Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации Сибирский Государственный Университет Телекоммуникаций и Информатики СибГУТИ

Кафедра прикладной математики и кибернетики

Лабораторная работа №6

по дисциплине “Современные технологии программирования”

Комплексное число

Выполнил:

Студент группы ИП-916

Меньщиков Д.А.

Работу проверил:

Агалаков А. А.

Новосибирск, 2022

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.

Выполнение лабораторной работы:

Код программы:

```
namespace lab6
{
    public class Complex
    {
        private double _real;
        private double _imag;

        public Complex(double a = 0, double b = 0)
        {
            _real = a;
            _imag = b;
        }

        public Complex(string str)
        {
            var splitter = str.Split(' ');

            _real = double.Parse(splitter[0]);
            _imag = double.Parse(splitter[1]);
        }

        static public Complex operator +(Complex rhs, Complex lhs)
        {
            return new Complex(lhs._real + rhs._real, lhs._imag + rhs._imag);
        }

        static public Complex operator -(Complex rhs, Complex lhs)
        {
            return new Complex(lhs._real - rhs._real, lhs._imag - rhs._imag);
        }

        static public Complex operator *(Complex rhs, Complex lhs)
        {
            return new Complex(lhs._real * rhs._real - lhs._imag * rhs._imag, lhs._real * rhs._imag + rhs._real * lhs._imag);
        }
```

```
42          static public Complex operator /(Complex rhs, Complex lhs)
43          {
44              double denomimator = rhs._real * rhs._real + rhs._imag * rhs._imag;
45              double nominatorLeft = lhs._real * rhs._real + lhs._imag * rhs._imag;
46              double nominatorRight = rhs._real * lhs._real - lhs._imag * rhs._imag;
47
48              return new Complex(nominatorLeft / denomimator, nominatorRight / denomimator);
49          }
50
51          static public bool operator ==(Complex lhs, Complex rhs)
52          {
53              return lhs._real == rhs._real && lhs._imag == rhs._imag;
54          }
55
56          static public bool operator !=(Complex lhs, Complex rhs)
57          {
58              return lhs._real != rhs._real || lhs._imag != rhs._imag;
59          }
60
61          public Complex Pow(int n = 2)
62          {
63              double phi = Math.Atan2(_imag, _real);
64              double r = Math.Sqrt(_real * _real + _imag * _imag);
65
66              double R = Math.Pow(r, n);
67              double Phi = n * phi;
68
69              double X = Math.Round(R * Math.Cos(Phi));
70              double Y = Math.Round(R * Math.Sin(Phi));
71
72              Complex complex = new Complex(X, Y);
73
74              return complex;
75          }
76
77          public double Abs()
78          {
79              return Math.Sqrt(_real * _real + _imag * _imag);
80          }
```

```
        public double AngleRadians()
        {
            if (_real > 0)
            {
                return Math.Atan(_imag / _real);
            }
            else if (_real < 0)
            {
                return Math.Atan(_imag / _real) + Math.PI;
            }
            else if (_real == 0 && _imag < 0)
            {
                return -Math.PI / 2;
            }
            else if (_real == 0 && _imag > 0)
            {
                return Math.PI / 2;
            }
            else
            {
                throw new Exception("Can't take a angle");
            }
        }

        public Complex Root(int n, int i)
        {
            List<Complex> roots = new List<Complex>();
            double phi = Math.Pow(Abs(), 1 / (double)n);
            for (int k = 0; k < n; k++)
            {
                double coeff = 2 * Math.PI * k;
                //roots[k] = new Complex(phi, 0) * new Complex(Math.Cos((AngleRadians() + coeff) / n), Math.Sin((AngleRadians() + coeff) / n));
                roots.Add(new Complex(phi, 0) * new Complex(Math.Cos((AngleRadians() + coeff) / n), Math.Sin((AngleRadians() + coeff) / n)));
            }

            return roots[i];
        }

        public double GetReal()
        {
            return _real;
        }
```

```
124
125         public double GetImag()
126         {
127             return _imag;
128         }
129
130         public string RealString()
131         {
132             return _real.ToString();
133         }
134
135         public string ImagString()
136         {
137             return _imag.ToString();
138         }
139
140         public string ToString()
141         {
142             string img = _imag >= 0 ? ImagString() : "(" + ImagString() + ")";
143             return RealString() + "+i" + img;
144         }
145
146         public void Show()
147         {
148             Console.WriteLine($"real: {_real}, imag: {_imag}");
149         }
150     }
151 }
```

Модульные тесты:

```
4   namespace TestComplexClass
5   {
6       [TestClass]
7       public class UnitTest1
8       {
9           [TestMethod]
10          public void TestConstructor()
11          {
12              var _ = new Complex(1, 5);
13          }
14
15          [TestMethod]
16          public void TestConstructorZero()
17          {
18              var _ = new Complex(0, 0);
19          }
20
21          [TestMethod]
22          public void TestConstructorUnderZero()
23          {
24              var _ = new Complex(-1, -5);
25          }
26
27          [TestMethod]
28          public void TestSum()
29          {
30              var first = new Complex(1, 5);
31              var second = new Complex(2, 3);
32
33              var actual = first + second;
34              var expected = new Complex(3, 8);
35
36              Assert.IsTrue(expected == actual);
37
38          }
```

```
41          [TestMethod]
42          public void TestSumWithUnderZeroOneVars()
43          {
44              var first = new Complex(2, 5);
45              var second = new Complex(-1, -5);
46
47              var actual = first + second;
48              var expected = new Complex(1, 0);
49
50              Assert.IsTrue(expected == actual);
51          }
52
53          [TestMethod]
54          public void TestSumWithUnderZeroTwoVars()
55          {
56              var first = new Complex(-2, -5);
57              var second = new Complex(-1, -5);
58
59              var actual = first + second;
60              var expected = new Complex(-3, -10);
61
62              Assert.IsTrue(expected == actual);
63          }
64
65          [TestMethod]
66          public void TestMinus()
67          {
68              var first = new Complex(2, 5);
69              var second = new Complex(1, 5);
70
71              var actual = first - second;
72              var expected = new Complex(-1, 0);
73
74              Assert.IsTrue(expected == actual);
75          }
```

```
77          [TestMethod]
78          public void TestMinusWithUnderZeroOneVars()
79          {
80              var first = new Complex(2, 5);
81              var second = new Complex(-1, -5);
82
83              var actual = first - second;
84              var expected = new Complex(-3, -10);
85
86              Assert.IsTrue(expected == actual);
87          }
88
89          [TestMethod]
90          public void TestMinusWithUnderZeroTwoVars()
91          {
92              var first = new Complex(-2, -5);
93              var second = new Complex(-1, -5);
94
95              var actual = first - second;
96              var expected = new Complex(1, 0);
97
98              Assert.IsTrue(expected == actual);
99          }
100
101         [TestMethod]
102         public void TestMultiply()
103         {
104             var first = new Complex(2, 5);
105             var second = new Complex(1, 5);
106
107             var actual = first * second;
108             var expected = new Complex(-23, 15);
109
110             Assert.IsTrue(expected == actual);
111         }
```

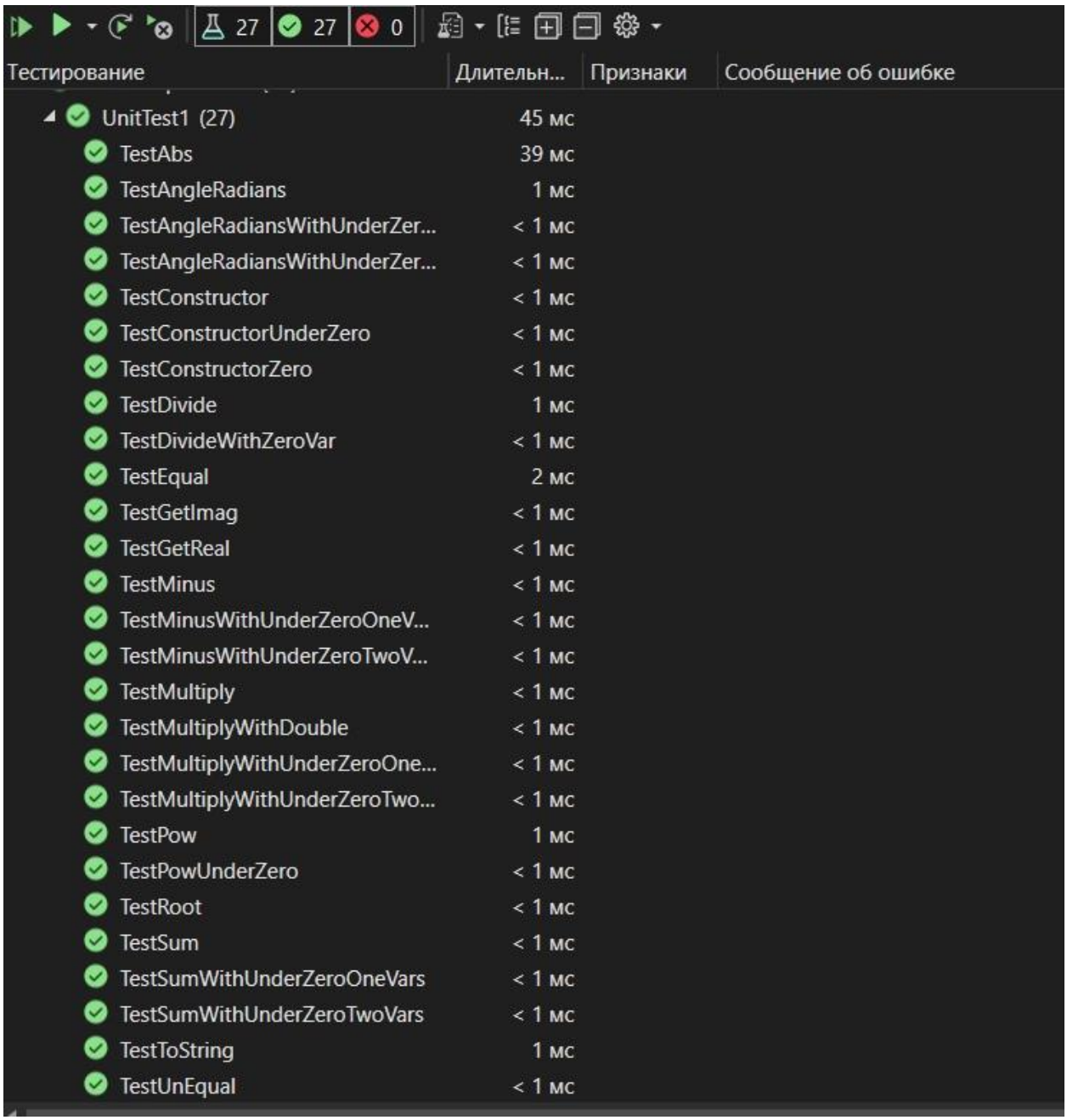
```
113         [TestMethod]
114         public void TestMultiplyWithUnderZeroOneVars()
115         {
116             var first = new Complex(2, 5);
117             var second = new Complex(-1, -5);
118 
119             var actual = first * second;
120             var expected = new Complex(23, -15);
121 
122             Assert.IsTrue(expected == actual);
123         }
124 
125         [TestMethod]
126         public void TestMultiplyWithUnderZeroTwoVars()
127         {
128             var first = new Complex(-2, -5);
129             var second = new Complex(-1, -5);
130 
131             var actual = first * second;
132             var expected = new Complex(-23, 15);
133 
134             Assert.IsTrue(expected == actual);
135         }
136 
137         [TestMethod]
138         public void TestMultiplyWithDouble()
139         {
140             var first = new Complex(0.12, 0.44);
141             var second = new Complex(0.76, 0.424);
142 
143             var actual = first * second;
144             var expected = new Complex(-0.09536, 0.38528);
145 
146             Assert.IsTrue(expected == actual);
147         }
```

```
149         [TestMethod]
150         public void TestDivide()
151         {
152             var first = new Complex(32, 16);
153             var second = new Complex(2, 16);
154 
155             var actual = first / second;
156             var expected = new Complex(0.25, -0.15);
157 
158             Assert.IsTrue(expected == actual);
159         }
160 
161         [TestMethod]
162         public void TestDivideWithZeroVar()
163         {
164             var first = new Complex(-32, 16);
165             var second = new Complex(-2, 16);
166 
167             var actual = first / second;
168             var expected = new Complex(0.25, -0.15);
169 
170             Assert.IsTrue(expected == actual);
171         }
172 
173         [TestMethod]
174         public void TestEqual()
175         {
176             var first = new Complex(32, 16);
177             var second = new Complex(32, 16);
178 
179             var actual = first * second;
180             var expected = new Complex(0.25, 0.15);
181 
182             Assert.IsTrue(first == second);
183         }
```

```
185         [TestMethod]
186         public void TestUnEqual()
187         {
188             var first = new Complex(32, 16);
189             var second = new Complex(132, 16);
190 
191             var actual = first * second;
192             var expected = new Complex(0.25, 0.15);
193 
194             Assert.IsTrue(first != second);
195         }
196 
197         [TestMethod]
198         public void TestPow()
199         {
200             var first = new Complex(32, 16);
201 
202             var actual = first.Pow(2);
203             var expected = new Complex(768, 1024);
204 
205             Assert.IsTrue(expected == actual);
206         }
207 
208         [TestMethod]
209         public void TestPowUnderZero()
210         {
211             var first = new Complex(-32, 16);
212 
213             var actual = first.Pow(2);
214             var expected = new Complex(768, -1024);
215 
216             Assert.IsTrue(expected == actual);
217         }
```

```
219         [TestMethod]
220         public void TestAbs()
221         {
222             var first = new Complex(2, 2);
223 
224             var expected = 2.8284271247461903;
225             var actual = first.Abs();
226 
227             Assert.IsTrue(actual == expected);
228         }
229 
230         [TestMethod]
231         public void TestAngleRadians()
232         {
233             var first = new Complex(2, 2);
234 
235             var expected = 0.78539816339744828;
236             var actual = first.AngleRadians();
237 
238             Assert.IsTrue(actual == expected);
239         }
240 
241         [TestMethod]
242         public void TestAngleRadiansWithUnderZeroOneVars()
243         {
244             var first = new Complex(-2, 2);
245 
246             var expected = 2.356194490192345;
247             var actual = first.AngleRadians();
248 
249             Assert.IsTrue(actual == expected);
250         }
251 
252         [TestMethod]
253         public void TestAngleRadiansWithUnderZeroTwoVars()
254         {
255             var first = new Complex(-2, -2);
256 
257             var expected = 3.9269908169872414;
258             var actual = first.AngleRadians();
259 
260             Assert.IsTrue(actual == expected);
261         }
```

```
264         [TestMethod]
265         public void TestGetReal()
266         {
267             var first = new Complex(2, 2);
268 
269             var expected = 2;
270             var actual = first.GetReal();
271 
272             Assert.IsTrue(actual == expected);
273         }
274 
275         [TestMethod]
276         public void TestGetImag()
277         {
278             var first = new Complex(2, 4);
279 
280             var expected = 4;
281             var actual = first.GetImag();
282 
283             Assert.IsTrue(actual == expected);
284         }
285 
286         [TestMethod]
287         public void TestToString()
288         {
289             var first = new Complex(2, 2);
290 
291             var expected = "2+i2";
292             var actual = first.ToString();
293 
294             Assert.IsTrue(actual == expected);
295         }
296 
297         [TestMethod]
298         public void TestRoot()
299         {
300             var first = new Complex(3840, 2048);
301 
302             var expected = new Complex(64, 16);
303             var actual = first.Root(2, 0);
304 
305             Assert.IsTrue(actual == expected);
306         }
307     }
308 }
```


27
27
0
Тестирование
Длительн...
Признаки
Сообщение об ошибке
UnitTest1 (27) 45 мс
TestAbs 39 мс
TestAngleRadians 1 мс
TestAngleRadiansWithUnderZer... < 1 мс
TestAngleRadiansWithUnderZer... < 1 мс
TestConstructor < 1 мс
TestConstructorUnderZero < 1 мс
TestConstructorZero < 1 мс
TestDivide 1 мс
TestDivideWithZeroVar < 1 мс
TestEqual 2 мс
TestGetImag < 1 мс
TestGetReal < 1 мс
TestMinus < 1 мс
TestMinusWithUnderZeroOneV... < 1 мс
TestMinusWithUnderZeroTwoV... < 1 мс
TestMultiply < 1 мс
TestMultiplyWithDouble < 1 мс
TestMultiplyWithUnderZeroOne... < 1 мс
TestMultiplyWithUnderZeroTwo... < 1 мс
TestPow 1 мс
TestPowUnderZero < 1 мс
TestRoot < 1 мс
TestSum < 1 мс
TestSumWithUnderZeroOneVars < 1 мс
TestSumWithUnderZeroTwoVars < 1 мс
TestToString 1 мс
TestUnEqual < 1 мс