ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

|  |  |
| --- | --- |
| СОГЛАСОВАНО  заведующий отделением  Программной инженерии,  факультета Бизнес-информатики,  профессор кафедры УРПО  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Авдошин С.М.  «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 г. | УТВЕРЖДАЮ  заведующий отделением  Программной инженерии,  факультета Бизнес-информатики,  профессор кафедры УРПО  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Авдошин С.М.  «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 г. |

**программа оптимизации, Инспирированная поведением лягушек**

**Текст программы**

**Лист утверждения**

**А.В.00001-01 ТЗ 01**

**RU.17701729.503200-01 12 01-1**

Инв. № подл.

Подпись и дата

Взам. инв. №

Инв. № дубл.

Подпись и дата

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | | |
|  |  | | | \_ |
|  |  |  | | |
|  |  |  | | |
|  |  | Исполнитель: студент группы 171ПИ | | |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_/Ремнев Н.В./ | | |
|  |  | | “\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_2013 г. | |
|  |  |  | | |
|  |  |  | | |

**УТВЕРЖДЕНО**

RU.17701729.503200-01 12 01-1

**программа оптимизации, Инспирированная поведением лягушек**

**Текст программы**

**RU.17701729.503200-01 12 01-1**

Инв. № подл.

Подпись и дата

Взам. инв. №

Инв. № дубл.

Подпись и дата

# Содержание

1. Код файла Functions.cs 3
2. Код файла OptimizationAlgorithms.cs 4
3. Код файла FunctionGraph.cs 11
4. Код файла MainForm.cs 14
5. Код файла Graphiks.cs 21
6. Код файла PopulationShow.cs 39

# КОД ФАЙЛА Functions.cs

using System;

namespace Библиотека\_классов

{//класс функций

public class Functions

{

//делегат для передачи выбранной функции

public delegate double FunctionVyb(double x, double y);

//методы доступных функций

public static double Function1(double x, double y)

{

return x \* y;

}

public static double Function2(double x, double y)

{

return x\*x + y \* y;

}

public static double Function3(double x, double y)

{

return Math.Sin(x) + Math.Cos(x) + Math.Sin(y) + Math.Cos(y);

}

public static double Function4(double x, double y)

{

return x \* Math.Cos(y) + y \* y \* Math.Sin(x);

}

public static double Function5(double x, double y)

{

return Math.Pow(x + y, 3) - 3 \* x \* y;

}

public static double Function6(double x, double y)

{

return (x / 3) + Math.Cos((x + 2 \* y) / 7);

}

public static double Function7(double x, double y)

{

return Math.Pow(Math.Sin(x), 3) + Math.Sin(x) - Math.Cos(y) + Math.Pow(Math.Cos(y), 3);

}

public static double Function8(double x, double y)

{

return Math.Pow(x, 5) + Math.Pow(y, 2) - 3 \* Math.Pow(x, 3) + 7 \* Math.Pow(y, 4) - 2 \* x \* y;

}

}

}

# КОД ФАЙЛА OptimizationAlgorithms.cs

using System;

namespace Библиотека\_классов

{//класс статических функций

public static class OptimizationAlgorithms

{//метод формирует новую популяцию

public static double[,] NewPopulation(double MinIntX, double MaxIntX, double MinIntY, double MaxIntY)

{

Random gen = new Random();

int populationsize = gen.Next(2, 100); //размер первоначальной популяции

double[,] newpop = new double[populationsize, 2]; //матрица популяции

for (int i = 0; i < populationsize; i++)

{

newpop[i, 0] = MinIntX + gen.NextDouble() \* (MaxIntX - MinIntX); //задаем случайно популяцию в искомом промежутке

newpop[i, 1] = MinIntY + gen.NextDouble() \* (MaxIntY - MinIntY);

}

return newpop;

}

//метод для нахождения количество элементов в мемеплексах

public static int MemeplexNumber(int populationsize)

{

int n = 0; //количество элементов в мемеплексах

Random gen = new Random();

while (true)

{

n = gen.Next(1, populationsize); //полагаем, что количество элементов одинаково во всех мемеплексах

if (populationsize % n == 0)

break;

}

return n;

}

//улучшения в мемеплексах для поиска максимума

public static double[,] PopulationGoodModernization(double[,] newpop, double MinIntX, double MaxIntX, double MinIntY, double MaxIntY, int populationsize, int n, Functions.FunctionVyb Fun)

{

double[, ,] memeplexes = new double[(int)populationsize / n, n, 2]; //создаем матрицу-мемеплексов

Random gen = new Random();

double[] max = new double[2];

max[0] = newpop[0, 0];

max[1] = newpop[0, 1]; //глобальный максимум

for (int i = 0; i < populationsize; i++)

if (Fun(max[0], max[1]) < Fun(newpop[i, 0], newpop[i, 1]))

{

max[0] = newpop[i, 0]; //поиск агента с максимальным значением фитнесс-функции

max[1] = newpop[i, 1];

}

for (int i = 0; i < memeplexes.GetLength(0); i++)

for (int j = 0; j < n; j++)

{

memeplexes[i, j, 0] = newpop[i + j, 0];

memeplexes[i, j, 1] = newpop[i + j, 1]; //задание мемеплексов

}

for (int i = 0; i < memeplexes.GetLength(0); i++) //модернизируем популяцию

{

double[] Sbest = new double[2];

Sbest[0] = memeplexes[i, 0, 0];

Sbest[1] = memeplexes[i, 0, 1];

double[] Sworst = new double[2]; //задание лучшего и худшего агентов

Sworst[0] = memeplexes[i, 0, 0];

Sworst[1] = memeplexes[i, 0, 1];

int SWind = 0; //индекс худшего агента

for (int k = 0; k < n; k++) //проходим по мемеплексу и ищем лучшего и худшего агентов в мемеплексе

{

if (Fun(Sbest[0], Sworst[1]) < Fun(memeplexes[i, k, 0], memeplexes[i, k, 1]))

{

Sbest[0] = memeplexes[i, k, 0]; //ищем лучшего агента в мемеплексе

Sbest[1] = memeplexes[i, k, 1];

}

else

if (Fun(Sworst[0], Sworst[1]) > Fun(memeplexes[i, k, 0], memeplexes[i, k, 1]))

{

Sworst[0] = memeplexes[i, k, 0]; //ищем худшего агента

Sworst[1] = memeplexes[i, k, 1];

SWind = k; //задаем индекс худшего агента

}

}

//начинаем улучшать позицию худшего агента

double[] Sworsttry = new double[2];

Sworsttry[0] = MinIntX;

Sworsttry[1] = MinIntY;

Sworsttry[0] = Sworst[0] + gen.NextDouble() \* (Sbest[0] - Sworst[0]); //улучшение на локальном максимуме

Sworsttry[1] = Sworst[1] + gen.NextDouble() \* (Sbest[1] - Sworst[1]);

if (Sworsttry[0] < MinIntX)

Sworsttry[0] = MinIntX; //если агент выходит за рамки присваиваем граничные значения

if (Sworsttry[1] < MinIntY)

Sworsttry[1] = MinIntY;

if (Sworsttry[0] > MaxIntX)

Sworsttry[0] = MaxIntX;

if (Sworsttry[1] > MaxIntY)

Sworsttry[1] = MaxIntY;

if (Fun(Sworsttry[0], Sworsttry[1]) > Fun(Sworst[0], Sworst[1]))

{

memeplexes[i, SWind, 0] = Sworsttry[0]; //если улучшение прошло успешно, то фиксируем улучшение

memeplexes[i, SWind, 1] = Sworsttry[1];

}

else //если улучшение не помогло, пробуем улучшить на глобальном максимуме

{

Sworsttry[0] = MinIntX;

Sworsttry[1] = MinIntY; //задаем начальное значение

Sworsttry[0] = Sworst[0] + gen.NextDouble() \* (max[0] - Sworst[0]); //улучшение на глобальном максимуме

Sworsttry[1] = Sworst[1] + gen.NextDouble() \* (max[1] - Sworst[1]);

if (Sworsttry[0] < MinIntX)

Sworsttry[0] = MinIntX; //если агент выходит за рамки присваиваем граничные значения

if (Sworsttry[1] < MinIntY)

Sworsttry[1] = MinIntY;

if (Sworsttry[0] > MaxIntX)

Sworsttry[0] = MaxIntX;

if (Sworsttry[1] > MaxIntY)

Sworsttry[1] = MaxIntY;

if (Fun(Sworsttry[0], Sworsttry[1]) > Fun(Sworst[0], Sworst[1])) //если улучшение помогло, то фиксируем его

{

memeplexes[i, SWind, 0] = Sworsttry[0];

memeplexes[i, SWind, 1] = Sworsttry[1];

}

else //если и это улучшение не прошло, то генерируем агента по новой

{

memeplexes[i, SWind, 0] = MinIntX + gen.NextDouble() \* (MaxIntX - MinIntX);

memeplexes[i, SWind, 1] = MinIntY + gen.NextDouble() \* (MaxIntY - MinIntY);

}

}

}

for (int i = 0; i < memeplexes.GetLength(0); i++)

{

for (int j = 0; j < n; j++)

{

newpop[i + j, 0] = memeplexes[i, j, 0]; //собираем популяцию воедино заново из измененного мемеплекса

newpop[i + j, 1] = memeplexes[i, j, 1];

}

}

return newpop;

}

//улучшения в мемеплексах для поиска минимума

public static double[,] PopulationBadModernization(double[,] newpop, double MinIntX, double MaxIntX, double MinIntY, double MaxIntY, int populationsize, int n, Functions.FunctionVyb Fun)

{

double[, ,] memeplexes = new double[(int)populationsize / n, n, 2]; //создаем матрицу-мемеплексов

Random gen = new Random();

double[] max = new double[2];

max[0] = newpop[0, 0];

max[1] = newpop[0, 1]; //глобальный максимум (минимальное значение)

for (int i = 0; i < populationsize; i++)

if (Fun(max[0], max[1]) > Fun(newpop[i, 0], newpop[i, 1]))

{

max[0] = newpop[i, 0]; //поиск агента с минимальным значением фитнесс функции

max[1] = newpop[i, 1];

}

for (int i = 0; i < memeplexes.GetLength(0); i++)

for (int j = 0; j < n; j++)

{

memeplexes[i, j, 0] = newpop[i + j, 0];

memeplexes[i, j, 1] = newpop[i + j, 1]; //задание мемеплексов

}

for (int i = 0; i < memeplexes.GetLength(0); i++) //модернизируем популяцию

{

double[] Sbest = new double[2];

Sbest[0] = memeplexes[i, 0, 0];

Sbest[1] = memeplexes[i, 0, 1];

double[] Sworst = new double[2]; //задание лучшего и худшего агентов

Sworst[0] = memeplexes[i, 0, 0];

Sworst[1] = memeplexes[i, 0, 1];

int SWind = 0; //индекс худшего агента

for (int k = 0; k < n; k++) //проходим по мемеплексу и ищем лучшего и худшего агентов в мемеплексе

{

if (Fun(Sbest[0], Sworst[1]) > Fun(memeplexes[i, k, 0], memeplexes[i, k, 1]))

{

Sbest[0] = memeplexes[i, k, 0]; //ищем лучшего агента в мемеплексе

Sbest[1] = memeplexes[i, k, 1];

}

else

if (Fun(Sworst[0], Sworst[1]) < Fun(memeplexes[i, k, 0], memeplexes[i, k, 1]))

{

Sworst[0] = memeplexes[i, k, 0]; //ищем худшего агента

Sworst[1] = memeplexes[i, k, 1];

SWind = k; //задаем индекс худшего агента

}

}

//начинаем улучшать позицию худшего агента в мемеплексе

double[] Sworsttry = new double[2];

Sworsttry[0] = MinIntX;

Sworsttry[1] = MinIntY;

Sworsttry[0] = Sworst[0] + gen.NextDouble() \* (Sbest[0] - Sworst[0]); //улучшение на локальном максимуме

Sworsttry[1] = Sworst[1] + gen.NextDouble() \* (Sbest[1] - Sworst[1]);

if (Sworsttry[0] < MinIntX)

Sworsttry[0] = MinIntX; //если агент выходит за рамки присваиваем граничные значения

if (Sworsttry[1] < MinIntY)

Sworsttry[1] = MinIntY;

if (Sworsttry[0] > MaxIntX)

Sworsttry[0] = MaxIntX;

if (Sworsttry[1] > MaxIntY)

Sworsttry[1] = MaxIntY;

if (Fun(Sworsttry[0], Sworsttry[1]) < Fun(Sworst[0], Sworst[1]))

{

memeplexes[i, SWind, 0] = Sworsttry[0]; //если улучшение прошло успешно, то фиксируем улучшение

memeplexes[i, SWind, 1] = Sworsttry[1];

}

else //если улучшение неудачно, пробуем улучшить на глобальном максимуме

{

Sworsttry[0] = MinIntX;

Sworsttry[1] = MinIntY; //задаем начальное значение

Sworsttry[0] = Sworst[0] + gen.NextDouble() \* (max[0] - Sworst[0]); //улучшение на глобальном максимуме

Sworsttry[1] = Sworst[1] + gen.NextDouble() \* (max[1] - Sworst[1]);

if (Sworsttry[0] < MinIntX)

Sworsttry[0] = MinIntX; //если агент выходит за рамки присваиваем граничные значения

if (Sworsttry[1] < MinIntY)

Sworsttry[1] = MinIntY;

if (Sworsttry[0] > MaxIntX)

Sworsttry[0] = MaxIntX;

if (Sworsttry[1] > MaxIntY)

Sworsttry[1] = MaxIntY;

if (Fun(Sworsttry[0], Sworsttry[1]) < Fun(Sworst[0], Sworst[1])) //если улучшение помогло, фиксируем

{

memeplexes[i, SWind, 0] = Sworsttry[0];

memeplexes[i, SWind, 1] = Sworsttry[1];

}

else //если и это улучшение не прошло, то генерируем агента по новой

{

memeplexes[i, SWind, 0] = MinIntX + gen.NextDouble() \* (MaxIntX - MinIntX); //задаем случайно популяцию - в промежутке искомом

memeplexes[i, SWind, 1] = MinIntY + gen.NextDouble() \* (MaxIntY - MinIntY);

}

}

}

for (int i = 0; i < memeplexes.GetLength(0); i++)

{

for (int j = 0; j < n; j++)

{

newpop[i + j, 0] = memeplexes[i, j, 0]; //собираем популяцию воедино заново из измененного мемеплекса

newpop[i + j, 1] = memeplexes[i, j, 1];

}

}

return newpop;

}

//перемешивание популяции

public static double[,] PopulationShuffle(double[,] newpop)

{

Random gen = new Random();

for (int i = newpop.GetLength(0) - 1; i > 0; i--)

{

int j = gen.Next(i); //генерируем номер

double[] tmp = new double[2];

tmp[0] = newpop[j, 0]; //проводим обмен

tmp[1] = newpop[j, 1];

newpop[j, 0] = newpop[i - 1, 0];

newpop[j, 1] = newpop[i - 1, 1];

newpop[i - 1, 0] = tmp[0];

newpop[i - 1, 1] = tmp[1];

}

return newpop;

}

//максимум в популяции

public static double[] Maximum(double[,] newpop, Functions.FunctionVyb Fun)

{

double[] max = new double[2];

max[0] = newpop[0, 0];

max[1] = newpop[0, 1];

for (int i = 0; i < newpop.GetLength(0); i++)

{

if (Fun(max[0], max[1]) < Fun(newpop[i, 0], newpop[i, 1]))

{ //присваиваем, если значение больше промежуточного максимума

max[0] = newpop[i, 0];

max[1] = newpop[i, 1];

}

}

return max;

}

//минимум в популяции

public static double[] Minimum(double[,] newpop, Functions.FunctionVyb Fun)

{

double[] min = new double[2];

min[0] = newpop[0, 0];

min[1] = newpop[0, 1];

for (int i = 0; i < newpop.GetLength(0); i++)

{

if (Fun(min[0], min[1]) > Fun(newpop[i, 0], newpop[i, 1]))

{ //присваиваем, если значение меньше промежуточного минимума

min[0] = newpop[i, 0];

min[1] = newpop[i, 1];

}

}

return min;

}

//выполнение алгоритма полностью - максимум функции

public static double[] FunctionMaximum(int kol, double MinIntX, double MaxIntX, double MinIntY, double MaxIntY, Functions.FunctionVyb Fun)

{

double[,] newpop = NewPopulation(MinIntX, MaxIntX, MinIntY, MaxIntY); //массив популяции

int n=MemeplexNumber(newpop.GetLength(0));

for (int i = 0; i < kol; i++)

{

PopulationGoodModernization(newpop, MinIntX, MaxIntX, MinIntY, MaxIntY, newpop.GetLength(0), n, Fun); //улучшение

newpop = PopulationShuffle(newpop); //перемешиваем популяцию

}

return Maximum(newpop, Fun);

}

//выполнение алгоритма полностью - минимум функции

public static double[] FunctionMinimum(int kol, double MinIntX, double MaxIntX, double MinIntY, double MaxIntY, Functions.FunctionVyb Fun)

{

double[,] newpop = NewPopulation(MinIntX, MaxIntX, MinIntY, MaxIntY); //массив популяции

int n = MemeplexNumber(newpop.GetLength(0));

for (int i = 0; i < kol; i++)

{

PopulationBadModernization(newpop, MinIntX, MaxIntX, MinIntY, MaxIntY, newpop.GetLength(0), n, Fun); //улучшение

newpop = PopulationShuffle(newpop); //перемешиваем популяцию

}

return Minimum(newpop,Fun);

}

}

}

# КОД ФАЙЛА FunctionGraph.cs

using System;

using System.Windows.Forms;

using Библиотека\_классов;

using Chart3DLib;

namespace Визуализация

{

public partial class FunctionGraph : Form

{

double minX, maxX, minY, maxY; //поля для заданных границ

Functions.FunctionVyb Fun; //поле для функции

Point3[,] pts = new Point3[20, 20]; //задание матрицы точек с тремя координатами для построения графика

//конструктор формы

public FunctionGraph(Functions.FunctionVyb Fun, double minX, double maxX, double minY, double maxY)

{

InitializeComponent();

FunctionGraphik.C3DrawChart.ChartType = DrawChart.ChartTypeEnum.SurfaceFillContour; //определяем вид графика - контурный и объемный

Azimuth.Text = PovorotAzimuth.Value.ToString(); //для поворота по горизонтали выводим в соответсвующую метку

Elevation.Text = PovorotElevation.Value.ToString(); //для поворота по вертикали выводим в соответсвующую метку

this.Fun = Fun; //присваиваем функцию переданную в конструктор полю

if (minX == maxX) //присваиваем границы переданные - полям, если границы одинаковые - расширяем, чтобы график не выродился в точку

{

this.minX = minX-1;

this.maxX = maxX+1;

}

else

{

this.minX = minX;

this.maxX = maxX;

}

if (minY == maxY)

{

this.minY = minY-1;

this.maxY = maxY+1;

}

else

{

this.minY = minY;

this.maxY = maxY;

}

AddData(); //вызываем метод для построения

}

//метод для построения графика

private void AddData()

{

FunctionGraphik.C3Axes.XMin = (float)minX; //задаем границы для графика

FunctionGraphik.C3Axes.XMax = (float)maxX;

FunctionGraphik.C3Axes.YMin = (float)minY;

FunctionGraphik.C3Axes.YMax = (float)maxY;

FunctionGraphik.C3Labels.XLabel = "Х"; //именуем оси графика

FunctionGraphik.C3Labels.YLabel = "Y";

FunctionGraphik.C3Labels.ZLabel = "Z";

FunctionGraphik.C3Labels.Title = "График функции"; //заголовок для графика

FunctionGraphik.C3Axes.XTick = (float)((FunctionGraphik.C3Axes.XMax - FunctionGraphik.C3Axes.XMin) / 4); //для осей задаем количество меток

FunctionGraphik.C3Axes.YTick = (float)((FunctionGraphik.C3Axes.YMax - FunctionGraphik.C3Axes.YMin) / 4);

FunctionGraphik.C3DataSeries.XDataMin = FunctionGraphik.C3Axes.XMin; //задаем минимальные границы, чтобы график при повороте не ускакивал

FunctionGraphik.C3DataSeries.YDataMin = FunctionGraphik.C3Axes.YMin;

FunctionGraphik.C3DataSeries.XSpacing = (float)((FunctionGraphik.C3Axes.XMax - FunctionGraphik.C3Axes.XMin) / 20); //шаг, с которым мы увеличиваем координаты

FunctionGraphik.C3DataSeries.YSpacing = (float)((FunctionGraphik.C3Axes.YMax - FunctionGraphik.C3Axes.YMin) / 20);

float min = (float)Fun(FunctionGraphik.C3Axes.XMin, FunctionGraphik.C3Axes.YMin); //для поиска минимума/максимума для координаты Z

float max = (float)Fun(FunctionGraphik.C3Axes.XMin, FunctionGraphik.C3Axes.YMin);

for (int i = 0; i < 20; i++)

{ //задаем матрицу

float x = FunctionGraphik.C3DataSeries.XDataMin + i \* FunctionGraphik.C3DataSeries.XSpacing; //изменяем координату по Х

for (int j = 0; j < 20; j++)

{

float y = FunctionGraphik.C3DataSeries.YDataMin + j \* FunctionGraphik.C3DataSeries.YSpacing; //изменяем координату по Y

float z = (float)Fun(x, y); //задаем координату Z

pts[i, j] = new Point3(x, y, z, 1); //добавляем точку в матрицу

if (min > z) min = z; //ищем минимум

if (max < z) max = z; //ищем максимум

}

}

FunctionGraphik.C3Axes.ZMin = (float)min; //задаем границы оси Z

FunctionGraphik.C3Axes.ZMax = (float)max;

FunctionGraphik.C3Axes.ZTick = (float)((max - min) / 4); //для осей задаем количество меток

FunctionGraphik.C3ViewAngle.Elevation = PovorotElevation.Value; //задаем поворот графика

FunctionGraphik.C3ViewAngle.Azimuth = PovorotAzimuth.Value; //задаем поворот графика

FunctionGraphik.C3DataSeries.PointArray = pts; //"даем" графику матрицу точек

}

//для вертикального поворота

private void PovorotElevation\_Scroll(object sender, EventArgs e)

{

Elevation.Text = PovorotElevation.Value.ToString();

AddData();

}

//для горизонтального поворота

private void PovorotAzimuth\_Scroll(object sender, EventArgs e)

{

Azimuth.Text = PovorotAzimuth.Value.ToString();

AddData();

}

//перестраиваем график при увеличении размера формы

private void FunctionGraphik\_Resize(object sender, EventArgs e)

{

AddData();

}

}

}

# КОД ФАЙЛА MainForm.cs

using System;

using System.Windows.Forms;

using Библиотека\_классов;

namespace Визуализация

{

public partial class MainForm : Form

{

Functions.FunctionVyb Fun = Functions.Function1; //действующая функция

int KolIt = 1; //количество итераций

double MiX; //границы

double MiY;

double MaX;

double MaY;

int clickitem = 1; //счетчик для выбора функции

public Graphiks newForm; //для открытия формы графика

//конструктор формы

public MainForm()

{

InitializeComponent();

Function.Image = Function.Image = Properties.Resources.Function1; //начальное изображение

IterationToolTip.SetToolTip(Kol\_It\_TrackBar, "Количество раз, которое выполнится алгоритм.");

MinXToolTip.SetToolTip(MinX, "Поле должно содержать рациональное число.");

MaxXToolTip.SetToolTip(MaxX, "Поле должно содержать рациональное число, большее чем минимальная граница по Х.");

MinYToolTip.SetToolTip(MinY, "Поле должно содержать рациональное число.");

MaxYToolTip.SetToolTip(MaxY, "Поле должно содержать рациональное число, большее чем минимальная граница по Y");

FuncToolTip.SetToolTip(Function, "Для выбора другой функции, кликните на функцию.");

}

//проверка корректности для текстбокса - минимальной границы Х

private void MinX\_TextChanged(object sender, EventArgs e)

{

if (!double.TryParse(MinX.Text, out MiX))

Mistake\_minX.SetError(MinX, "Поле должно содержать рациональное число");

else

Mistake\_minX.Clear();

}

//проверка корректности для текстбокса - минимальной границы Y

private void MinY\_TextChanged(object sender, EventArgs e)

{

if (!double.TryParse(MinY.Text, out MiY))

Mistake\_minY.SetError(MinY, "Поле должно содержать рациональное число");

else

Mistake\_minY.Clear();

}

//проверка корректности для текстбокса - максимальной границы Х

private void MaxX\_TextChanged(object sender, EventArgs e)

{

if (!double.TryParse(MaxX.Text, out MaX))

Mistake\_minX.SetError(MaxX, "Поле должно содержать рациональное число, большее чем минимальная граница по Х");

else

Mistake\_minX.Clear();

}

//проверка корректности для текстбокса - максимальной границы Y

private void MaxY\_TextChanged(object sender, EventArgs e)

{

if (!double.TryParse(MaxY.Text, out MaY))

Mistake\_maxY.SetError(MaxY, "Поле должно содержать рациональное число, большее чем минимальная граница по Y");

else

Mistake\_maxY.Clear();

}

//методы для очищения ошибок - для всех текстбоксов границ

private void MinX\_Leave(object sender, EventArgs e)

{

if (MinX.Text == "")

Mistake\_minX.Clear();

}

private void MinY\_Leave(object sender, EventArgs e)

{

if (MinY.Text == "")

Mistake\_minY.Clear();

}

private void MaxX\_Leave(object sender, EventArgs e)

{

if ((MaxX.Text == "")||((double.TryParse(MinX.Text, out MiX)) &&(double.TryParse(MaxX.Text, out MaX))&&(MiX-MaX>=0)))

Mistake\_maxX.Clear();

}

private void MaxY\_Leave(object sender, EventArgs e)

{

if ((MaxY.Text == "") || ((double.TryParse(MinY.Text, out MiY)) && (double.TryParse(MaxY.Text, out MaY)) && (MiY - MaY >= 0)))

Mistake\_maxY.Clear();

}

//нажатие на кнопку график

private void GraphOpen\_Click(object sender, EventArgs e)

{//проверка на ошибки и полностью введены ли данные

bool f = false;

if (!(Maximum.Checked || Minimum.Checked))

{

MessageBox.Show(@"Выберите максимум или минимум!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else f = true;

if (!(f && (Mistake\_minX.GetError(MinX) == "") && (MinX.Text != "")))

{

MessageBox.Show(@"Неверно введена или вообще не введена минимальная граница X!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_minX.SetError(MinX, "Поле должно содержать рациональное число");

return;

}

else f = false;

if (!(!f && (Mistake\_maxX.GetError(MaxX) == "") && (MaxX.Text != "")))

{

MessageBox.Show(@"Неверно введена или вообще не введена максимальная граница X!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_minX.SetError(MaxX, "Поле должно содержать рациональное число, большее чем минимальная граница по Х");

return;

}

else f = true;

if (!(f && (Mistake\_minY.GetError(MinY) == "") && (MinY.Text != "")))

{

MessageBox.Show(@"Неверно введена или вообще не введена минимальная граница Y!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_minY.SetError(MinY, "Поле должно содержать рациональное число");

return;

}

else f = false;

if (!(!f && (Mistake\_maxY.GetError(MaxY) == "") && (MaxY.Text != "")))

{

MessageBox.Show(@"Неверно введена или вообще не введена максимальная граница Y!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_maxY.SetError(MaxY, "Поле должно содержать рациональное число, большее чем минимальная граница по Y");

return;

}

else f = true;

if (!((MiX <= MaX) && f))

{

MessageBox.Show(@"Минимальная граница по Х больше максимальной!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_maxY.SetError(MaxX, "Поле должно содержать рациональное число, большее чем минимальная граница по Х");

return;

}

else f = false;

if (!((MiY <= MaY) && !f))

{

MessageBox.Show(@"Минимальная граница по Y больше максимальной!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_maxY.SetError(MaxY, "Поле должно содержать рациональное число, большее чем минимальная граница по Y");

return;

}

else f = false;

if (!f)

{ //если нет ошибок открываем новую форму

Mistake\_maxX.Clear();

Mistake\_maxY.Clear();//очищаем ошибки

Mistake\_minX.Clear();

Mistake\_minY.Clear();

newForm = new Graphiks(KolIt, MiX, MaX, MiY, MaY, Maximum.Checked, Minimum.Checked, Fun);

newForm.Show(); //показываем форму графиков

}

}

//выбор функции

private void Function\_Click(object sender, EventArgs e)

{

clickitem++; //наращиваем количество нажатий

switch (clickitem)

{

case 2: Function.Image = Properties.Resources.Function2;

Fun = Functions.Function2; //переключаем картинку и функции

break;

case 3: Function.Image = Properties.Resources.Function3;

Fun = Functions.Function3; //переключаем картинку и функции

break;

case 4: Function.Image = Properties.Resources.Function4;

Fun = Functions.Function4; //переключаем картинку и функции

break;

case 5: Function.Image = Properties.Resources.Function5;

Fun = Functions.Function5; //переключаем картинку и функции

break;

case 6: Function.Image = Properties.Resources.Function6;

Fun = Functions.Function6; //переключаем картинку и функции

break;

case 7: Function.Image = Properties.Resources.Function7;

Fun = Functions.Function7; //переключаем картинку и функции

break;

case 8: Function.Image = Properties.Resources.Function8;

Fun = Functions.Function8; //переключаем картинку и функции

break;

default:

clickitem =1; //количество кликов - ставим начальное значение

Function.Image = Properties.Resources.Function1;

Fun = Functions.Function1; //переключаем картинку и функции

break;

}

}

//выбор количества итераций на трекбаре

private void Kol\_It\_TrackBar\_Scroll(object sender, EventArgs e)

{//отображаем в текстбоксе количества итераций - выбранное количество итераций

Kol\_It\_Number.Text = Kol\_It\_TrackBar.Value.ToString();

}

//ввод количества итераций в текстбокс

private void Kol\_It\_Number\_TextChanged(object sender, EventArgs e)

{

if (int.TryParse(Kol\_It\_Number.Text, out KolIt) && (KolIt > 0) && (KolIt < 1000))

{ //если данные корректны

Kol\_It\_TrackBar.Value = KolIt; //на трекбаре ставим введенное количество итераций

while (Kol\_It\_Number.Text[0] == '0') //убираем ведущие нули

Kol\_It\_Number.Text = KolIt.ToString();

}

else

{

if (KolIt >= 1000) //если количество итераций введено больше 1000

{

KolIt = 1000;

Kol\_It\_TrackBar.Value = KolIt;

Kol\_It\_Number.Text = "1000";

}

else

if (KolIt <= 0)

{//если количество итераций введено меньше 1

KolIt = 1;

Kol\_It\_TrackBar.Value = KolIt;

Kol\_It\_Number.Text = "1";

}

while (Kol\_It\_Number.Text[0] == '0') //убираем ведущие нули

Kol\_It\_Number.Text = KolIt.ToString();

}

}

//нажатие на кнопку результат

private void Result\_Click(object sender, EventArgs e)

{

//проверка на ошибки и полностью введены ли данные

bool f = false;

if (!(Maximum.Checked || Minimum.Checked))

{

MessageBox.Show(@"Выберите максимум или минимум!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else f = true;

if (!(f && (Mistake\_minX.GetError(MinX) == "") && (MinX.Text != "")))

{

MessageBox.Show(@"Неверно введена или вообще не введена минимальная граница X!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_minX.SetError(MinX, "Поле должно содержать рациональное число");

return;

}

else f = false;

if (!(!f && (Mistake\_maxX.GetError(MaxX) == "") && (MaxX.Text != "")))

{

MessageBox.Show(@"Неверно введена или вообще не введена максимальная граница X!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_minX.SetError(MaxX, "Поле должно содержать рациональное число, большее чем минимальная граница по Х");

return;

}

else f = true;

if (!(f && (Mistake\_minY.GetError(MinY) == "") && (MinY.Text != "")))

{

MessageBox.Show(@"Неверно введена или вообще не введена минимальная граница Y!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_minY.SetError(MinY, "Поле должно содержать рациональное число");

return;

}

else f = false;

if (!(!f && (Mistake\_maxY.GetError(MaxY) == "") && (MaxY.Text != "")))

{

MessageBox.Show(@"Неверно введена или вообще не введена максимальная граница Y!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_maxY.SetError(MaxY, "Поле должно содержать рациональное число, большее чем минимальная граница по Y");

return;

}

else f = true;

if (!((MiX <= MaX) && f))

{

MessageBox.Show(@"Минимальная граница по Х больше максимальной!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_maxY.SetError(MaxX, "Поле должно содержать рациональное число, большее чем минимальная граница по Х");

return;

}

else f = false;

if (!((MiY <= MaY) && !f))

{

MessageBox.Show(@"Минимальная граница по Y больше максимальной!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

Mistake\_maxY.SetError(MaxY, "Поле должно содержать рациональное число, большее чем минимальная граница по Y");

return;

}

else f = false;

if (!f)

{ //если нет ошибок, то в зависимости от того максимум или минимум выводим результат

Mistake\_maxX.Clear();

Mistake\_maxY.Clear();//очищаем ошибки

Mistake\_minX.Clear();

Mistake\_minY.Clear();

if (Maximum.Checked)

{

double[] ar = new double[2];

ar = OptimizationAlgorithms.FunctionMaximum(KolIt, MiX, MaX, MiY, MaY, Fun);

string s = String.Format(@"Максимум функции:

Координата Х: {0}

Координата Y: {1}", ar[0], ar[1]);

MessageBox.Show(s, "Результат", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

if (Minimum.Checked)

{

double[] ar = new double[2];

ar = OptimizationAlgorithms.FunctionMinimum(KolIt, MiX, MaX, MiY, MaY, Fun);

string s = String.Format(@"Минимум функции:

Координата Х: {0}

Координата Y: {1}", ar[0], ar[1]);

MessageBox.Show(s, "Результат", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

}

}

}

}

# КОД ФАЙЛА Graphiks.cs

using System;

using System.Windows.Forms;

using Библиотека\_классов;

using System.Drawing;

using System.IO;

using ZedGraph;

namespace Визуализация

{

public partial class Graphiks : Form

{

int Kol; //ряд полей класса, используемых для нескольких методов

double MaxX, MinX, MaxY, MinY; //границы интервалов

bool Max = false; //значения показывающие максимум или минимум

bool Min = false;

double[,] newpop; //популяция

int KolIterations = 0; //количество итераций

double[,] lastmassiv; //популяция предыдущей итерации

bool flag = false; //флаг для проверки нажатия на кнопку предыдущая итерация

double[] Isk = new double[2]; //массивы для максимума и минимума

double[] Isk2 = new double[2];

Functions.FunctionVyb Fun; //действующая функция

GraphPane graph; //экземпляр графика

PointPairList list = new PointPairList(); //лист точек

FunctionGraph newForm;

//построение осей

public void Postroenye()

{

graph = zedGraph.GraphPane; //связываем с элементом на форме

graph.Title.IsVisible = false; //прячем заголовок

Setka(); //делаем сетку

Points(newpop); //выводим точки

ZnakVyvod(newpop); //выводим максимум минимум

graph.XAxis.Title.IsVisible = false; //прячем названия осей

graph.YAxis.Title.IsVisible = false;

if (MinX == MaxX) //если равные координаты

{

graph.XAxis.Scale.Min = MinX-0.5;

graph.XAxis.Scale.Max = MaxX+0.5;

}

else

{//если неравные координаты

graph.XAxis.Scale.Min = MinX;

graph.XAxis.Scale.Max = MaxX;

}

if (MinY == MaxY)

{//если равные координаты

graph.YAxis.Scale.Min = MinY - 0.5;

graph.YAxis.Scale.Max = MaxY + 0.5;

}

else

{//если неравные координаты

graph.YAxis.Scale.Min = MinY;

graph.YAxis.Scale.Max = MaxY;

}

graph.XAxis.Cross = 0.0; //делаем оси пересекающимися в (0,0)

graph.YAxis.Cross = 0.0;

zedGraph.AxisChange(); //обновление графика и перерисовка его

zedGraph.Invalidate();

}

//построение сетки для графика

public void Setka()

{

// включаем отображение сетки напротив крупных рисок по оси X

graph.XAxis.MajorGrid.IsVisible = true;

// задаем вид пунктирной линии для крупных рисок по оси X

graph.XAxis.MajorGrid.DashOn = 10; // длина штрихов равна 10 пикселям

graph.XAxis.MajorGrid.DashOff = 5; //5 пикселей - пропуск

// включаем отображение сетки напротив крупных рисок по оси Y

graph.YAxis.MajorGrid.IsVisible = true;

// аналогично задаем вид пунктирной линии для крупных рисок по оси Y

graph.YAxis.MajorGrid.DashOn = 10;

graph.YAxis.MajorGrid.DashOff = 5;

// включаем отображение сетки напротив мелких рисок по оси X

graph.YAxis.MinorGrid.IsVisible = true;

// задаем вид пунктирной линии для крупных рисок по оси Y

graph.YAxis.MinorGrid.DashOn = 1; // длина штрихов равна одному пикселю

graph.YAxis.MinorGrid.DashOff = 2; // затем 2 пикселя - пропуск

// включаем отображение сетки напротив мелких рисок по оси Y

graph.XAxis.MinorGrid.IsVisible = true;

// аналогично задаем вид пунктирной линии для крупных рисок по оси Y

graph.XAxis.MinorGrid.DashOn = 1;

graph.XAxis.MinorGrid.DashOff = 2;

}

//метод для вывода точек

public void Points(double[,] massiv)

{

graph.CurveList.Clear(); //очищаем предыдущие точки

list.Clear(); //очищаем лист

if (Max)

{ //если ищем максимум

Isk = OptimizationAlgorithms.Maximum(massiv, Fun);

Isk2 = OptimizationAlgorithms.Minimum(massiv, Fun);

PointPairList list1 = new PointPairList();

list1.Add(Isk[0], Isk[1]); //добавляем точки максимума-минимума в отдельные листы

PointPairList list2 = new PointPairList();

list2.Add(Isk2[0], Isk2[1]);

LineItem myCurve1 = graph.AddCurve("", list1, Color.Red, SymbolType.Circle); //добавляем точку максимума

myCurve1.Label.IsVisible = false; //прячем название у точек

myCurve1.Symbol.Fill.Type = FillType.Solid; //определяем заливку точек

myCurve1.Symbol.Size = 7; //определяем размер точек

LineItem myCurve2 = graph.AddCurve("", list2, Color.Yellow, SymbolType.Circle); //добавляем точку минимума

myCurve2.Label.IsVisible = false; //прячем название у точек

myCurve2.Symbol.Fill.Type = FillType.Solid; //определяем заливку точек

myCurve2.Symbol.Size = 7; //определяем размер точек

}

if (Min)

{ //если ищем минимум

Isk2 = OptimizationAlgorithms.Maximum(massiv, Fun);

Isk = OptimizationAlgorithms.Minimum(massiv, Fun);

PointPairList list1 = new PointPairList();

list1.Add(Isk[0], Isk[1]); //добавляем точки максимума-минимума в отдельные листы

PointPairList list2 = new PointPairList();

list2.Add(Isk2[0], Isk2[1]);

LineItem myCurve1 = graph.AddCurve("", list1, Color.Red, SymbolType.Circle); //добавляем точку минимума

myCurve1.Label.IsVisible = false; //прячем название у точек

myCurve1.Symbol.Fill.Type = FillType.Solid; //определяем заливку точек

myCurve1.Symbol.Size = 7; //определяем размер точек

LineItem myCurve2 = graph.AddCurve("", list2, Color.Yellow, SymbolType.Circle); //добавляем точку максимума

myCurve2.Label.IsVisible = false; //прячем название у точек

myCurve2.Symbol.Fill.Type = FillType.Solid; //определяем заливку точек

myCurve2.Symbol.Size = 7; //определяем размер точек

}

for (int i = 0; i < massiv.GetLength(0); i++)

{

double x = massiv[i, 0];

double y = massiv[i, 1];

list.Add(x, y); //формируем лист из точек

}

LineItem myCurve = graph.AddCurve("",list, Color.Green, SymbolType.Circle); //добавляем лист точек как линию

myCurve.Label.IsVisible = false; //прячем название у точек

myCurve.Line.IsVisible = false; //прячем линию, оставляем только точки

myCurve.Symbol.Fill.Type = FillType.Solid; //определяем заливку точек

myCurve.Symbol.Size = 7; //определяем размер точек

zedGraph.AxisChange(); //обновление графика и перерисовка его

zedGraph.Invalidate();

}

//метод вывода результата, когда итерации закончились

public void ResultVyvod()

{

if (Max)

{//если ищем максимум

Isk = OptimizationAlgorithms.Maximum(newpop, Fun);

string s = String.Format(@"Максимум функции:

Координата Х: {0}

Координата Y: {1}", Isk[0], Isk[1]); //формируем строку

MessageBox.Show(s, "Выполнено", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

if (Min)

{//если ищем минимум

Isk = OptimizationAlgorithms.Minimum(newpop, Fun);

string s = String.Format(@"Минимум функции:

Координата Х: {0}

Координата Y: {1}", Isk[0], Isk[1]); //формируем строку

MessageBox.Show(s, "Выполнено", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

}

//метод для вывода максимума/минимума предыдущей итерации

//аналогичен нижнему методу, только без улучшений

public void ZnakVyvod(double[,] massiv)

{

if (Max)

{

Isk = OptimizationAlgorithms.Maximum(massiv, Fun);

Isk2 = OptimizationAlgorithms.Minimum(massiv, Fun);

switch ((int)KolZnakov.Value)

{//в зависимости от количества знаков заполняем информацию об агентах

case 0:

BestX.Text = String.Format("Координата Х: {0:f0}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f0}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f0}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f0}", Isk2[1]);

break;

case 1:

BestX.Text = String.Format("Координата Х: {0:f1}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f1}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f1}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f1}", Isk2[1]);

break;

case 2:

BestX.Text = String.Format("Координата Х: {0:f2}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f2}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f2}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f2}", Isk2[1]);

break;

case 3:

BestX.Text = String.Format("Координата Х: {0:f3}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f3}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f3}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f3}", Isk2[1]);

break;

case 4:

BestX.Text = String.Format("Координата Х: {0:f4}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f4}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f4}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f4}", Isk2[1]);

break;

case 5:

BestX.Text = String.Format("Координата Х: {0:f5}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f5}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f5}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f5}", Isk2[1]);

break;

case 6:

BestX.Text = String.Format("Координата Х: {0:f6}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f6}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f6}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f6}", Isk2[1]);

break;

case 7:

BestX.Text = String.Format("Координата Х: {0:f7}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f7}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f7}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f7}", Isk2[1]);

break;

case 8:

BestX.Text = String.Format("Координата Х: {0:f8}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f8}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f8}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f8}", Isk2[1]);

break;

case 9:

BestX.Text = String.Format("Координата Х: {0:f9}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f9}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f9}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f9}", Isk2[1]);

break;

case 10:

BestX.Text = String.Format("Координата Х: {0:f10}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f10}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f10}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f10}", Isk2[1]);

break;

case 11:

BestX.Text = String.Format("Координата Х: {0:f11}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f11}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f11}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f11}", Isk2[1]);

break;

case 12:

BestX.Text = String.Format("Координата Х: {0:f12}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f12}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f12}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f12}", Isk2[1]);

break;

case 13:

BestX.Text = String.Format("Координата Х: {0:f13}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f13}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f13}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f13}", Isk2[1]);

break;

case 14:

BestX.Text = String.Format("Координата Х: {0:f14}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f14}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f14}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f14}", Isk2[1]);

break;

case 15:

BestX.Text = String.Format("Координата Х: {0:f15}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f15}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f15}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f15}", Isk2[1]);

break;

}

}

if (Min)

{

Isk = OptimizationAlgorithms.Minimum(massiv, Fun);

Isk2 = OptimizationAlgorithms.Maximum(massiv, Fun);

switch ((int)KolZnakov.Value)

{//в зависимости от количества знаков заполняем информацию об агентах

case 0:

BestX.Text = String.Format("Координата Х: {0:f0}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f0}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f0}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f0}", Isk2[1]);

break;

case 1:

BestX.Text = String.Format("Координата Х: {0:f1}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f1}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f1}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f1}", Isk2[1]);

break;

case 2:

BestX.Text = String.Format("Координата Х: {0:f2}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f2}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f2}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f2}", Isk2[1]);

break;

case 3:

BestX.Text = String.Format("Координата Х: {0:f3}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f3}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f3}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f3}", Isk2[1]);

break;

case 4:

BestX.Text = String.Format("Координата Х: {0:f4}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f4}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f4}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f4}", Isk2[1]);

break;

case 5:

BestX.Text = String.Format("Координата Х: {0:f5}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f5}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f5}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f5}", Isk2[1]);

break;

case 6:

BestX.Text = String.Format("Координата Х: {0:f6}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f6}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f6}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f6}", Isk2[1]);

break;

case 7:

BestX.Text = String.Format("Координата Х: {0:f7}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f7}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f7}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f7}", Isk2[1]);

break;

case 8:

BestX.Text = String.Format("Координата Х: {0:f8}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f8}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f8}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f8}", Isk2[1]);

break;

case 9:

BestX.Text = String.Format("Координата Х: {0:f9}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f9}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f9}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f9}", Isk2[1]);

break;

case 10:

BestX.Text = String.Format("Координата Х: {0:f10}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f10}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f10}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f10}", Isk2[1]);

break;

case 11:

BestX.Text = String.Format("Координата Х: {0:f11}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f11}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f11}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f11}", Isk2[1]);

break;

case 12:

BestX.Text = String.Format("Координата Х: {0:f12}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f12}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f12}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f12}", Isk2[1]);

break;

case 13:

BestX.Text = String.Format("Координата Х: {0:f13}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f13}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f13}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f13}", Isk2[1]);

break;

case 14:

BestX.Text = String.Format("Координата Х: {0:f14}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f14}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f14}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f14}", Isk2[1]);

break;

case 15:

BestX.Text = String.Format("Координата Х: {0:f15}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f15}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f15}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f15}", Isk2[1]);

break;

}

}

}

//метод для вывода максимума/минимума и улучшения

public void ZnakVyvodNextIt()

{

if (Max)

{//улучшение и ищем максимум-минимум

OptimizationAlgorithms.PopulationGoodModernization(newpop, MinX, MaxX, MinY, MaxY, newpop.GetLength(0), OptimizationAlgorithms.MemeplexNumber(newpop.GetLength(0)), Fun);

Isk = OptimizationAlgorithms.Maximum(newpop, Fun);

Isk2 = OptimizationAlgorithms.Minimum(newpop, Fun);

switch ((int)KolZnakov.Value)

{ //в зависимости от количества знаков заполняем информацию об агентах

case 0:

BestX.Text = String.Format("Координата Х: {0:f0}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f0}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f0}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f0}", Isk2[1]);

break;

case 1:

BestX.Text = String.Format("Координата Х: {0:f1}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f1}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f1}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f1}", Isk2[1]);

break;

case 2:

BestX.Text = String.Format("Координата Х: {0:f2}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f2}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f2}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f2}", Isk2[1]);

break;

case 3:

BestX.Text = String.Format("Координата Х: {0:f3}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f3}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f3}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f3}", Isk2[1]);

break;

case 4:

BestX.Text = String.Format("Координата Х: {0:f4}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f4}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f4}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f4}", Isk2[1]);

break;

case 5:

BestX.Text = String.Format("Координата Х: {0:f5}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f5}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f5}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f5}", Isk2[1]);

break;

case 6:

BestX.Text = String.Format("Координата Х: {0:f6}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f6}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f6}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f6}", Isk2[1]);

break;

case 7:

BestX.Text = String.Format("Координата Х: {0:f7}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f7}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f7}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f7}", Isk2[1]);

break;

case 8:

BestX.Text = String.Format("Координата Х: {0:f8}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f8}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f8}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f8}", Isk2[1]);

break;

case 9:

BestX.Text = String.Format("Координата Х: {0:f9}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f9}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f9}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f9}", Isk2[1]);

break;

case 10:

BestX.Text = String.Format("Координата Х: {0:f10}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f10}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f10}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f10}", Isk2[1]);

break;

case 11:

BestX.Text = String.Format("Координата Х: {0:f11}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f11}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f11}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f11}", Isk2[1]);

break;

case 12:

BestX.Text = String.Format("Координата Х: {0:f12}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f12}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f12}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f12}", Isk2[1]);

break;

case 13:

BestX.Text = String.Format("Координата Х: {0:f13}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f13}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f13}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f13}", Isk2[1]);

break;

case 14:

BestX.Text = String.Format("Координата Х: {0:f14}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f14}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f14}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f14}", Isk2[1]);

break;

case 15:

BestX.Text = String.Format("Координата Х: {0:f15}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f15}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f15}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f15}", Isk2[1]);

break;

}

}

if (Min)

{//улучшение и ищем максимум-минимум

OptimizationAlgorithms.PopulationBadModernization(newpop, MinX, MaxX, MinY, MaxY, newpop.GetLength(0), OptimizationAlgorithms.MemeplexNumber(newpop.GetLength(0)), Fun);

Isk2 = OptimizationAlgorithms.Maximum(newpop, Fun);

Isk = OptimizationAlgorithms.Minimum(newpop, Fun);

switch ((int)KolZnakov.Value)

{

case 0:

BestX.Text = String.Format("Координата Х: {0:f0}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f0}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f0}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f0}", Isk2[1]);

break;

case 1:

BestX.Text = String.Format("Координата Х: {0:f1}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f1}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f1}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f1}", Isk2[1]);

break;

case 2:

BestX.Text = String.Format("Координата Х: {0:f2}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f2}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f2}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f2}", Isk2[1]);

break;

case 3:

BestX.Text = String.Format("Координата Х: {0:f3}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f3}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f3}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f3}", Isk2[1]);

break;

case 4:

BestX.Text = String.Format("Координата Х: {0:f4}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f4}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f4}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f4}", Isk2[1]);

break;

case 5:

BestX.Text = String.Format("Координата Х: {0:f5}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f5}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f5}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f5}", Isk2[1]);

break;

case 6:

BestX.Text = String.Format("Координата Х: {0:f6}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f6}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f6}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f6}", Isk2[1]);

break;

case 7:

BestX.Text = String.Format("Координата Х: {0:f7}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f7}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f7}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f7}", Isk2[1]);

break;

case 8:

BestX.Text = String.Format("Координата Х: {0:f8}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f8}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f8}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f8}", Isk2[1]);

break;

case 9:

BestX.Text = String.Format("Координата Х: {0:f9}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f9}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f9}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f9}", Isk2[1]);

break;

case 10:

BestX.Text = String.Format("Координата Х: {0:f10}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f10}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f10}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f10}", Isk2[1]);

break;

case 11:

BestX.Text = String.Format("Координата Х: {0:f11}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f11}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f11}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f11}", Isk2[1]);

break;

case 12:

BestX.Text = String.Format("Координата Х: {0:f12}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f12}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f12}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f12}", Isk2[1]);

break;

case 13:

BestX.Text = String.Format("Координата Х: {0:f13}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f13}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f13}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f13}", Isk2[1]);

break;

case 14:

BestX.Text = String.Format("Координата Х: {0:f14}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f14}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f14}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f14}", Isk2[1]);

break;

case 15:

BestX.Text = String.Format("Координата Х: {0:f15}", Isk[0]);

BestY.Text = String.Format("Координата Y: {0:f15}", Isk[1]);

WorstX.Text = String.Format("Координата Х: {0:f15}", Isk2[0]);

WorstY.Text = String.Format("Координата Y: {0:f15}", Isk2[1]);

break;

}

}

}

//метод копирования массива

public void CopyLastMassiv()

{

for (int i = 0; i < newpop.GetLength(0); i++)

{

lastmassiv[i, 0] = newpop[i, 0];

lastmassiv[i, 1] = newpop[i, 1];

}

}

//конструктор формы

public Graphiks(int KolIt, double MiX, double MaX, double MiY, double MaY, bool MaxF, bool MinF, Functions.FunctionVyb x)

{

Kol = KolIt;//присваиваем переданные значения из другой формы

MaxX = MaX;

MinX = MiX;

MaxY = MaY;

MinY = MiY;

Max = MaxF;

Min = MinF;

Fun = x;

InitializeComponent();

Timer.Interval = 1000; //интервал для таймера

lastiteration.Enabled = false; //начальные параметры доступа некоторых объектов

Stop\_button.Enabled = false;

newpop = OptimizationAlgorithms.NewPopulation(MinX, MaxX, MinY, MaxY); //формируем новую популяцию

lastmassiv = new double[newpop.GetLength(0), 2]; //инициализируем массив для предыдущей итерации

Postroenye(); //строим график без точек

zedGraph.IsShowPointValues = true;// включаем показ всплывающих подсказок при наведении курсора на график

// для изменения формата представления координат обрабатываем событие для графика

zedGraph.PointValueEvent += new ZedGraphControl.PointValueHandler(zedGraph\_PointValueEvent);

GraphSupport.SetToolTip(zedGraph, "Красным цветом обозначается лучший агент, желтым - худший агент, зеленым - остальные агенты"); //подсказка для графика

}

//нажатие на запуск автовыполнения итераций

private void VypolnitAuto\_Click(object sender, EventArgs e)

{

Timer.Start(); //запускаем таймер

VypolnitAuto.Enabled = false; //меняем доступность кнопок

Stop\_button.Enabled = true;

lastiteration.Enabled = true;

}

//нажатие на кнопку следующей итерации

private void nextiteration\_Click(object sender, EventArgs e)

{

if (!flag) //случай, если до этого не была нажата кнопка предыдущая итерация

{

lastiteration.Enabled = true;

CopyLastMassiv(); //копируем предыдущий массив

ZnakVyvodNextIt(); //производим улучшения

newpop = OptimizationAlgorithms.PopulationShuffle(newpop); //тасуем популяцию

Points(newpop); //выводим точки

KolIterations++; //наращиваем количество итераций

IterNow.Text = KolIterations.ToString();

if (KolIterations == Kol) //если все итерации выполнены

{

Timer.Stop(); //останавливаем таймер

ResultVyvod(); //выводим результат

KolIterations = 0; //обнуляем количество итераций

nextiteration.Enabled = false; //меняем доступность кнопок

Stop\_button.Enabled = false;

VypolnitAuto.Enabled = false;

}

}

else

{ //случай, если до этого была нажата кнопка предыдущая итерация

lastiteration.Enabled = true;

ZnakVyvod(newpop); //все аналогично предыдущему случаю без улучшений

Points(newpop);

KolIterations++;

IterNow.Text = KolIterations.ToString();

if (KolIterations == Kol)

{

Timer.Stop();

ResultVyvod();

KolIterations = 0;

nextiteration.Enabled = false; //меняем доступность кнопок

Stop\_button.Enabled = false;

VypolnitAuto.Enabled = false;

}

flag = false;

}

}

//событие, когда закончился круг в таймере

private void Timer\_Tick(object sender, EventArgs e)

{

nextiteration.PerformClick(); //симулируем нажатие на следующую итерацию

IterNow.Text = KolIterations.ToString();

if (KolIterations == Kol)

{

Timer.Stop(); //останавливаем таймер

ResultVyvod(); //выводим результат

KolIterations = 0;

nextiteration.Enabled = false; //меняем доступность кнопок

Stop\_button.Enabled = false;

}

}

//нажатие на остановку автозапуска

private void Stop\_button\_Click(object sender, EventArgs e)

{

Timer.Stop(); //останавливаем таймер

VypolnitAuto.Enabled = true; //меняем доступность кнопок

Stop\_button.Enabled = false;

}

//изменение временного интервала автозапуска

private void TimeTrackBar\_Scroll(object sender, EventArgs e)

{

Timer.Interval = 1000 \* TimeTrackBar.Value; //меняем интервал таймера

}

//нажатие на кнопку предыдущая операция

private void lastiteration\_Click(object sender, EventArgs e)

{

flag = true;

nextiteration.Enabled = true; //меняем доступность кнопок

lastiteration.Enabled = false;

if (KolIterations != 0) //на случай если была выполнена последняя итерация

KolIterations--;

else

KolIterations = Kol - 1;

IterNow.Text = KolIterations.ToString();

ZnakVyvod(lastmassiv); //выводим максимум/минимум

Points(lastmassiv); //строим точки предыдущей итерации

}

//нажатие на кнопку "действующая популяция" - открытие новой формы

private void Population\_Click(object sender, EventArgs e)

{

PopulationShow newForm = new PopulationShow(KolIterations, newpop,Fun);

newForm.ShowDialog();

}

//изменение количества знаков

private void KolZnakov\_ValueChanged(object sender, EventArgs e)

{

ZnakVyvod(newpop);

}

//нажатие на кнопку формирование новой популяции

private void NewPopulation\_Click(object sender, EventArgs e)

{

lastiteration.Enabled = false; //начальные параметры доступа некоторых объектов

Stop\_button.Enabled = false;

VypolnitAuto.Enabled = true;

nextiteration.Enabled = true;

Population.Enabled = true;

KolZnakov.Enabled = true;

newpop = OptimizationAlgorithms.NewPopulation(MinX, MaxX, MinY, MaxY); //формируем новую популяцию

lastmassiv = new double[newpop.GetLength(0), 2]; //инициализируем массив для предыдущей итерации

Postroenye(); //строим график без точек

KolIterations = 0; //обнуляем количество итераций

IterNow.Text = KolIterations.ToString(); //выводим на форму

}

//событие наведение курсора на точку

private string zedGraph\_PointValueEvent(ZedGraphControl sender, GraphPane graph, CurveItem curve, int ind)

{

PointPair point = curve[ind]; //получаем точку по индексу

string result = string.Format(@"Координата X: {0}

Координата Y: {1}", point.X, point.Y); //формирование строки

return result;

}

//нажатие на кнопку график функции

private void GraphFunOpen\_Click(object sender, EventArgs e)

{

newForm = new FunctionGraph(Fun, MinX, MaxX, MinY, MaxY);

newForm.Show(); //показываем форму

}

}

}

# КОД ФАЙЛА PopulationShow.cs

using System;

using System.Windows.Forms;

using Библиотека\_классов;

namespace Визуализация

{

public partial class PopulationShow : Form

{

double[,] newpop; //переменные для возможности использовать вне конструктора

Functions.FunctionVyb Fun;

public PopulationShow(int Kol, double[,] newpop, Functions.FunctionVyb Fun)

{//конструктор формы

InitializeComponent();

this.newpop = newpop;

this.Fun = Fun; //присваивание внешним переменным

Iteration.Text = Kol.ToString(); //задание количества итераций

Population.RowCount = newpop.GetLength(0);

Population.ColumnCount = 3;

Population.Columns[0].HeaderText = "Координата Х"; //именуем столбцы

Population.Columns[1].HeaderText = "Координата Y";

Population.Columns[2].HeaderText = "Значение функции";

for (int i = 0; i < newpop.GetLength(0); i++)

{ //заполняем таблицу

switch ((int)KolZnakov.Value)

{ //в зависимости от выбранного количества знаков, заполняем таблицу

case 0:

Population[0, i].Value = string.Format("{0:f0}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f0}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f0}",Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 1:

Population[0, i].Value = string.Format("{0:f1}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f1}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f1}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 2:

Population[0, i].Value = string.Format("{0:f2}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f2}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f2}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 3:

Population[0, i].Value = string.Format("{0:f3}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f3}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f3}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 4:

Population[0, i].Value = string.Format("{0:f4}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f4}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f4}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 5:

Population[0, i].Value = string.Format("{0:f5}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f5}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f5}",Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 6:

Population[0, i].Value = string.Format("{0:f6}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f6}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f6}",Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 7:

Population[0, i].Value = string.Format("{0:f7}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f7}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f7}",Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 8:

Population[0, i].Value = string.Format("{0:f8}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f8}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f8}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 9:

Population[0, i].Value = string.Format("{0:f9}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f9}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f9}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 10:

Population[0, i].Value = string.Format("{0:f10}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f10}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f10}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 11:

Population[0, i].Value = string.Format("{0:f11}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f11}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f11}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 12:

Population[0, i].Value = string.Format("{0:f12}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f12}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f12}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 13:

Population[0, i].Value = string.Format("{0:f13}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f13}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f13}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 14:

Population[0, i].Value = string.Format("{0:f14}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f14}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f14}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 15:

Population[0, i].Value = string.Format("{0:f15}",newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f15}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f15}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

}

}

}

private void KolZnakov\_ValueChanged(object sender, EventArgs e)

{ //производим изменения, при изменении количества знаков

for (int i = 0; i < newpop.GetLength(0); i++)

{ //заполняем таблицу

switch ((int)KolZnakov.Value)

{ //в зависимости от количества знаков заполняем таблицу

case 0:

Population[0, i].Value = string.Format("{0:f0}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f0}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f0}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 1:

Population[0, i].Value = string.Format("{0:f1}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f1}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f1}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 2:

Population[0, i].Value = string.Format("{0:f2}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f2}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f2}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 3:

Population[0, i].Value = string.Format("{0:f3}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f3}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f3}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 4:

Population[0, i].Value = string.Format("{0:f4}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f4}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f4}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 5:

Population[0, i].Value = string.Format("{0:f5}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f5}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f5}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 6:

Population[0, i].Value = string.Format("{0:f6}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f6}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f6}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 7:

Population[0, i].Value = string.Format("{0:f7}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f7}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f7}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 8:

Population[0, i].Value = string.Format("{0:f8}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f8}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f8}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 9:

Population[0, i].Value = string.Format("{0:f9}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f9}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f9}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 10:

Population[0, i].Value = string.Format("{0:f10}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f10}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f10}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 11:

Population[0, i].Value = string.Format("{0:f11}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f11}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f11}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 12:

Population[0, i].Value = string.Format("{0:f12}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f12}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f12}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 13:

Population[0, i].Value = string.Format("{0:f13}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f13}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f13}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 14:

Population[0, i].Value = string.Format("{0:f14}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f14}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f14}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

case 15:

Population[0, i].Value = string.Format("{0:f15}", newpop[i, 0]);

Population[1, i].Value = string.Format("{0:f15}", newpop[i, 1]);

Population[2, i].Value = string.Format("{0:f15}", Fun(newpop[i, 0], newpop[i, 1]));

Population.Rows[i].HeaderCell.Value = String.Format("{0}", i + 1);

break;

}

}

}

}

}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Изм. | Номера листов | | | | Всего листов (страниц) в докум. | № докум. | Входящий №  сопроводительного докум. и дата | Подп. | Дата |
| изме-ненных | заме-ненных | новых | анну-лиро-ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |