);
}
[TestMethod]
public void TestMethod14angleDegrees()
{
    TComplex tc = new TComplex(1, 1);
    double res = tc.angleDegrees();
    Assert.AreEqual(res, 45, 0.00001);
}
[TestMethod]
public void TestMethod12module2()
{
    TComplex tc = new TComplex(5, 8);
    double res = tc.module();
    Assert.AreEqual(res, 9.433981132056, 0.00001);
}
[TestMethod]
public void TestMethod15power()
{
    TComplex tc = new TComplex(5, 8);
    TComplex res = tc.power(4);
    Assert.AreEqual(res.getImaginaryDouble(), -
6239.99999899, 0.00001);
}
[TestMethod]
public void TestMethod16root()
{
    TComplex tc = new TComplex(1, 0);
    TComplex res = tc.root(5, 2);
    Assert.AreEqual(res.getImaginaryDouble(),
0.587785252292473, 0.00001);
}
[TestMethod]
public void TestMethod17isEqualTo_d()
{
    TComplex tc = new TComplex(7, 2);
    TComplex tc2 = new TComplex(7, 2);
    bool x = tc.isEqualTo_d(tc2);
    Assert.IsTrue(x);
}

```

```

[TestMethod]
public void TestMethod18isNotEqualTo_d()
{
    TComplex tc = new TComplex(7, 2);
    TComplex tc2 = new TComplex(6, 2);
    bool x = tc.isNotEqualTo_d(tc2);
    Assert.IsTrue(x);
}
[TestMethod]
public void TestMethod19getRealString()
{
    TComplex tc = new TComplex(7, 2);
    string str = tc.getRealString();
    Assert.AreEqual(str, "7");
}
[TestMethod]
public void TestMethod20getImaginaryString()
{
    TComplex tc = new TComplex(7, 2);
    string str = tc.getImaginaryString();
    Assert.AreEqual(str, "2");
}
[TestMethod]
public void TestMethod21ToString()
{
    TComplex tc = new TComplex(7, 2);
    string str = tc.ToString();
    Assert.AreEqual(str, "7+i*2");
}
[TestMethod]
public void TestMethod22ToString()
{
    TComplex tc = new TComplex(7, -2);
    string str = tc.ToString();
    Assert.AreEqual(str, "7-i*2");
}
}
}

```