```
using System;
using System.Threading;

//Полином 17[П] + 5[Д]i+(1[А]+10[И]i)z+(3[В]+14[М]i)z^2+13[Л]z^3
namespace PAVLOV_DMITRY_206
{
    struct Complex//структура для работы с комплексными числами
    {
        public double X, Y;

        public Complex(double x, double y)
        {
            X = x;
            Y = y;
        }

        public static Complex operator -(Complex a)
        {
            a.X = -a.X;
            a.Y = -a.Y;
            return a;
        }

        public static Complex operator -(Complex a, Complex b)
        {
            a.X -= b.X;
            a.Y -= b.Y;
            return a;
        }

        public static Complex operator +(Complex a, Complex b)
        {
            a.X += b.X;
            a.Y += b.Y;
            return a;
        }

        public static Complex operator *(Complex a, Complex b)
        {
            return new Complex(a.X * b.X - a.Y * b.Y, a.X * b.Y + b.X *
              a.Y);
        }

        public static Complex operator *(double b, Complex a)
        {
            a.X *= b;
            a.Y *= b;
            return a;
        }

        public static Complex operator /(Complex a, Complex b)
        {
            a.X = (a.X * b.X - a.Y * b.Y) / (b.X * b.X + b.Y * b.Y);
```

```
53              a.Y = (a.Y * b.X + a.X * b.Y) / (b.X * b.X + b.Y * b.Y);
54              return a;
55          }
56
57          public static Complex operator /(double b, Complex a)
58          {
59              a.X /= b;
60              a.Y /= b;
61              return a;
62          }
63
64          public static Complex Power(Complex a, int power)
65          {
66              if (power == 0)
67              {
68                  a.X = 1;
69                  a.Y = 0;
70                  return a;
71              }
72              Complex b = a;
73              for (int i = 2; i <= power; i++)
74              {
75                  b = b * a;
76              }
77              return b;
78          }
79          public static Complex Value(Vector X)
80          {
81              return new Complex(X.X, X.Y);
82          }
83
84      }
85
86      struct Vector
87      {
88          public double X, Y;
89
90          public Vector(double x, double y)
91          {
92              X = x;
93              Y = y;
94          }
95
96          public static Vector operator -(Vector a, Vector b)
97          {
98              a.X -= b.X;
99              a.Y -= b.Y;
100             return a;
101         }
102
103         public static Vector operator +(Vector a, Vector b)
104         {
105             a.X += b.X;
```

```
106             a.Y += b.Y;
107             return a;
108         }
109
110         public static Vector operator *(double b, Vector a)
111         {
112             a.X *= b;
113             a.Y *= b;
114             return a;
115         }
116
117         public static Vector toVector(Complex X)
118         {
119             return new Vector(X.X,X.Y);
120         }
121
122         public static double Norm(Vector a)
123         {
124             return Math.Sqrt(a.X * a.X + a.Y * a.Y);
125         }
126
127     }
128
129     static class Polynom
130     {
131         //Коээфициенты полинома 17[П] + 5[Д]i+(1[А]+10[И]i)z+(3[В]+14[М]i)
              z^2+13[Л]z^3
132         static public Complex[] Coef = new Complex[4] { new Complex(17, 5),
              new Complex(1, 10), new Complex(3, 14), new Complex(13, 0) };
133
134        /* static public Complex Poly(Complex[] coef, Complex X)// вычисление
               полинома
135         {
136             Complex z = coef[0] + coef[1] * X + coef[2] * Complex.Power(X,
                  2) + coef[3] * Complex.Power(X, 3);
137             return z;
138         }*/
139
140         static public double getRe(Complex[] coef, Vector X)//вычисление
              действительной части полинома
141         {
142             return coef[0].X + coef[1].X * X.X - coef[1].Y * X.Y + coef[2].X
                    * Math.Pow(X.X, 2) - 2 * coef[2].Y * X.X * X.Y -
143                     coef[2].X * Math.Pow(X.Y, 2) + coef[3].X * Math.Pow(X.X,
                        3) - 3 * coef[3].Y * Math.Pow(X.X, 2) * X.Y -
144                     3 * coef[3].X * X.X * Math.Pow(X.Y, 2) + coef[3].Y *
                        Math.Pow(X.Y, 3);
145         }
146         static public double getRe_dx(Complex[] coef, Vector X)
147         {
148             return coef[1].X + 2 * coef[2].X * X.X - 2 * coef[2].Y * X.Y + 3
                    * coef[3].X * Math.Pow(X.X, 2) -
149                     6 * coef[3].Y * X.X * X.Y - 3 * coef[3].X * Math.Pow(X.Y,
```

```
                    2);
150         }
151         static public double getRe_dy(Complex[] coef, Vector X)
152         {
153             return -coef[1].Y - 2 * coef[2].Y * X.X - 2 * coef[2].X * X.Y -
                  3 * coef[3].Y * Math.Pow(X.X, 2) -
154                 6 * coef[3].X * X.X * X.Y + 3 * coef[3].Y * Math.Pow(X.Y,
                    2);
155         }
156
157         static public double getIm(Complex[] coef, Vector X)//вычисление
              мнимой части полинома
158         {
159             return coef[0].Y + X.X * coef[1].Y + coef[1].X * X.Y + coef[2].Y
                  * Math.Pow(X.X, 2) + 2 * coef[2].X * X.X * X.Y -
160                 coef[2].Y * Math.Pow(X.Y, 2) + coef[3].Y * Math.Pow(X.X,
                    3) + 3 * coef[3].X * Math.Pow(X.X, 2) * X.Y -
161                 3 * coef[3].Y * X.X * Math.Pow(X.Y, 2) - coef[3].X *
                    Math.Pow(X.Y, 3);
162         }
163         static public double getIm_dx(Complex[] coef, Vector X)//вычисление
              производной от мнимой части полинома по x
164         {
165             return coef[1].Y + 2 * coef[2].Y * X.X + 2 * coef[2].X * X.Y + 3
                  * coef[3].Y * Math.Pow(X.X, 2) +
166                 6 * coef[3].X * X.X * X.Y - 3 * coef[3].Y * Math.Pow(X.Y,
                    2);
167         }
168         static public double getIm_dy(Complex[] coef, Vector X)//вычисление
              производной от мнимой части полинома по y
169         {
170             return coef[1].X + 2 * coef[2].X * X.X - 2 * coef[2].Y * X.Y + 3
                  * coef[3].X * Math.Pow(X.X, 2) -
171                 6 * coef[3].Y * X.X * X.Y - 3 * coef[3].X * Math.Pow(X.Y,
                    2);
172         }
173
174         static public double Funct(Complex[] coef, Vector X)//вычисление
              функции |Polynom|^2
175         {
176             return Math.Pow(getRe(coef, X), 2) + Math.Pow(getIm(coef, X),
                  2);
177         }
178
179         static public Vector Grad(Complex[] coef, Vector X)//вычисление
              градиента от функции
180         {
181             return new Vector(2 * getRe(coef, X) * getRe_dx(coef, X) + 2 *
                  getIm(coef, X) * getIm_dx(coef, X),
182                 2 * getRe(coef, X) * getRe_dy(coef, X) + 2 * getIm(coef, X)
                    * getIm_dy(coef, X));
183         }
184
```

```
185         static public Complex[] Horner_sMethod(Complex[] oldCoef, int power,
               Complex root) //Метод Горнера
186         {
187             Complex[] newCoef = new Complex[4];
188             if (power == 3)
189             {
190                 newCoef[3] = new Complex(0, 0);
191                 newCoef[2] = oldCoef[3];
192                 newCoef[1] = oldCoef[2] + root * newCoef[2];
193                 newCoef[0] = oldCoef[1] + root * newCoef[1];
194             }
195             else if (power == 2)
196             {
197                 newCoef[3] = new Complex(0, 0);
198                 newCoef[2] = new Complex(0, 0);
199                 newCoef[1] = oldCoef[2];
200                 newCoef[0] = oldCoef[1] + root * newCoef[1];
201             }
202             return newCoef;
203         }
204     }
205     class Program
206     {
207         public const double Eps = 1e-6;
208         static Complex[] coef = new Complex[4];
209         static Complex[] roots = new Complex[3];
210         static void CrushingGradient()//градиентный метод с дроблением шага
211         {
212             double a = 1, lambda = 0.95, delta = 0.5;
213             for (int i = 0; i < 4; i++)
214             {
215                 coef = Polynom.Coef;
216             }
217             Vector G= new Vector();
218             Vector X_k = new Vector();
219             int count = 0;
220             for (int i = 0; i < 2; i++)
221             {
222                 Vector X_i = new Vector(0, 0);//начальная точка
223                 G = Polynom.Grad(coef, X_i);
224                 while (Vector.Norm(G) > Eps)
225                 {
226                     X_k = X_i - a * G;
227                     if (count%4==0)Console.WriteLine("x={0:00.00000000},y=
                       {1:00.00000000}={2:00.00000000}", X_k.X, X_k.Y,
                       Polynom.Funct(coef, Vector.toVector(Complex.Value
                       (X_k))));
228                     count++;
229                     if (Polynom.Funct(coef, X_k) - Polynom.Funct(coef, X_i)
                       <= -a * delta * (Math.Pow(G.X, 2) + Math.Pow(G.Y, 2)))
230                     {
231                         X_i = X_k;
232                         G = Polynom.Grad(coef, X_k);
```

```
233                     }
234                     else a = a * lambda;
235                 }
236                 roots[i] = Complex.Value(X_k);
237                 Console.WriteLine
                      ("_____________________________________________");
238                 Console.WriteLine("{3} КОРЕНЬ: {0:00.00000000}+i*
                      {1:00.00000000}={2:00.00000000}", roots[i].X, roots[i].Y,
                      Polynom.Funct(coef, Vector.toVector(roots[i])),i+1);
239                 Console.WriteLine
                      ("_____________________________________________");
240                 coef = Polynom.Horner_sMethod(coef, 3 - i, roots[i]);
241             }
242             roots[2] = -coef[0] / coef[1];
243             Console.WriteLine
                  ("_____________________________________________");
244             Console.WriteLine("{3} КОРЕНЬ: {0:00.00000000}+i*{1:00.00000000}
                  ={2:00.00000000}", roots[2].X, roots[2].Y, Polynom.Funct(coef,
                   Vector.toVector(roots[2])),3);
245             Console.WriteLine
                  ("_____________________________________________");
246         }
247         static void Main()
248         {
249             CrushingGradient();
250             Console.ReadKey();
251         }
252     }
253 }
254
```