```
using System;
//[Д]+[И]*x+[М]*x^2+[А]*x^3+[П]+[А]*y+[В]*y^2+[Л]*y^3+[О]*y^4+[В]*y^5+[Л]+[Ь]*xy+[В]*(xy)^2+[О]*(xy)^3

//f(x)=5+10*x+14*x^2+x^3
//f(y)=17+y+3*y^2+13*y^3+16*y^4+3*y^5
//f(x,y)=13+30*xy+3*(xy)^2+16*(xy)^3
//z=f(x)+f(y)+f(x,y)=5+10*x+14*x^2+x^3+17+y+3*y^2+13*y^3+16*y^4+3*y^5+13+30*xy+3*(xy)^2+16*(xy)^3
//g(z)=x^2+y^2-1<=0
//h(z)=x+y<=0
namespace Задача_3
{
    class Point
    {
        public double X;
        public double Y;

        public Point(double x, double y)//конструктор
        {
            X = x;
            Y = y;
        }
        public static Point operator +(Point a, Point b)
        {
            return new Point(a.X + b.X, a.Y + b.Y);
        }
        public static Point operator -(Point a, Point b)
        {
            return new Point(a.X - b.X, a.Y - b.Y);
        }
        public static Point operator *(double a, Point b)
        {
            return new Point(a * b.X, a * b.Y);
        }
    }

    class Poly//весь полином P
    {
        public int x_power, y_power;
        public double[,] coef;

        public Poly(double[,] Coef)//конструктор полинома
        {
            coef = Coef;
            x_power = coef.GetLength(0);//максимальная степень x
            y_power = coef.GetLength(1);//максимальная степень y
        }

        public double Value(Point a)//вычисление значения полинома в данной точке
        {
            double c = 0;
```

```
51              for (int i = 0; i < x_power; i++)
52              {
53                  for (int j = 0; j < y_power; j++)
54                  {
55                      c = c + coef[i, j] * Math.Pow(a.X, i) * Math.Pow(a.Y,
                        j);
56                  }
57              }
58              return c;
59          }
60
61          public Point Grad(Point a)//градиент от полинома
62          {
63              Point c = new Point(0, 0);
64              for (int i = 0; i < x_power; i++)
65              {
66                  for (int j = 0; j < y_power; j++)
67                  {
68                      if (i!=x_power-1)
69                      {
70                          c.X = c.X + coef[i + 1, j] * (i + 1) * Math.Pow(a.X,
                        i) * Math.Pow(a.Y, j);
71                      }
72                      if (j!=y_power-1)
73                      {
74                          c.Y = c.Y + coef[i, j + 1] * (j + 1) * Math.Pow(a.X,
                        i) * Math.Pow(a.Y, j);
75                      }
76                  }
77              }
78              return c;
79          }
80      }
81
82      class Program
83      {
84          private const double Eps = 1e-8;
85          static double[,] PolyCoef =//задание коеффициентов
86          {   /*x*/
87             /*y*/{35,  1, 18,  1, 12, 16, 3, 1},//по горизонтали - фамилия
88              {10, 30,  0,  0,  0,  0, 0, 0},//по вертикали - имя
89              {14,  0,  3,  0,  0,  0, 0, 0},//по диагонали - отчество
90              { 1,  0,  0, 16,  0,  0, 0, 0}
91          };
92          static Poly Polynom=new Poly(PolyCoef);//создание экземпляра класса
              для работы с ним
93          static double[] Condition(Point a)//1 ограничение, 2 ограничение
94          {
95              return new[] {a.X*a.X+a.Y*a.Y-1,a.X+a.Y};
96          }
97          static Point[] GradCondition(Point a)//градиент 1 ограничения,
              градиент 2 ограничения
98          {
```

```
 99              return new[] {new Point(2 * a.X, 2 * a.Y), new Point(1, 1)};
100          }
101
102          static double Penalty(Point a)//вычисление H
103          {
104              double c = 0;
105              double penalty_sum=0;//Значение H
106              for (int i = 0; i < 2; i++)
107              {
108                  double check=Condition(a)[i]; //проверка условий g,h
109                  if (check > 0)
110                  {
111                      penalty_sum += Math.Pow(check, 2);
112                  }
113              }
114              return penalty_sum;
115          }
116
117          static double GetPointValue(Point a,double r)//Значение f(x,y)+r*H
118          {
119              return Polynom.Value(a)+r * Penalty(a);
120          }
121
122          static void PenaltyCalculate()
123          {
124              Point currentPoint = new Point(1, 1);//выбор начальной точки
125              double r = 1e6;//выбор r
126              double value_CP = GetPointValue(currentPoint, r); //подсчет
                   значения в 1 точке
127              double value_PP = value_CP;//предыдущая точка = текущей
128              int super_iter = 1;
129              do//вычисления с одним r(k)
130              {
131                  Console.WriteLine("Супер итерация: {0}.\n", super_iter);
132                  super_iter++;
133                  double dx = 0.5;//шаг по x
134                  double dy = 0.5;//шаг по y
135                  double epsilon = Eps*100;//точность для безусловной
                       минимизации
136                  int iter = 1;
137                  double[] constrains = Condition(currentPoint);//вывод
                       ограничений
138                  Console.WriteLine("{0})  X={1:0.00000}, Y={2:0.00000} |
                       {3:0.00000} | {4:E2}  {5:E2} | {6:0.00000} | {7:E2}
                       {8:E2}", iter, currentPoint.X,currentPoint.Y, value_CP,
                       dx, dy, Polynom.Value(currentPoint), constrains[0],
                       constrains[1]);
139                  iter++;
140                  //безусловная минимизация методом покоординатного спуска
141                  do
142                  {
143                      value_PP = value_CP;//предыдущая точка = текущей
144                      Point nextPoint;//следующая точка
```

```
145                     double value_NP;//значение сл.т.
146                     //по x
147                     do
148                     {
149                         nextPoint = currentPoint + new Point(dx, 0);//x + dx
150                         value_NP = GetPointValue(nextPoint, r);//Значение в x + dx
151                         if (value_NP > value_CP)
152                         {
153                             nextPoint = currentPoint - new Point(dx, 0);
154                             value_NP = GetPointValue(nextPoint, r);
155                             if (value_NP > value_CP)//если значение следующей точки меньше предыдущего изменяем направление движения
156                                 dx *= 0.5;
157                             else
158                                 dx *= -1;
159                         }
160                     } while (value_NP > value_CP);//пока значение следующей точки больше предыдущей
161                     while (value_NP < value_CP)//пока значение текущей точки меньше следующей
162                     {
163                         currentPoint = nextPoint;
164                         value_CP = value_NP;
165                         nextPoint = currentPoint + new Point(dx, 0);
166                         value_NP = GetPointValue(nextPoint, r);
167                     }
168                     //по y
169                     do
170                     {
171                         nextPoint = currentPoint + new Point(0, dy);
172                         value_NP = GetPointValue(nextPoint, r);
173                         if (value_NP > value_CP)
174                         {
175                             nextPoint = currentPoint - new Point(0, dy);
176                             value_NP = GetPointValue(nextPoint, r);
177                             if (value_NP > value_CP)
178                                 dy *= 0.5;
179                             else
180                                 dy *= -1;
181                         }
182                     } while (value_NP > value_CP);
183                     while (value_NP < value_CP)
184                     {
185                         currentPoint = nextPoint;
186                         value_CP = value_NP;
187                         nextPoint = currentPoint + new Point(0, dy);
188                         value_NP = GetPointValue(nextPoint, r);
189                     }
190                     Console.WriteLine("{0})  X={1:0.00000}, Y={2:0.00000} | {3:0.00000} | {4:E2}  {5:E2} | {6:0.00000} | {7:E2}  {8:E2}", iter, currentPoint.X, currentPoint.Y, value_CP,
```

```
                    dx, dy, Polynom.Value(currentPoint), constrains[0],
                    constrains[1]);
191                  iter++;
192                } while ((dx >= epsilon) || (dy >= epsilon) || (value_PP -
                   value_CP >= epsilon));
193                //конец безусловной минимизации
194
195                r *= 10;//r(k+1)=10*r(k)
196            } while (Penalty(currentPoint) > Eps);//проверка EPS
197            Console.WriteLine("Point X={0},Y=
                 {1}",currentPoint.X,currentPoint.Y);
198        }
199        static void Main(string[] args)
200        {
201            Console.WriteLine("Iter) Optimum | f(x,y)+r*H(x,y) | step_X
                 step_Y | f(x) | g1(x)  g2(x)\n");
202            PenaltyCalculate();
203            Console.ReadKey();
204        }
205    }
206 }
207
```