Содержание

[Введение 3](#_Toc25572636)

[1. Анализ объекта 4](#_Toc25572637)

[1.1 Описание предметной области. 4](#_Toc25572638)

[1.2 Построение концептуальной модели предметной области. 7](#_Toc25572639)

[2 Постановка задачи 8](#_Toc25572640)

[2.1 Определение требований к программной системе. 8](#_Toc25572641)

[2.2 Описание аналогов системы. 9](#_Toc25572642)

[2.3 Обзор и обоснование выбора средств реализации 10](#_Toc25572643)

[3 Проектирование 11](#_Toc25572644)

[3.1 Разработка архитектуры программного продукта 11](#_Toc25572645)

[3.2 Проектирование структур хранения данных. 11](#_Toc25572646)

[4 Реализация 12](#_Toc25572647)

[4.1 Разработка архитектуры программного продукта 12](#_Toc25572648)

[4.2 Разработка интерфейса программного продукта 14](#_Toc25572649)

[Разработка алгоритмов реализации вариантов использования. 18](#_Toc25572650)

[5. Тестирование 20](#_Toc25572651)

[Литература 23](#_Toc25572652)

[Приложение А 24](#_Toc25572653)

# Введение

Решение оптимизационной задачи в общем случае заключается в определении таких значений входных переменных исследуемого объекта, которым соответствует наилучшее (минимальное или максимальное) значение целевой функции. Технологические системы, как правило, являются многомерными, с большим количеством входных факторов, на значение которых к тому же накладываются дополнительные ограничения. Это требует использования методов многомерной условной оптимизации.

Данная курсовая работа представляет собой приложение для решения многомерных оптимизационных задач методом циклического покоординатного спуска.

# Анализ объекта

# 1.1 Описание предметной области.

Один из классификационных признаков делит оптимизационные задачи на два класса: *задачи безусловной оптимизации* и *задачи условной оптимизации*. Первые из них характеризуются тем, что минимум функции *f*: **R***m* → **R** ищется на всем пространстве:

|  |  |
| --- | --- |
| *f*(*x*) → min,   *x*  **R***m*. | (2) |

В задачах же второго класса поиск минимума идет на некотором собственном подмножестве Ω пространства **R***m*:

|  |  |
| --- | --- |
| *f*(*x*) → min,   *x*  Ω. | (3) |

Множество Ω часто выделяется *ограничениями типа равенств*

|  |  |
| --- | --- |
| *g*0(*x*) = Θ, | (4) |

где *g*0: **R***m* → **R***k*, и/или *ограничениями типа неравенств*

|  |  |
| --- | --- |
| *g*1(*x*) ≤ Θ, | (5) |

где *g*1: **R***m* → **R***l*.

Другой классификационный признак задач оптимизации — свойства функций *f* и множеств Ω. Например задачи (2) и (3) называются *линейными* (часто говорят о *задачах линейного программирования*), если функция *f* — аффинная, а множество Ω — многогранное (множество Ω называется *многогранным*, если оно выделяется ограничениями вида (4) и (5) с аффинными функциями *g*0 и *g*1).

**Замечание**. Линейная задача безусловной оптимизации (1) имеет решение (причем обязательно неединственное) в том и только том случае, если *f*(*x*) ≡ const.

Если функции *f*, *g*0 и *g*1 квадратичные, то говорят о *задачах квадратичного программирования* или о *квадратичных задачах оптимизации* (условных или безусловных). Если эти функции выпуклые, то говорят о *задачах выпуклого программирования* (если множество Ω задается каким-либо другим образом, а не только ограничениями типа (4) и (5), то в задачах выпуклого программирования требуют его выпуклость). Наконец, в общем случае говорят о *задачах нелинейного программирования*. В таких задачах обычно предполагается гладкость фигурирующих в них функций.

# Метод покоординатного спуска

Основная идея этого метода заключается в том, что поиск точки минимума *х*\* сводится к поочередному изменению переменных вдоль одной из координатных осей:

*xir+1 = xir +* λ*irIi, i =* 1,2*, …, n*. (1)

где *Ii — i*-й координатный *n*-мерный вектор с компонентами:

*lij* = 1, если *i = j*;

*lij* = 0 – в противном случае.

Длина шага λ*ir* вдоль направления поиска *Ii* может выбираться равной некоторой постоянной величине Δ*i* по следующему правилу:

λ*ir* = Δ*i*, если Q(*xr* + Δ*iIi*) < Q(*xr*);

λ*ir* = -Δ*i*, если Q(*xr* – Δ*iIi*) < Q(*xr*) < Q(*xr* + Δ*iIi*). (2)

Если окажется, что λ*ir* = 0 для всех *i* = 1, 2, …, *n*, то длина пробных шагов Δ*i* должна быть уменьшена (Δ*i* = Δ*i*/β, где β > 1). Поиск считается законченным при выполнении условия:

max Δ*i* < ε. (3)

Алгоритм F29, реализующий описанную стратегию поиска точки минимума *x*\*, называется методом **покоординатного спуска с постоянным шагом**.

Когда длина шага λ*ir* на каждой итерации определяется с помощью одномерной задачи оптимизации

Q(*xr* + λ*irIi*) = min Q(*xr* + ∑λ*krIk* + λ*iIi*) (4)

приходим к алгоритму F30, реализующему **релаксационный метод Гаусса — Зейделя**, процедура поиска точки минимума *х*\* в котором сводится к следующей последовательности действий.

1. Задается начальное приближение *хr=х*°.  
2. Осуществляется циклический покоординатный спуск из точки  
*хr* по формуле (1) с выбором длины шага λ*kr*, из условия (4) для  
всех *i* от 1 до *n*. Эта процедура образует внутренний цикл, в процессе которого осуществляется одномерная минимизация функции Q (*х*) по каждой переменной:

min Q(*х1r, …, хi-1r, xi, хi+1r, …, хnr*), *i* = 1, 2, …, *n*.

3. После окончания внутреннего цикла в качестве начального приближения *х*° принимается точка *хn* и все вычисления повторяются с п. 2.

4. Поиск точки минимума *х*\* заканчивается, если после очередного внутреннего цикла выполняется условие ||*хr – хn*|| < ε.

Геометрической интерпретацией траектории поиска, которая получается по алгоритмам F29 и F30 является ломаная, состоящая из отрезков прямых, параллельных осям координат.

Недостатком методов покоординатного спуска (алгоритмы F29 и F30) является то, что при минимизации функций, имеющих овраг, дно которого не ориентировано вдоль какой-то из координатных осей, процесс поиска сильно замедляется и может остановиться далеко от точки истинного минимума *x*\*.

В связи с этим рассмотрим алгоритм F31, реализующий **метод конфигураций**, который позволяет осуществлять поиск вдоль произвольно ориентированного относительно координатных осей дна оврага.

## 1.2 Построение концептуальной модели предметной области.

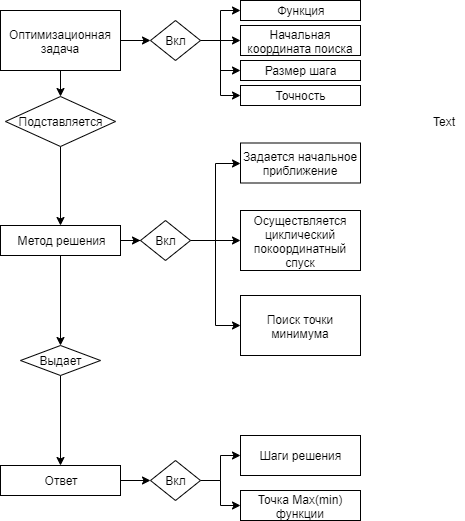


Рисунок 1.1 – модель предметной области

Концептуальная модель состоит задачи оптимизации, данные которой пользователю необходимо ввести (функция, начальная координата поиска, точность, размер шага). После ввода пользователь нажимает кнопку «Решить», и данные задачи подставляются в метод решения. Ход решения отображается в специальном окне. Как только задача решена, метод выдает пользователю ответ

# Постановка задачи

## 2.1 Определение требований к программной системе.

1. Приложение, позволяющее решить уравнение методом циклического покоординатного спуска
2. Поддержка ввода данных как с экрана, так и с файла
3. Поддержка вывода данных как на экран, так и в файл
4. Дружелюбный интерфейс
5. Поддержка Java версии 9 и более

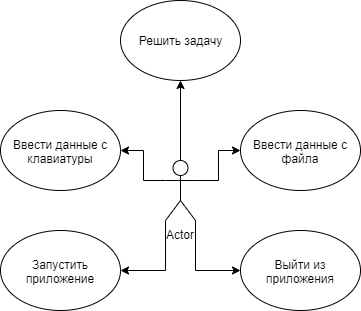


Рисунок 2.1 - диаграмма вариантов использования

## 2.2 Описание аналогов системы.

В сети Интернет можно найти большое количество работ, которые реализуют решение данных задач. Практически все они представляют собой сайты, так что для решения уравнений необходимо подключение к интернету.

Мой курсовой проект представляет собой приложение для ПК. Программа не требует подключения к интернету и имеет функции загрузки данных в файл.

## Обзор и обоснование выбора средств реализации

Данная курсовая работа выполнена на языке программирования высокого уровня C# с использованием компилятора Microsoft Visual Studio Community 2015.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений

Достоинства C#:

* C# создавался параллельно с каркасом Framework .Net и в полной мере учитывает все его возможности - как FCL, так и CLR;
* C# является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
* C# является мощным объектным языком с возможностями наследования и универсализации;
* C# является наследником языков C/C++, сохраняя лучшие черты этих популярных языков программирования. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от С++ к C#;
* Сохранив основные черты своего великого родителя, язык стал проще и надежнее. Простота и надежность, главным образом, связаны с тем, что на C# хотя и допускаются, но не поощряются такие опасные свойства С++ как указатели, адресация, разыменование, адресная арифметика;
* Благодаря каркасу Framework .Net, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;
* Мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных;
* Реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#.

# Проектирование

## 3.1 Разработка архитектуры программного продукта

Главное окно будет содержать следующие элементы:

* Поле для ввода функции
* Поля для ввода начальной координаты поиска
* Поля для ввода шагов поиска
* Поле точности
* Кнопка «пример» - для вывода примера
* Кнопка «решить» - для решения задачи
* Кнопки «сохранить» и «загрузить» для работы с файлом
* Окно вывода подробного решения

## 3.2 Проектирование структур хранения данных.

База данных не используется. Для хранения данных извне используется обычный файл с расширением «.txt». Для хранения данных внутри приложения используются следующие переменные:

* double "f">Минимизируемая функция
* double "x0">начальной точки посика
* double "interval">интервалы поиска
* double "h">Векор положительных приращений координат
* double "eps0">Точность

# 4 Реализация

## 4.1 Разработка архитектуры программного продукта

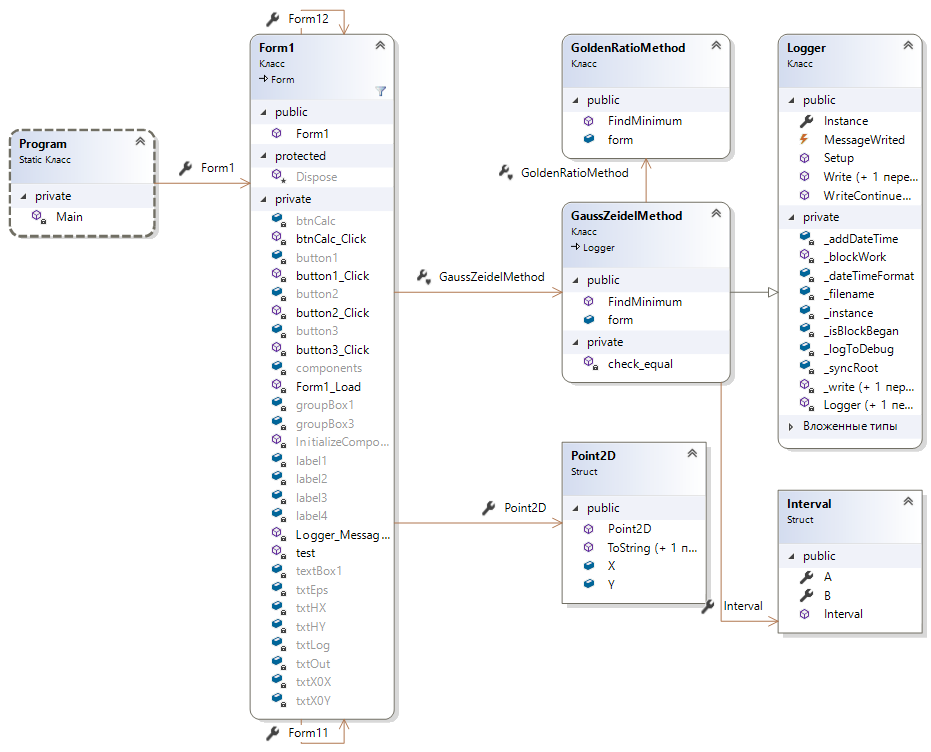


Рисунок 4.1 – Диаграмма классов

Запускаемым классом является Program. При запуске он выводит на экран Form1, куда пользователь заносит данные. Как только данные введены, создается точка Point2D с начальной координатой поиска, функция и эта точка передаются в класс GausseideMethod, реализующий метод покоординатного спуска. Также, в этом классе при помощи Loggerа пользователю выводится пошаговое решение.

## 4.2 Разработка интерфейса программного продукта

При запуске приложения, пользователю будет показано следующее

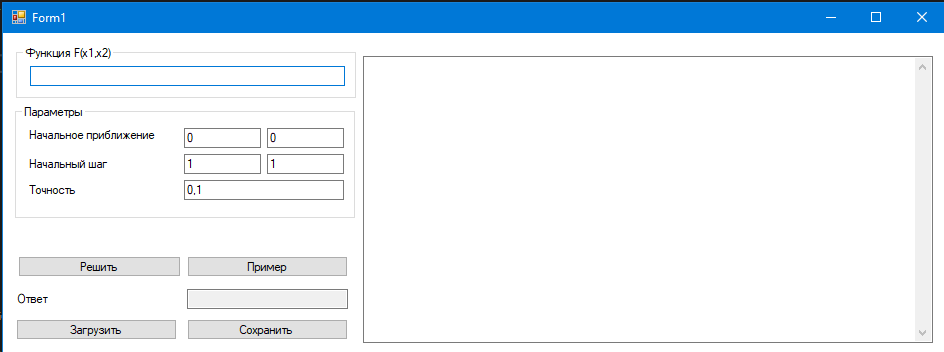


Рисунок 4.2.1 – Вид главного окна

Ввод функции

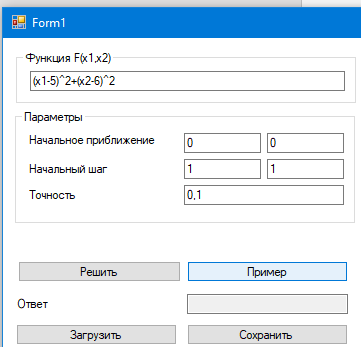


Рисунок 4.2.2 – Ввод функции

На случай неверно введенных данных, предусмотрен вывод соответствующего предупроеждения

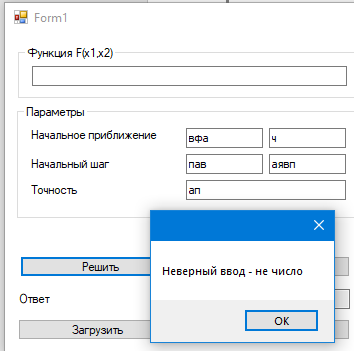


Рисунок 4.2.3 – Сообщение об ошибке

Результат будет выводиться в правом окне приложения

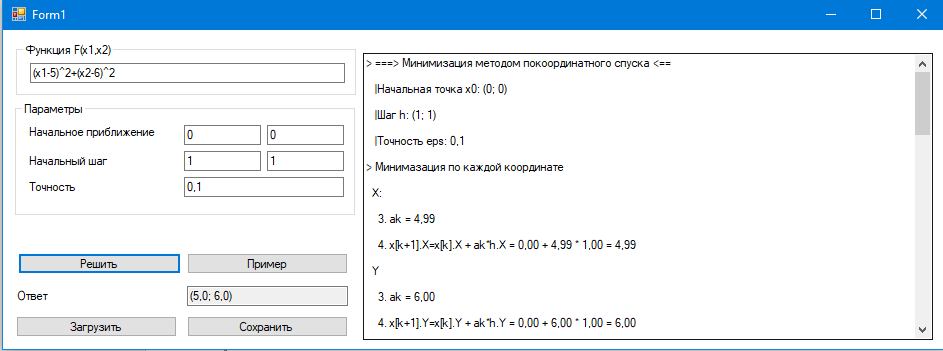


Рисунок 4.2.4 – Ввод результата

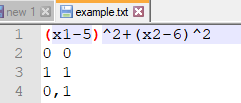


Рисунок 4.2.5 – Вид файла для чтения

Разработка алгоритмов реализации вариантов использования.

Общий алгоритм работы приложения:

1. Открывается главное меню, переход к 2 или
2. Вводит данных задачи, переход к 3
3. Кнопка загрузить, ввод данных из файла, переход к 4
4. Кнопка решить, нахождение решения, переход к 5
5. Вывод результатов решения в поле, переход к 6
6. Кнопка сохранить, сохранение данных в файл, переход к 7
7. Завершение работы приложения
8. Метод чтения данных из файла

private void button2\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";

try {

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

// получаем выбранный файл

using (StreamReader sr = new StreamReader(openFileDialog1.FileName, System.Text.Encoding.Default))

{

textBox1.Text = sr.ReadLine();

string[] xy0 = sr.ReadLine().Split(' ');

txtX0X.Text = xy0[0];

txtX0Y.Text = xy0[1];

xy0 = sr.ReadLine().Split(' ');

txtHX.Text = xy0[0];

txtHY.Text = xy0[1];

txtEps.Text = sr.ReadLine();

}

}

catch (Exception ex) {

MessageBox.Show("Ошибка чтения из файла");

}

}

1. Метод вызова метода решения

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

{

Point2D x0, h;

double eps = 0, x0x = 0, x0y = 0, hx = 0, hy = 0;

Point2D res;

//Парсинг параметров

try

{

x0x = double.Parse(txtX0X.Text);

x0y = double.Parse(txtX0Y.Text);

hx = double.Parse(txtHX.Text);

hy = double.Parse(txtHY.Text);

eps = double.Parse(txtEps.Text);

}

catch (FormatException exp)

{

MessageBox.Show("Неверный ввод - не число");

return;

}

x0 = new Point2D(x0x, x0y);

h = new Point2D(hx, hy);

//Определяем функцию

Func<double, double, double> f;

f = test;

res = GaussZeidelMethod.FindMinimum(f, x0, h, eps);

txtOut.Text = res.ToString(OptimizationUtils.GetFormat(eps));

}

1. Метод золотого сечения для нахождения решения одномерной задачи

public static Point2D FindMinimum(Func<double, double, double> f, Point2D x0, Point2D h, double eps0)

{

double finder\_h = 0.05; //Шаг поиска отрезка минимума

double z = 0.1; //Уменьшение шага поиска

double equal\_eps = eps0; //Совпадение

Point2D x = new Point2D(0, 0);

Func<double, double> f\_fixed;

Interval interval;

double ak;

Logger.Write("===> Минимизация методом покоординатного спуска <==");

Logger.WriteContinue(" |Начальная точка x0: {0}", x0);

Logger.WriteContinue(" |Шаг h: {0} ", h);

Logger.WriteContinue(" |Точность eps: {0} ", eps0);

string format = OptimizationUtils.GetFormat(eps0 / 10);

do

{

do

{

x0.X = x.X;

x0.Y = x.Y;

Logger.Write("Минимазация по каждой координате");

//Поочередная минимизация по каждой координате

//X

Logger.WriteContinue("X:");

f\_fixed = a => f(x0.X + a \* h.X, x0.Y);

interval = IntervalFinder.Find(x0.X, finder\_h, f\_fixed);

ak = GoldenRatioMethod.FindMinimum(interval, f\_fixed, eps0 / 5);

x.X = x0.X + ak \* h.X;

Logger.WriteContinue(" 3. ak = {0:" + format + "}", ak);

Logger.WriteContinue(" 4. x[k+1].X=x[k].X + ak\*h.X = {0:" + format + "} + {1:" + format + "} \* {2:" + format + "} = {3:" + format + "}", x0.X, ak, h.X, x.X);

//Y

Logger.WriteContinue("Y");

f\_fixed = a => f(x.X, x0.Y + a \* h.Y);

interval = IntervalFinder.Find(x0.Y, finder\_h, f\_fixed);

ak = GoldenRatioMethod.FindMinimum(interval, f\_fixed, eps0 / 5);

x.Y = x0.Y + ak \* h.Y;

Logger.WriteContinue(" 3. ak = {0:" + format + "}", ak);

Logger.WriteContinue(" 4. x[k+1].Y=x[k].Y + ak\*h.Y = {0:" + format + "} + {1:" + format + "} \* {2:" + format + "} = {3:" + format + "}", x0.Y, ak, h.Y, x.Y);

}

while (!check\_equal(x0, x, equal\_eps));

Logger.Write("x[k+1]==x[k]");

var h0 = h;

if (form.radioButton1.Checked && OptimizationUtils.EuclidNorm(h) <= eps0) break;

if (form.radioButton2.Checked && OptimizationUtils.EuclidNorm(h) >= eps0) break;

//Уменьшаем шаг

h.X = z \* h.X;

h.Y = z \* h.Y;

Logger.WriteContinue("9. Уменьшаем шаг: h = {0:" + format + "} \* {1:" + format + "} = {2:" + format + "}", z, h0, h);

Logger.WriteContinue("Переход на шаг 3");

} while (true);

Logger.Write("||h|| = {0:" + format + "} < eps", OptimizationUtils.EuclidNorm(h));

Logger.Write("Минимум(Максимум) найден: {0:" + format + "}", x);

return x;

}

# Тестирование

Проведем тестирование программы, пройдя все варианты взаимодействия. Запустим само приложение. Запуск происходит без проблем.

Попробуем найти решение с ошибочными значениями.

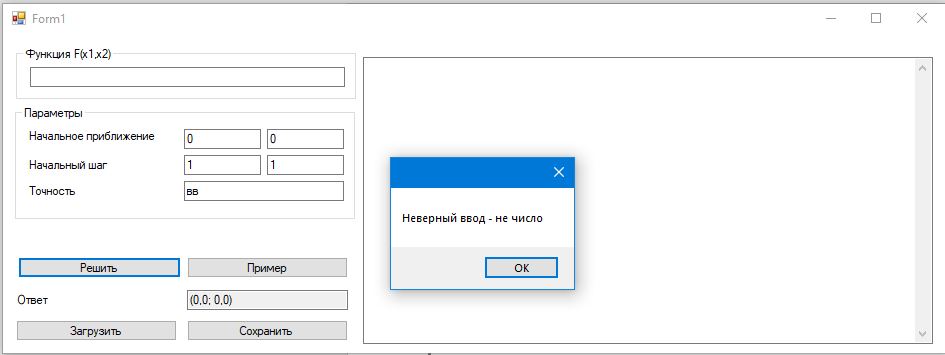


Рисунок 5.1 – Запуск главного окна

Проверим работу чтения из файла

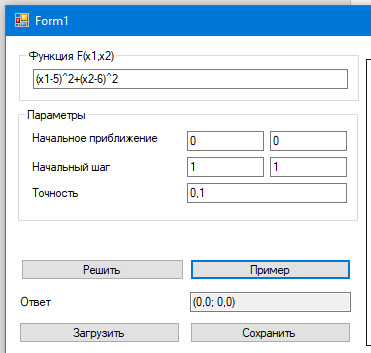


Рисунок 5.2 – Попытка считать значения из файла

Найдем решение данной задачи

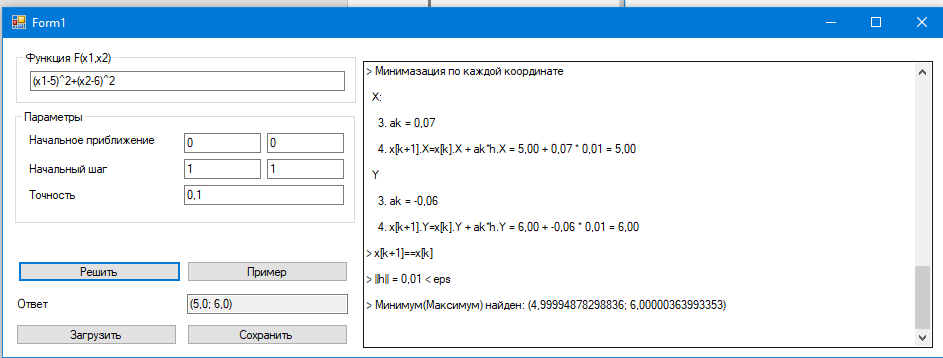


Рисунок 5.3 – Проверка решения задачи

Сохраним решение в файл. Сохранено успешно

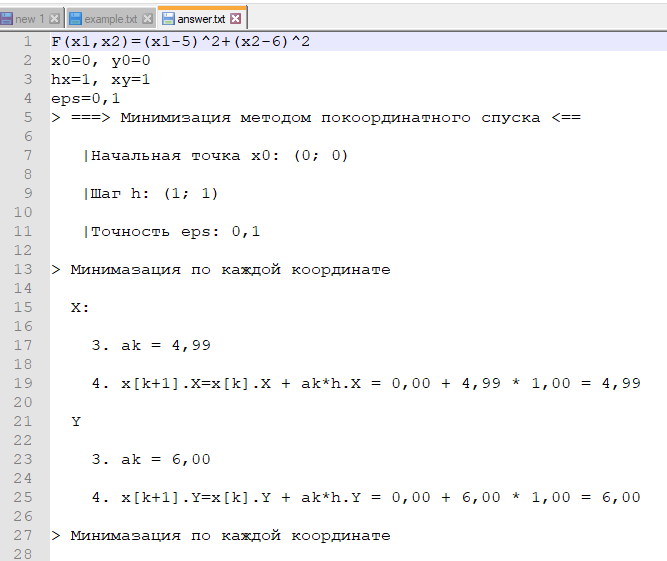


Рисунок 5.2 – Сохраненный ответ

Работа приложения происходит без ошибок

# Литература

1. Герберт, Ш.Г. C#3.0 Руководство для начинающих / Герберт, Ш.Г. – издательский дом Вильямс, 2009. – 688 с.
2. Постнов, К.В. Компьютерная графика: учеб. курс: М.: МГСУ. 2009. - 249с
3. Боресков, А.В. Компьютерная графика: Учебник и практикум для прикладного бакалавриата / А.В. Боресков, Е.В. Шикин. - Люберцы: Юрайт, 2016. - 219 c.
4. Visual C# 2008 Базовый курс/К. Уотсон [и др.]; под общ. ред. К. Нейгеля. – издательский дом Вильямс, 2009. – 1216 с.
5. С# 2008 и платформа .NET 3.5 для профессионалов/ К. Нейгел [и др.]; под общ. ред. Кристиан Нейгеля. – издательский дом Вильямс, 2008. – 1392 с.

# Приложение А

Исходный код, откомпилированная программа и записка находятся на диске.