МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО

ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**«Вятский государственный университет»**

**(ФГБОУ ВО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Допущен к защите

Руководитель проекта

\_\_\_\_\_\_\_\_/Исупов К.С./

(подпись) (Ф.И.О.)

«\_\_\_» \_\_\_\_\_\_\_\_\_202\_г.

Разработка программы визуализации решения линейного программирования графическим методом

Пояснительная записка курсового проекта по дисциплине

«Комплекс знаний бакалавра в области программного и аппаратного обеспечения вычислительно техники»

ТПЖА 090301.387 ПЗ

Разработал студент группы ИВТб-2301-04-00 \_\_\_\_\_\_\_\_\_\_\_\_/Жеребцов К. А./

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_/Исупов К.С./

Консультант \_\_\_\_\_\_\_\_\_\_/Коржавина А.С./

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_/

\_­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_/

Киров 2021



Реферат

Жеребцов К. А. Разработка программы визуализации решения линейного программирования графическим методом. ТПЖА.090301.387 ПЗ: Курс. проект/ВятГУ, каф. ЭВМ; рук. Исупов К. С. – Киров, 2021. ПЗ 42 c., 12 рис., 1 табл., 4 прил.

Объект исследования и разработки курсового проекта – графический метод решения задач линейного программирования.

Предмет курсового проекта –программа, которая визуализирует решение описанной в ней задачи линейного программирования графическим методом.

Целью курсового проекта является повышение эффективности изучения графического метода решения задач линейного программирования.

Результатом выполнения курсового проекта является программа визуализации решения задач линейного программирования графическим методом.

**Оглавление**

[Введение 6](#_Toc91505617)

[1. Обзор предметной области и постановка задачи 7](#_Toc91505618)

[1.1 Линейное программирование 7](#_Toc91505619)

[1.2 Графический метод решения задач линейного программирования 9](#_Toc91505620)

[1.3 Методика решения задач линейного программирования графическим методом 11](#_Toc91505621)

[1.4 Анализ существующих аналогов 12](#_Toc91505622)

[1.4.1 Microsoft Excel 13](#_Toc91505623)

[1.4.2 Mathcad 14](#_Toc91505624)

[1.4.3 GeoGebra 16](#_Toc91505625)

[1.4.4 Итоговая таблица сравнения аналогов 17](#_Toc91505626)

[1.5 Постановка задачи 18](#_Toc91505627)

[2. Разработка структуры приложения и алгоритмов 19](#_Toc91505628)

[2.1 Разработка алгоритма 20](#_Toc91505629)

[3. Программная реализация и отладка 21](#_Toc91505630)

[3.1 Выбор и обоснование языка программирования 21](#_Toc91505631)

[3.2 Разработка интерфейса пользователя 22](#_Toc91505632)

[3.3 Реализация программы 24](#_Toc91505633)

[3.4 Отладка программы 24](#_Toc91505634)

[3.4.1 Пример №1 25](#_Toc91505635)

[3.4.2 Пример №2 25](#_Toc91505636)

[3.4.3 Пример №3 26](#_Toc91505637)

[3.4.4 Итог 27](#_Toc91505638)

[Заключение 28](#_Toc91505639)

[Приложение А Схема алгоритма 29](#_Toc91505640)

[Приложение Б Структура программы 31](#_Toc91505641)

[Приложение В Исходный код 32](#_Toc91505642)

[Приложение Г Список используемой литературы 42](#_Toc91505643)

Введение

Линейное программирование - это наука о методах исследования и отыскания наибольших и наименьших значений линейной функции, на неизвестные которой наложены линейные ограничения [1]. Таким образом, задачи линейного программирования относятся к задачам на условный экстремум функции. Казалось бы, что для исследования линейной функции многих переменных на условный экстремум достаточно применить хорошо разработанные методы математического анализа, однако невозможность их использования можно довольно просто проиллюстрировать.

Для решения задач линейного программирования потребовалось создание специальных методов. В данном курсовом проекте будет рассмотрен графический метод решения задач линейного программирования. Графический метод применяется в основном при решении задач двумерного пространства и только некоторых задач трехмерного пространства, так как довольно трудно построить многогранник решений, который образуется в результате пересечения полупространств. Задачу пространства размерности больше трех изобразить графически вообще невозможно.

Данный курсовой проект является актуальным, потому как графический метод существенно нагляднее и обычно проще для понимания чем другой не менее распространённый способ – симплекс-метод, но занимает много времени, так как требует тщательного построения чертежа. Программа, разработанная в ходе выполнения курсового проекта, устраняет этот недостаток.

Таким образом, целью данной курсового проекта является повышение эффективности изучения графического метода решения задач линейного программирования.

1. Обзор предметной области и постановка задачи

1.1 Линейное программирование

****Линейное программирование – это один из первых и наиболее подробно изученных разделов математического программирования. Именно линейное программирование явилось тем разделом, с которого начала развиваться сама дисциплина «математическое программирование» [2].

Можно сказать, что линейное программирование применимо для построения математических моделей тех процессов, в основу которых может быть положена гипотеза линейного представления реального мира: экономических задач, задач управления и планирования, оптимального размещения оборудования и пр.

Задачами линейного программирования называются задачи, в которых линейны как целевая функция, так и ограничения в виде равенств и неравенств. Кратко задачу линейного программирования можно сформулировать следующим образом: найти вектор значений переменных, доставляющих экстремум линейной целевой функции при m ограничениях в виде линейных равенств или неравенств [3].

Основная задача линейного программирования (ОЗЛП) ставится следующим образом: Имеется ряд переменных x1, x2, …, xn. Требуется найти такие их неотрицательные значения, которые удовлетворяли бы системе линейных уравнений [3]:

a11 \* x1 + a12 \* x2 + …. + a1n \* xn = b1

a21 \* x1 + a22 \* x2 + …. + a2n \* xn = b2

……………………………………….

am1 \* x1 + am2 \* x2 + …. + amn \* xn = bm

и, кроме того, обращали бы в минимум линейную целевую функцию (ЦФ)

L = c1 \* x1 + c2 \* x2 + … + cn \* xn

Очевидно, случай, когда ЦФ нужно обратить не в минимум, а в максимум, легко сводится к предыдущему, если изменить знак функции и рассмотреть вместо нее функцию

L` = - L = -c1 \* x1 - c2 \* x2 - … - cn \* xn

Допустимым решением основной задачи линейного программирования называют любую совокупность переменных x1 ≥ 0, x2 ≥ 0, …, xn ≥ 0, удовлетворяющую уравнениям (1.1) [3].

Оптимальным решением называют то из допустимых решений, при котором ЦФ обращается в минимум [3].

На практике ограничения в задаче линейного программирования часто заданы не уравнениями, а неравенствами. В этом случае можно перейти к основной задаче линейного программирования.

Рассмотрим задачу линейного программирования с ограничениями-неравенствами, которые имеют вид

a11 \* x1 + a12 \* x2 + …. + a1n \* xn ≥ b1

a21 \* x1 + a22 \* x2 + …. + a2n \* xn ≥ b2

……………………………………….

am1 \* x1 + am2 \* x2 + …. + amn \* xn ≥ bm

и являются линейно-независимыми. Последнее означает, никакое из них нельзя представить в виде линейной комбинации других. Требуется найти xi > 0, которые удовлетворяют неравенствам и обращают в минимум целевую функцию.

Введём уравнения:

y1 = a1i + bi

y2 = a2i + b2

……………..

ym = ami + bm

где y1, y2, …, ym - добавочные переменные, которые также, как и x1, x2, …, xn являются неотрицательными.

Таким образом, имеем общую задачу линейного программирования - найти неотрицательные x1, x2, …, xn и y1, y2, …, ym чтобы они удовлетворяли системе уравнений (1.3) и обращали в минимум L = [3].

1.2 Графический метод решения задач линейного программирования

Графический метод довольно прост и нагляден для решения задач линейного программирования с двумя переменными. Он основан на геометрическом представлении допустимых решений и ЦФ задачи.

Каждое из неравенств задачи линейного программирования (1.2) определяет на координатной плоскости (x1Ox2) некоторую полуплоскость, а система неравенств в целом – пересечение соответствующих плоскостей. Множество точек пересечения данных полуплоскостей называется областью допустимых решений (ОДР). ОДР всегда представляет собой выпуклую фигуру, т.е. обладающую следующим свойством: если две точки А и В принадлежат этой фигуре, то и весь отрезок АВ принадлежит ей. ОДР графически может быть представлена выпуклым многоугольником, неограниченной выпуклой многоугольной областью, отрезком, лучом, одной точкой. В случае несовместности системы ограничений задачи (1.2) ОДР является пустым множеством [4].

ЦФ L(x) = c1 \* x1 + c2 \* x2 при фиксированном значении L(x) = L определяет на плоскости прямую линию c1 \* x1 + c2 \* x2 = L. Изменяя значения L, мы получим семейство параллельных прямых, называемых линиями уровня.

Вектор С = (с1, c2) с координатами из коэффициентов ЦФ при x1 и x2 перпендикулярен к каждой из линий уровня (см. рис. 1). Направление вектора C совпадает с направлением возрастания ЦФ, что является важным моментом для решения задач. Направление убывания ЦФ противоположно направлению вектора C [5].

Суть графического метода заключается в следующем. По направлению (против направления) вектора C в ОДР производится поиск оптимальной точки x\* = (x1\*, x2\*). Оптимальной считается точка, через которую проходит линия уровня Lmax (Lmin), соответствующая наибольшему (наименьшему) значению функции L(x). Оптимальное решение всегда находится на границе ОДР, например, в последней вершине многоугольника ОДР, через которую пройдет целевая прямая, или на всей его стороне [5].

При поиске оптимального решения задач линейного программирования возможны следующие ситуации: существует единственное решение задачи; существует бесконечное множество решений; ЦФ не ограничена; область допустимых решений – единственная точка; задача не имеет решений [5].

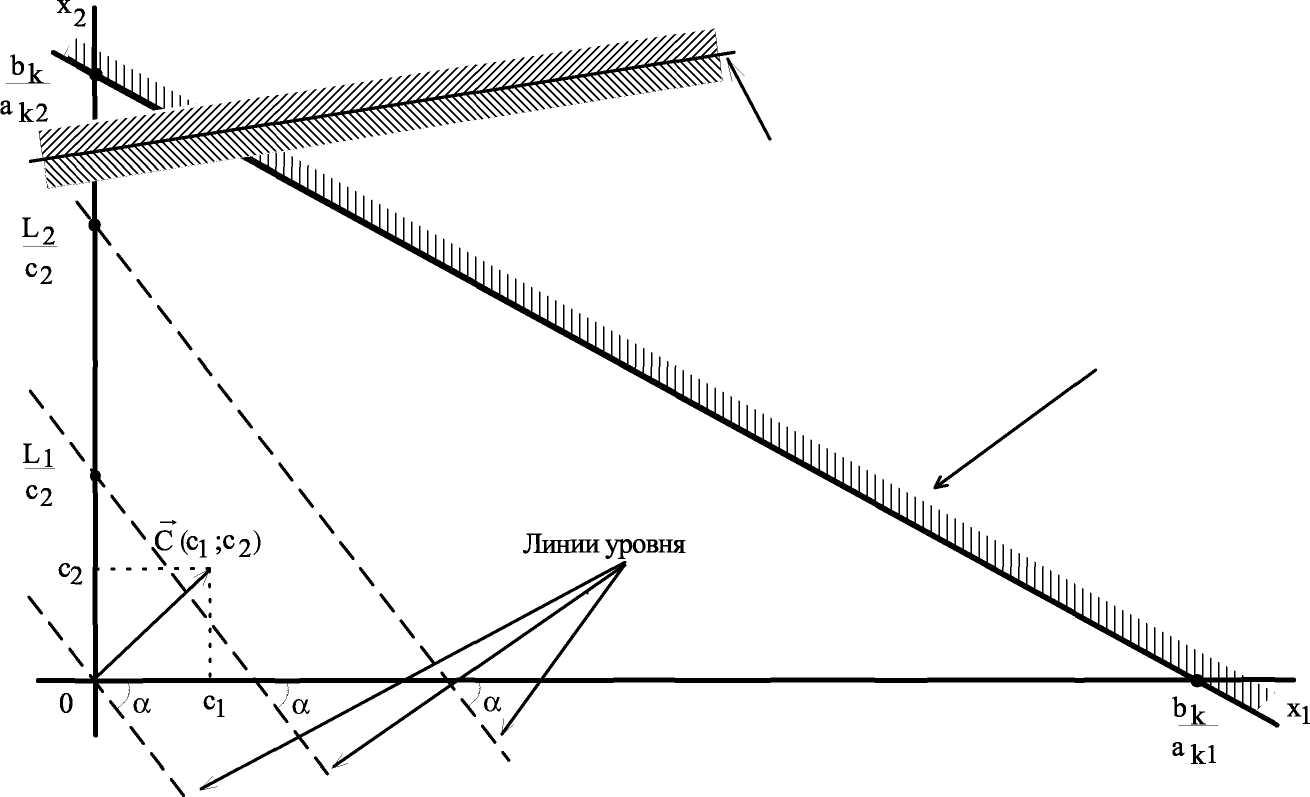


Рисунок 1 – Геометрическая интерпретация ограничений и ЦФ задачи

1.3 Методика решения задач линейного программирования графическим методом

На основе исследованной теоретической части был сформулирован следующий алгоритм решения задач линейного программирования графическим медом [4]:

1. В ограничениях задачи (1.2) заменить знаки неравенств знаками точных равенств и построить соответствующие прямые.
2. Найти и заштриховать полуплоскости, разрешенные каждым из ограничений – неравенств задачи (1.2). Для этого нужно подставить в конкретное неравенство координаты какой-либо точки, например, (0;0), и проверить истинность полученного неравенства.

Если неравенство истинное, то надо заштриховать полуплоскость, содержащую данную точку, иначе (неравенство ложное) надо заштриховать полуплоскость, не содержащую данную точку.

Ограничения-равенства разрешают только те точки, которые лежат на соответствующей прямой. Поэтому необходимо выделить на графике только сами прямые.

1. Определить область допустимых значений (ОДР) как часть плоскости, принадлежащую одновременно всем разрешающим областям, и выделить ее. При отсутствии ОДР задача не имеет решения.
2. Если область допустимых значений – не пустое множество, то нужно построить вектор С = (с1, с2), который начинается в точке (0;0) и заканчивается в точке (с1, с2).
3. Построить линию уровня, проходящую через точку (0;0), так, чтобы она находилась перпендикулярно вектору-градиенту.
4. Определить координаты точки максимума (минимума) х\* = (х1\*, х2\*) целевой функции и вычислить ее значение в этой точке.

1.4 Анализ существующих аналогов

Наиболее популярными приложениями для решения задач линейного программирования графическим медом являются Microsoft Excel и Mathcad. Также существует менее известный инструмент для построения графиков – GeoGebra. Оценка по всем критериям будет производиться по 5-ти балльной шкале, где 1 - плохо, а 5 – отлично. Сравнение будет производиться по следующим критериям:

1. Внешний вид – оценка визуального исполнения интерфейса пользователя
2. Удобство – оценка удобства ввода данных и формат результата
3. Функционал – оценка доступных пользователю функций для работы.

1.4.1 Microsoft Excel

Microsoft Excel — программа для работы с электронными таблицами, созданная корпорацией Microsoft для Microsoft Windows, Windows NT и Mac OS, а также Android, iOS и Windows Phone. Она предоставляет возможности экономико-статистических расчетов, графические инструменты и язык макропрограммирования VBA (Visual Basic for Application).

Так как Microsoft Excel в большей степени ориентирован на работу с таблицами, его интерфейс ориентирован на это. Имеет достаточно приятный внешний вид, оценка – 5.

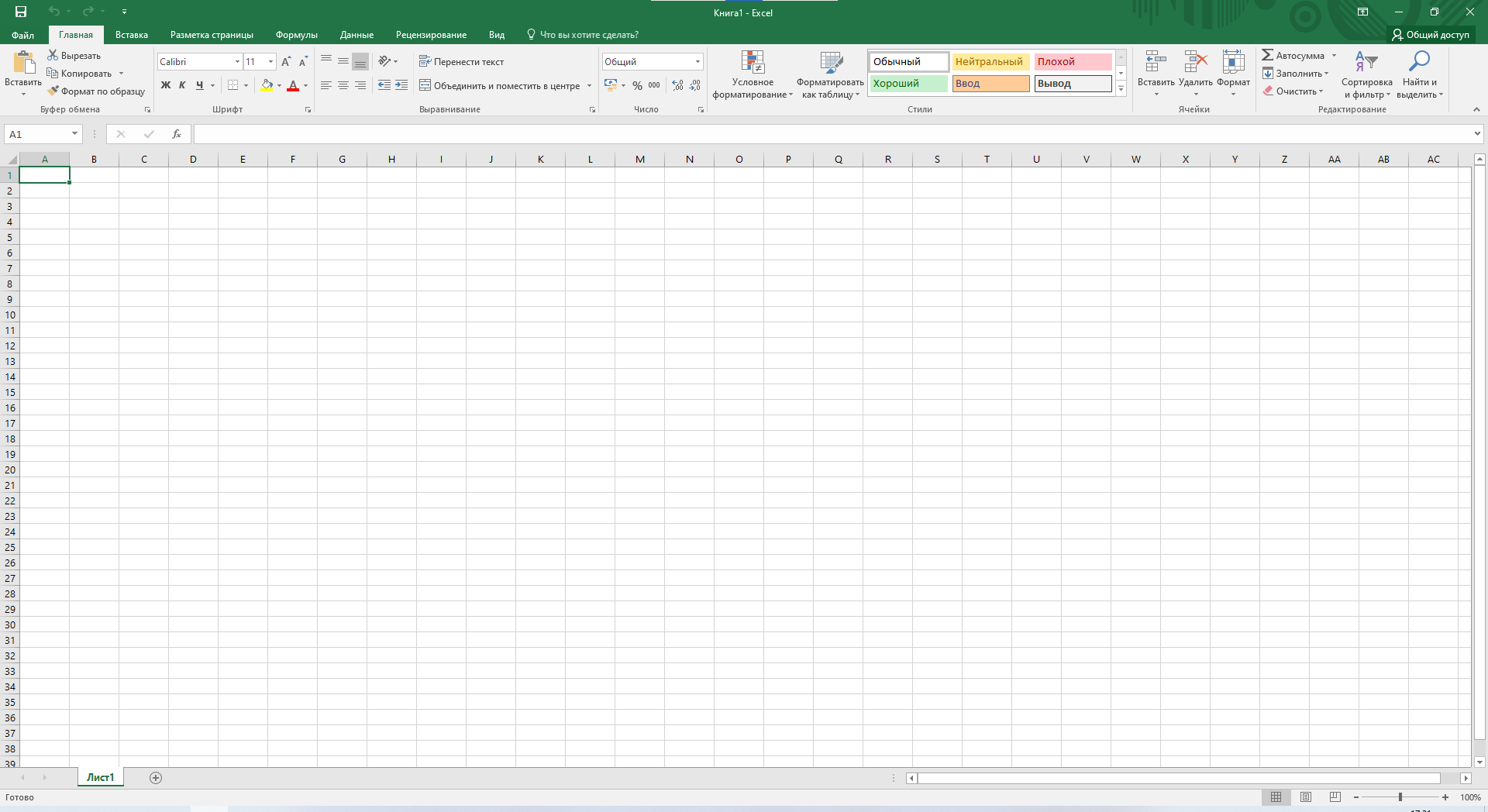


Рисунок 2 – Пользовательский интерфейс MS Excel

Что касается удобства задания исходных данных для решения задач линейного программирования, то в MS Excel нужно вручную определять формулы и ячейки, которые будут использоваться при расчетах. Также процесс расчета не интуитивный. Оценка – 3.

Функционал данного приложения достаточно широк:

1. автоматизация всех итоговых вычислений
2. над большими наборами данных можно производить однотипные расчёты
3. можно разрешать задачи с помощью подбора значений с различными параметрами
4. можно обрабатывать результаты экспериментов
5. производить табулирование функций и формул
6. подготавливать табличные документы
7. проводить поиск наиболее оптимальных значений для выбранных параметров
8. строить графики и диаграммы по уже введённым данным

Но если сравнивать MS Excel с тем же Mathcad, то можно заметить, что у второго функционал с точки зрения математики намного больше. Оценка – 4.

1.4.2 Mathcad

Mathcad — система компьютерной алгебры из класса систем автоматизированного проектирования, ориентированная на подготовку интерактивных документов с вычислениями и визуальным сопровождением, отличается лёгкостью использования и применения для коллективной работы.

Mathcad имеет интуитивный и простой для использования интерфейс пользователя. Для ввода формул и данных можно использовать как клавиатуру, так и специальные панели инструментов. Внешний вид – 4, удобство – 5.

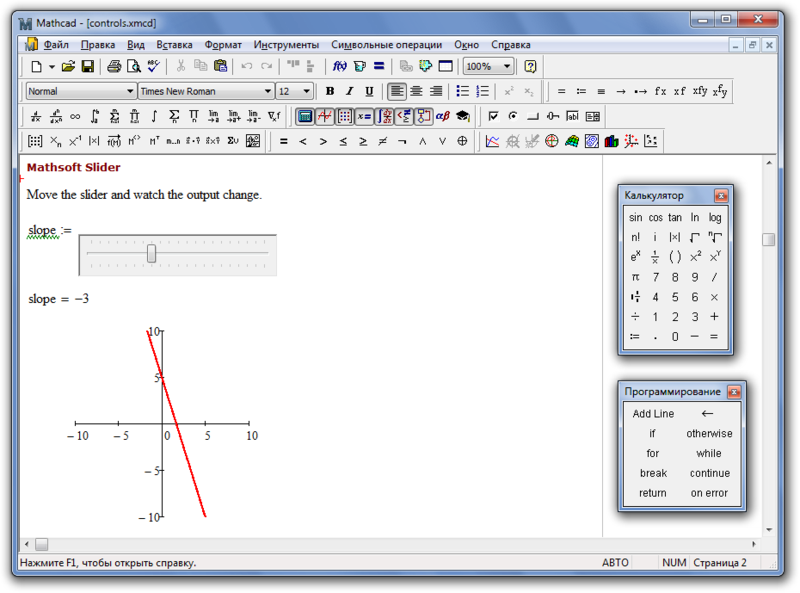


Рисунок 3 – Пользовательский интерфейс Mathcad

Mathcad содержит сотни операторов и встроенных функций для решения различных технических задач. Программа позволяет выполнять численные и символьные вычисления, производить операции со скалярными величинами, векторами и матрицами, автоматически переводить одни единицы измерения в другие.

Среди возможностей Mathcad можно выделить:

1. Решение дифференциальных уравнений, в том числе и численными методами
2. Построение двумерных и трёхмерных графиков функций (в разных системах координат, контурные, векторные и т. д.)
3. Использование греческого алфавита как в уравнениях, так и в тексте
4. Выполнение вычислений в символьном режиме
5. Выполнение операций с векторами и матрицами
6. Символьное решение систем уравнений
7. Аппроксимация кривых
8. Выполнение подпрограмм
9. Поиск корней многочленов и функций
10. Проведение статистических расчётов и работа с распределением вероятностей
11. Поиск собственных чисел и векторов
12. Вычисления с единицами измерения
13. Интеграция с САПР-системами, использование результатов вычислений в качестве управляющих параметров

Оценка функционала – 5

1.4.3 GeoGebra

GeoGebra — это бесплатная, кроссплатформенная динамическая математическая программа для всех уровней образования, включающая в себя геометрию, алгебру, таблицы, графы, статистику и арифметику, в одном удобном для использования пакете.

Программа имеет простой минималистичный и понятный интерфейс. Также, как и в Mathcad, можно использовать для ввода клавиатуру и специальную панель инструментов. Оценка внешнего вида – 4, а удобства – 5.

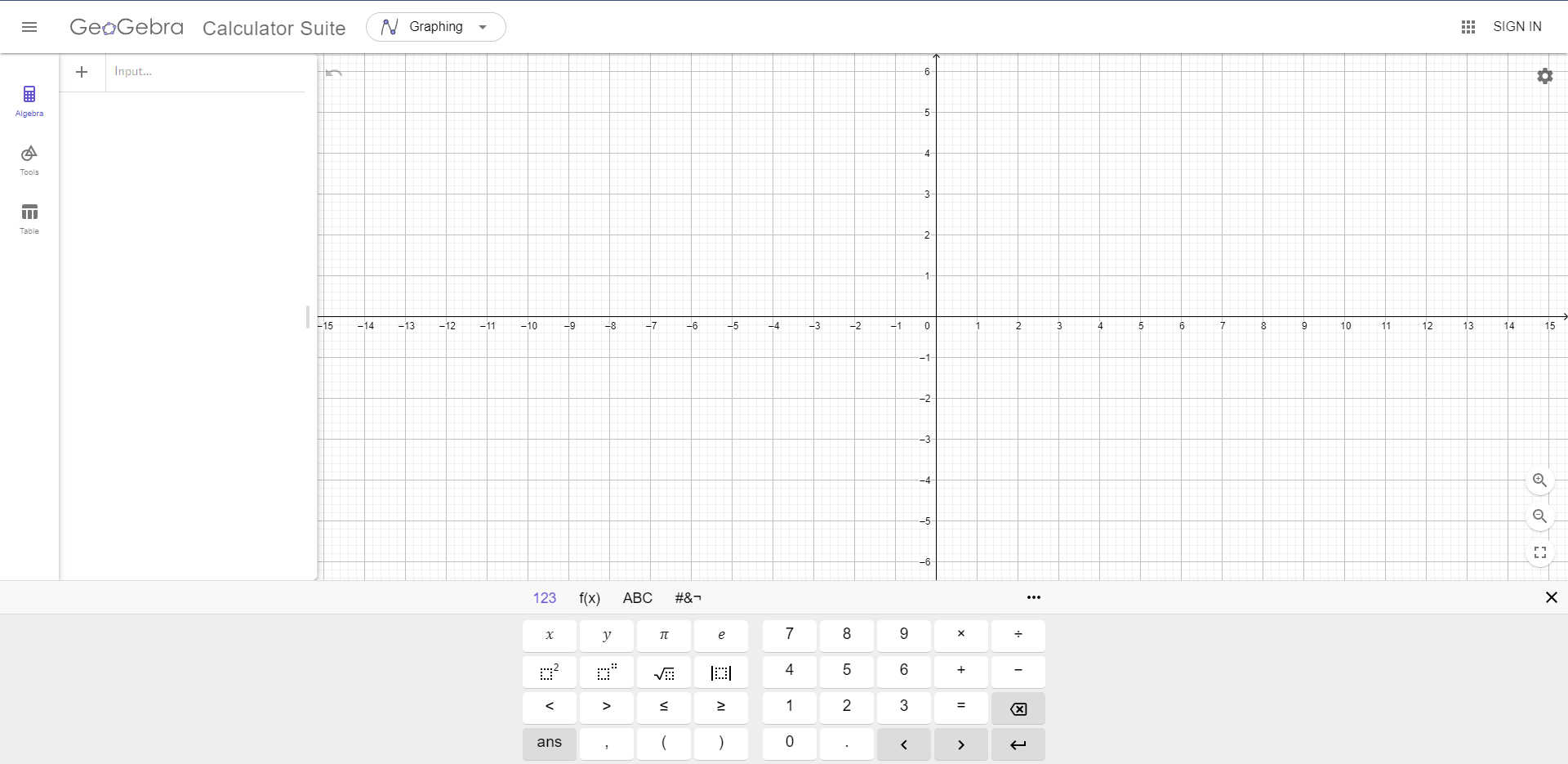


Рисунок 4 – Пользовательский интерфейс GeoGebra

Кроме того, у программы богатые возможности работы с функциями (построение графиков, вычисление корней, экстремумов, интегралов и т. д.) за счёт команд встроенного языка (который также позволяет управлять и геометрическими построениями). Оценка функционала – 5.

1.4.4 Итоговая таблица сравнения аналогов

Таким образом, получается следующая таблица:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Внешний вид | Удобство | Функционал |
| MS Excel | 5 | 3 | 4 |
| Mathcad | 4 | 5 | 5 |
| GeoGebra | 4 | 5 | 5 |

Таблица 1 Анализ аналогов существующих аналогов

1.5 Постановка задачи

Главной задачей данного курсового проекта является разработка программы визуализации решения задач линейного программирования графическим методом. Разработанная программа должна иметь удобный и понятный интерфейс для задания исходных данных. Кроме того, она должна выводить на экран все заданные условия и иметь возможность масштабирования и сдвига полученного изображения по координатным осям.

В данном разделе были проанализированы основные теоретические сведения, касающиеся исследуемой области, существующие аналоги, а также сформулирована и поставлена задача.

2. Разработка структуры приложения и алгоритмов

Структура разработанного в ходе выполнения курсового проекта представлена на рисунке 5.

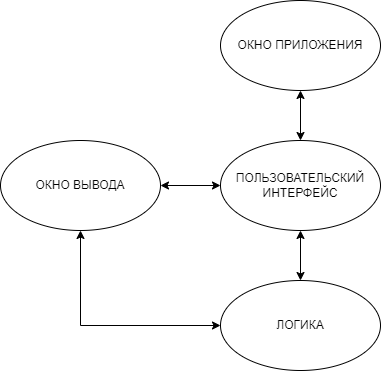


Рисунок 5 – Структура программы

Окно приложения представляет собой совокупность пользовательского интерфейса и окна вывода.

Пользовательский интерфейс – совокупность всех полей для ввода данных и кнопок.

Окно вывода – окно, в котором выводится конечное изображение, включающее в себя координатную плоскость х1Ох2, построенные графики заданных условий, штриховку области допустимых решений, вектор-градиент и линию уровня, которая, в свою очередь, является интерактивной: при нажатии на нее ЛКМ и сдвигом курсора мыши можно двигать данную линию к нужной точке на плоскости.

Логика – блок программы, отвечающий за расчеты при построениях.

2.1 Разработка алгоритма

Алгоритм программы достаточно простой. Сначала пользователь должен выбрать количество используемых ограничений (минимальным значение – 1, максимальное значение – 5). Далее от пользователя требуется ввести коэффициенты линейных уравнений (неравенств) вида: a1 \* х1 + a2 \* х2 = B, выбрать дополнительные условия, накладываемые на переменные, ввести коэффициенты целевой функции и указать координаты искомой точки минимума (максимума). Далее пользователь должен нажать кнопку «Выполнить». После выше описанных действий на экране появится окно вывода, где пользователь может двигать линию уровня в нужную точку, двигать координатную плоскость в положительных и отрицательных направлениях по осям и масштабировать изображения.

В данном разделе были рассмотрены структура программы и краткий алгоритм ее работы.

3. Программная реализация и отладка

В данном разделе описаны основные моменты по разработке интерфейса пользователя, используемые при разработке средства и проверка работоспособности конечного результата.

3.1 Выбор и обоснование языка программирования

В качестве языка программирования для разработки был выбран Free Pascal и среда разработки Lazarus. Lazarus — это специальная программа с интегрированной (свободной) средой для разработки на основе компилятора Free Pascal. Lazarus — программа с полностью открытым исходным кодом, которая направлена на обеспечение библиотек класса Free Pascal. FP — компилятор интегрированной среды разработки соответствует образцу программе Borland — Delphi — и библиотеки Лазарус, которые эквивалентны VCL.

Интерфейс программы очень прост, как для новичков, так и для тех, кто уже занимается программированием на профессиональном уровне. Этого удалось достичь принципом перетаскивания выбранного компонента. Программа позволяет создавать приложения с помощью перетаскивания всех необходимых элементов и помещать их в специальные окна. Большим преимуществом этой программы является возможность компиляции для любых операционных систем.

Основные причины выбора данного языка и среды разработки:

1. простота в использовании
2. удобство в создании простых графических приложений
3. есть весе необходимые для разработки программы компоненты

3.2 Разработка интерфейса пользователя

Система пользовательского интерфейса представляет собой совокупность полей для ввода значений коэффициентов и кнопок: «Выполнить», «+», «-», «Вверх», «Вниз», «Влево», «Вправо» - разбитых по подписанным категориям для удобства работы с программой.

В левом верхнем углу в выпадающем меню выбирается количество ограничений. Чуть ниже приведены 5 строк с полями для ввода чисел, которые могут быть отрицательными и положительными, целыми или дробными. Если выбрать максимальное количество ограничений, то есть 5, то активными для ввода будут все 5 строк, иначе, если выбрать число меньше 5, будет доступно только то количество строк, которое было выбрано пользователем.

В каждой строке есть выпадающие меню, в которых выбирается знак ограничения: «=», «≥», «≤». Также справа от каждой строки подписан цвет, которым будет построена прямая, чтобы было проще различать их между собой.

Еще ниже в выпадающем списке выбираются дополнительные условия, накладываемые на переменные х1 и х2. Проще говоря, можно выбрать либо всю координатную плоскость для поиска области допустимых значений, либо только определенную из четвертей.

Под этим списком находятся поля для ввода коэффициентов перед переменными целевой функции. Они также, как и коэффициенты перед переменными ограничений, могут быть отрицательными и положительными, целыми или дробными.

Ниже расположены поля для ввода координат точки, являющейся решением, а справа от них кнопка «Выполнить», при нажатии на которую будет построены все графики и заштрихованы полуплоскости, а также построены вектор-градиент и линия уровня.

Ниже всех расположены 2 категории кнопок: «Масштаб» и «Перемещение».

Справа от всех выше перечисленных элементов находится окно вывода изображения, на котором рисуются все указанные ограничения, штрихуются области, строятся направляющий вектор и линия уровня. Также для удобства выводится числа, соответствующие четным делениям на осях, и поставлены точки в пересечениях целочисленных координат.

Кроме того, была реализована возможность изменять масштаб и сдвиг по осям в разные стороны на клавиши «+», «-» и стрелки направления соответственно.

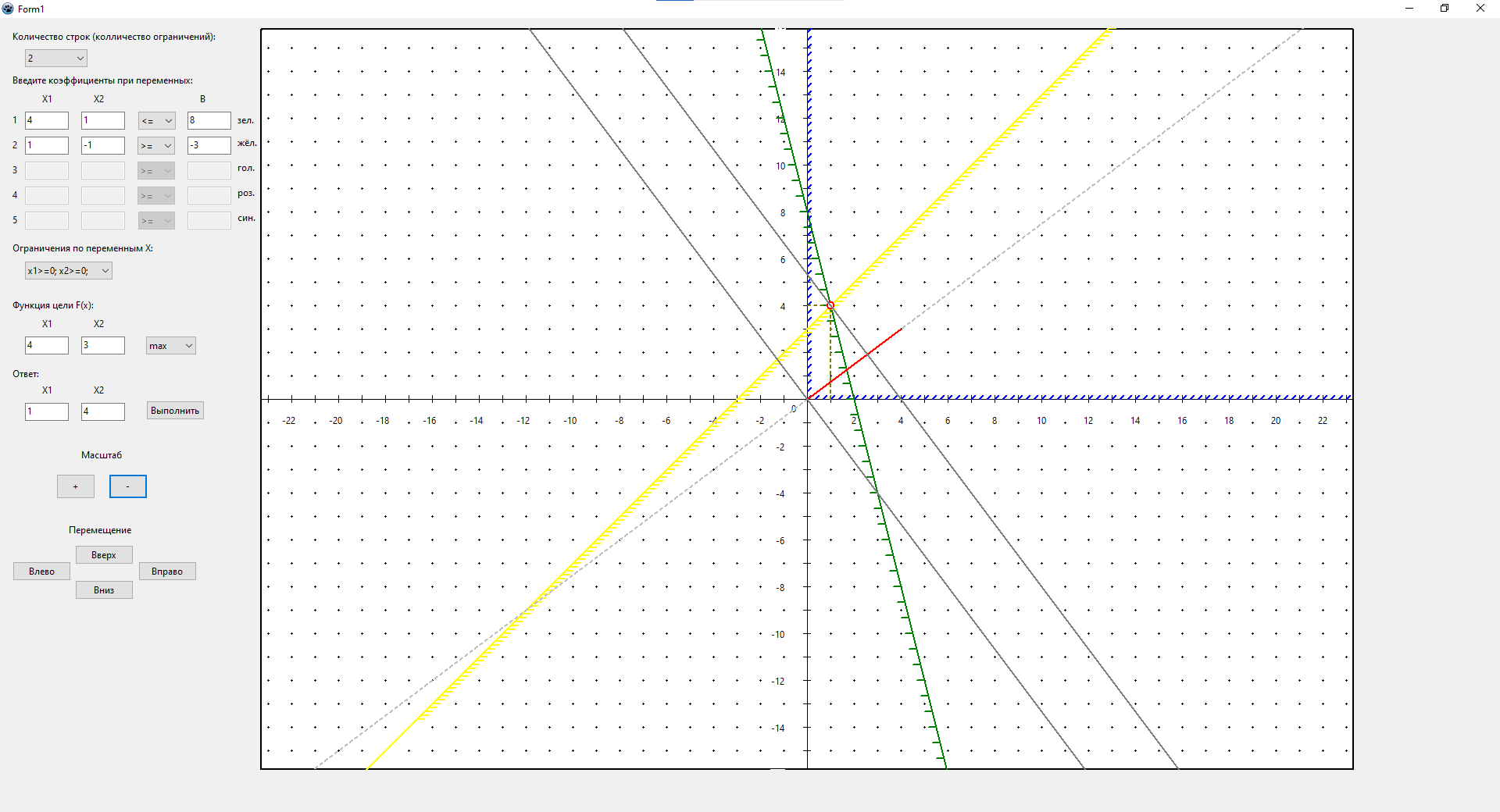


Рисунок 6 – Пользовательский интерфейс

3.3 Реализация программы

Программная реализация достаточно тривиальна. Все функции реализованы с помощью стандартных компонентов среды разработки Lazarus.

Ввод коэффициентов при переменных ограничений и целевой функции, а также ввод координат точки решения (если решением являются все точки, принадлежащие какой-либо прямой, то нужно указать любую из них) реализован с помощью компонента TEdit, который позволяет вводить как числа, так и символы.

Выпадающие списки для выбора количества ограничений, знаков каждого ограничения и выбора направления (максимизации или минимизации), реализованы с помощью компонента TComboBox.

Кнопки, отвечающие за отрисовку изображения, изменения масштаба и сдвига, являются компонентом TButton.

Подписи групп компонентов реализованы с помощью TLabel.

Само изображение координатной плоскости выводится с помощью компонента TPaintBox.

3.4 Отладка программы

Для проверки работоспособности разработанной программы проведем тестирование на нескольких примерах.

3.4.1 Пример №1

Решить задачу графическим методом на минимум:

F = x - 2y → min

5x + 3y ≥ 30

x – y ≤ 3

-3x + 5y ≤ 15

x ≥ 0, y ≥ 0.

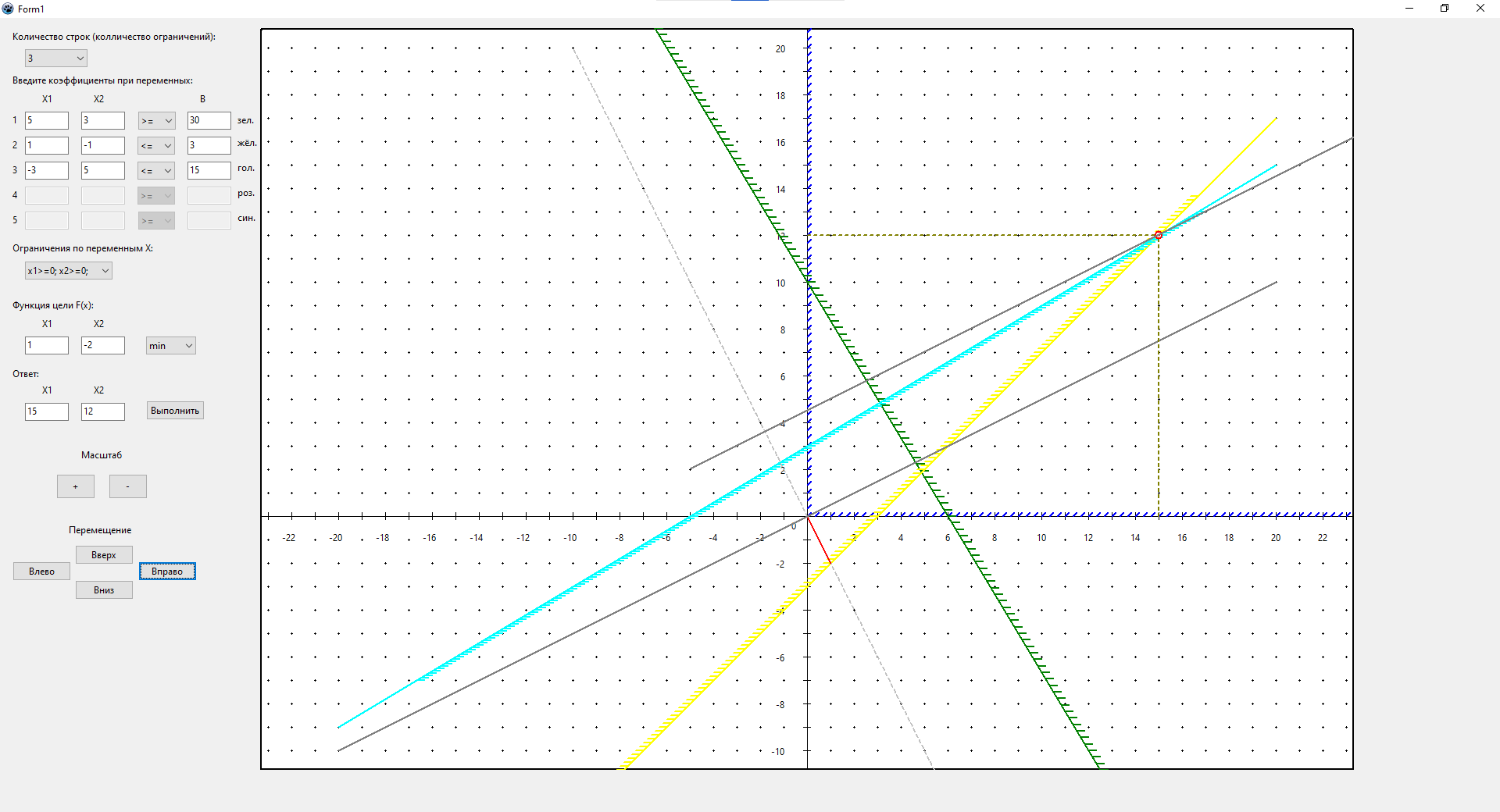


Рисунок 7 – Проверка первого примера

3.4.2 Пример №2



Решить задачу графическим методом на максимум:

F = 2x – y → max

x ≤ 3

x ≥ -1

-2x – 3y ≤ 6

-x + 2y ≤ 6

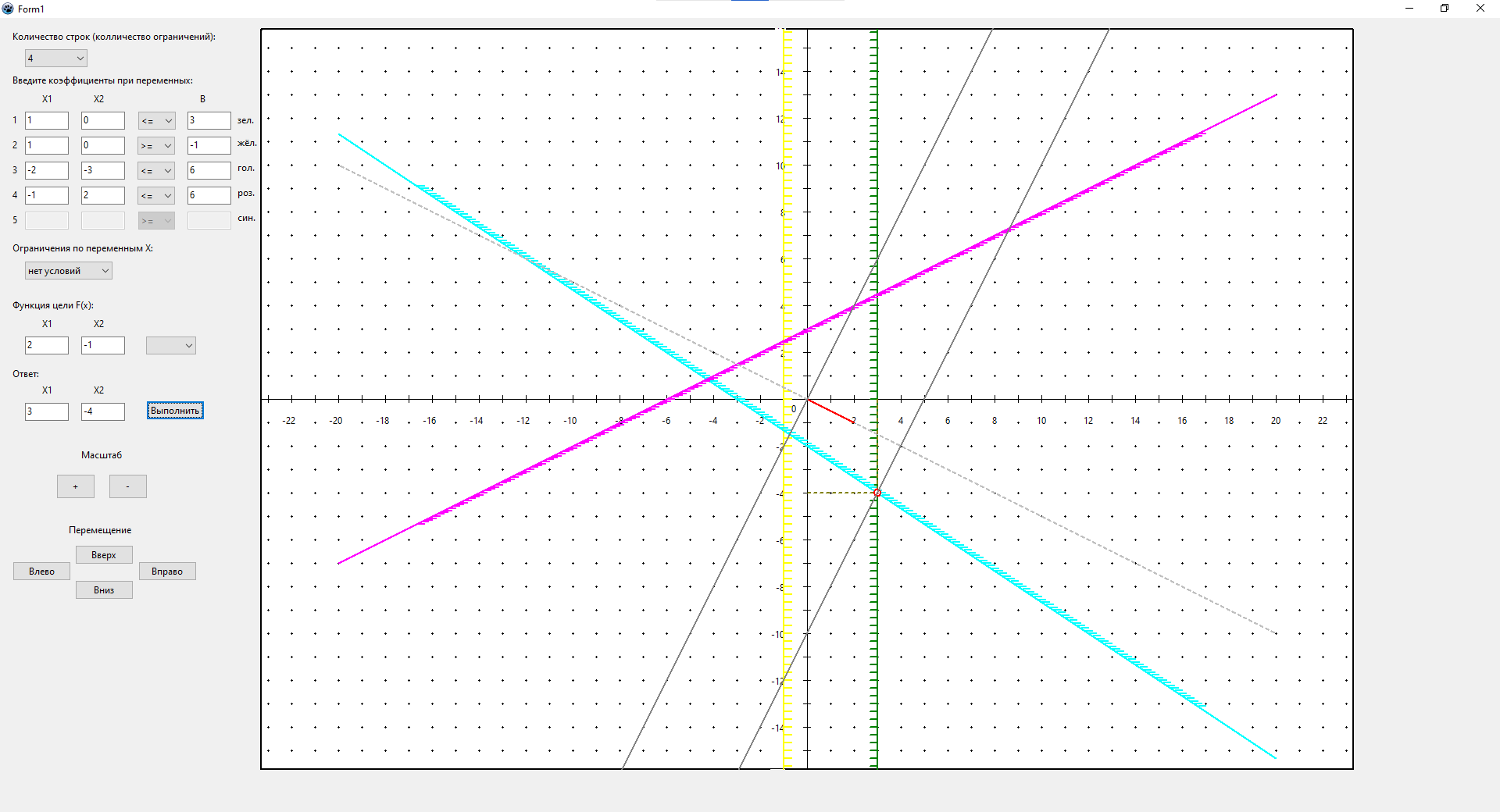


Рисунок 8 – Проверка второго примера

3.4.3 Пример №3

Решить задачу графическим методом на максимум:

F = x + 2y → max

x + 2y ≤ 6

2x + y ≤ 8

y ≥ 2

x ≥ 0, y ≥ 0.

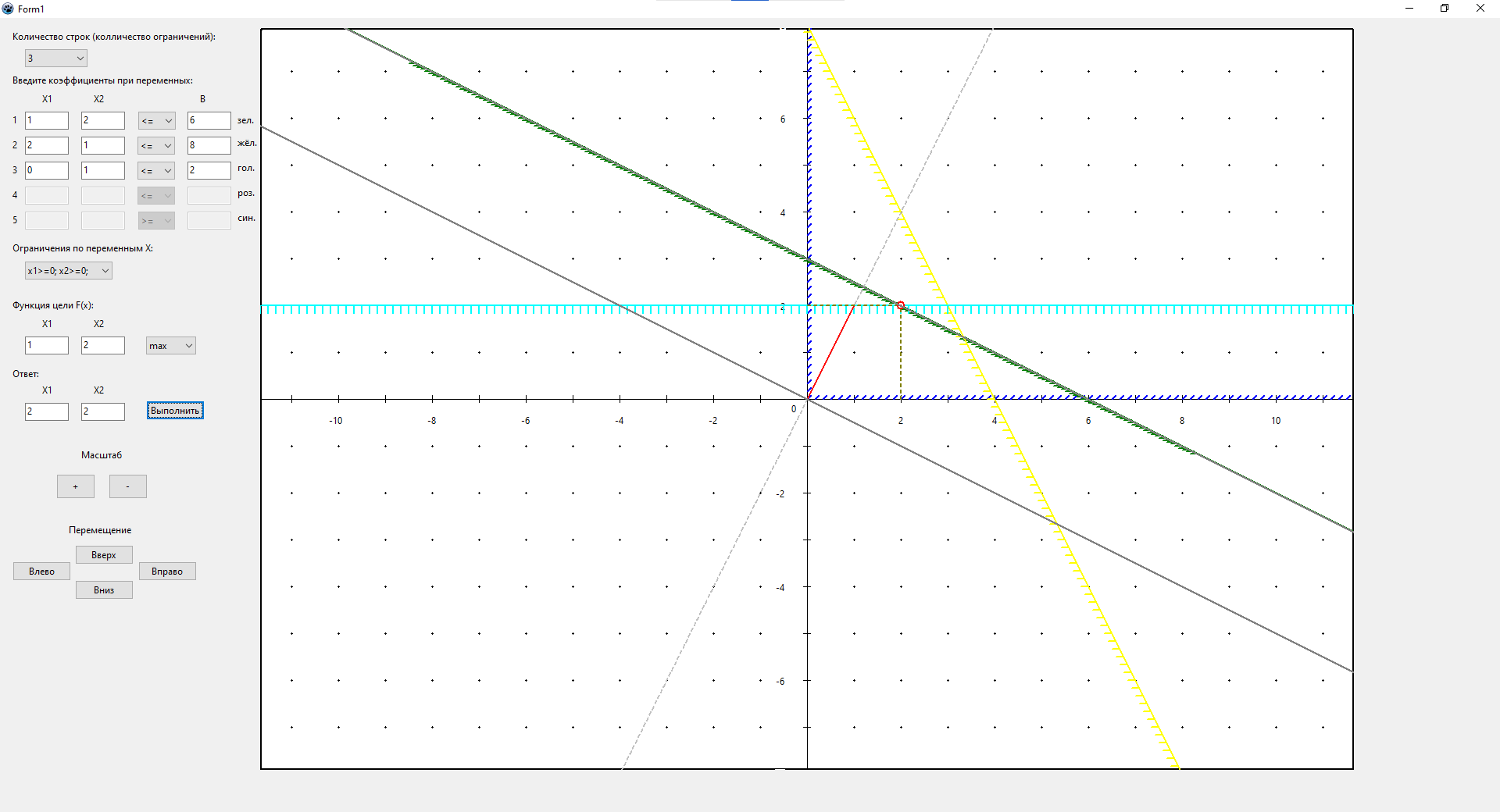


Рисунок 9 – Проверка 3 примера

3.4.4 Итог

Как видно, программа полностью работоспособна и отлажена, обрабатывает все необходимые действия пользователя

Заключение

В данном курсовом проекте были рассмотрены графический метод решения задач линейного программирования и основной теоретический материал по этой теме, а также изучены основные преимущества и недостатки этого метода. К плюсам можно отнести наглядность и простоту в использовании, а к минусам – большие временные затраты в некоторых случаях и достаточно сложные построения при решении задач с тремя переменными. Из последнего следует невозможность решения задач большей размерности.

Анализ существующих аналогов помог понять, что должна представлять из себя программа визуализации решения задач линейного программирования графическим способом.

В результате выполнения курсового проекта была разработана программа, позволяющая описать любую задачу линейного программирования с двумя переменными и не более чем 5 ограничениями.

Приложение А

(обязательное)

Схема алгоритма

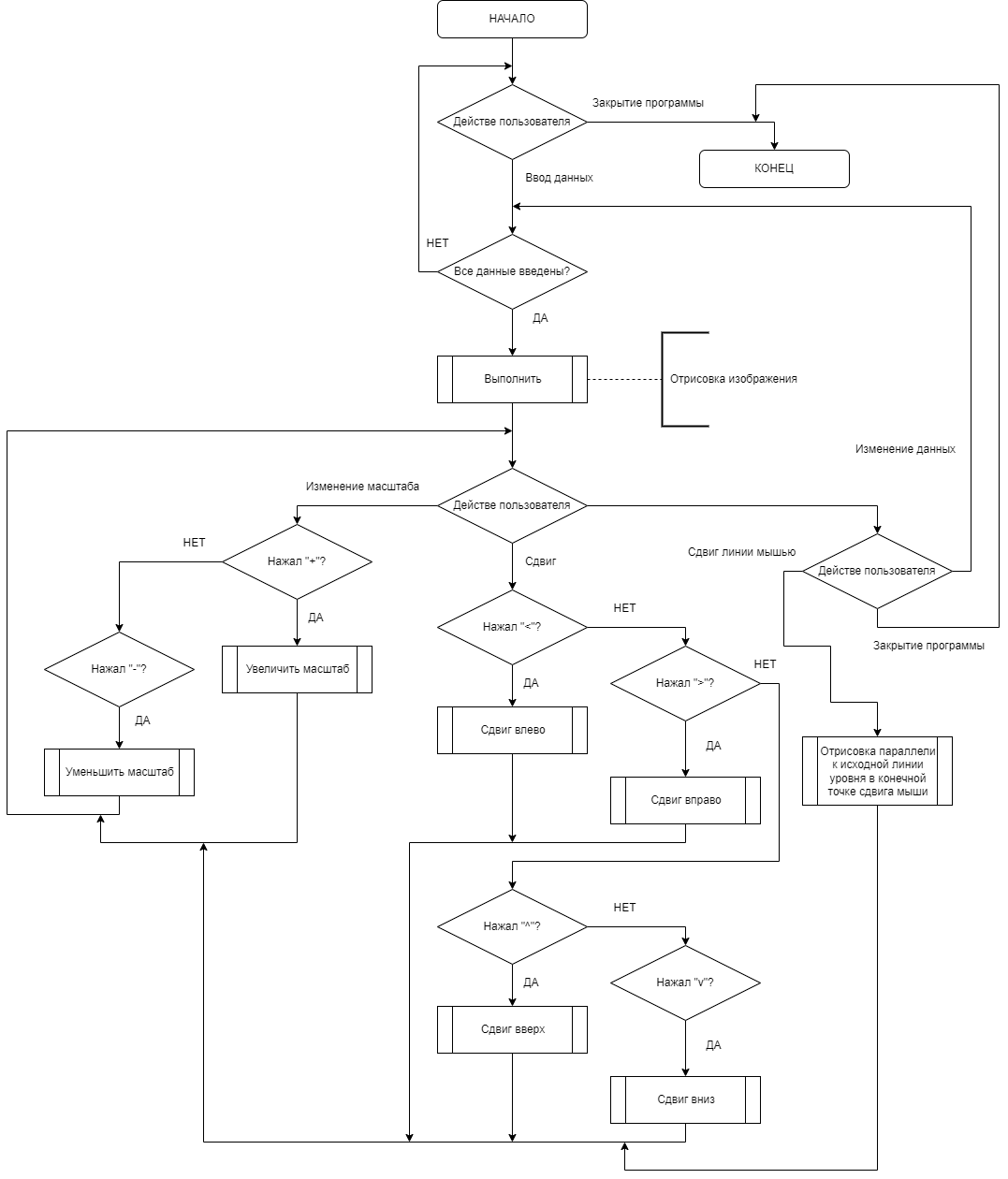


Рисунок А.1 – Схема алгоритма программы

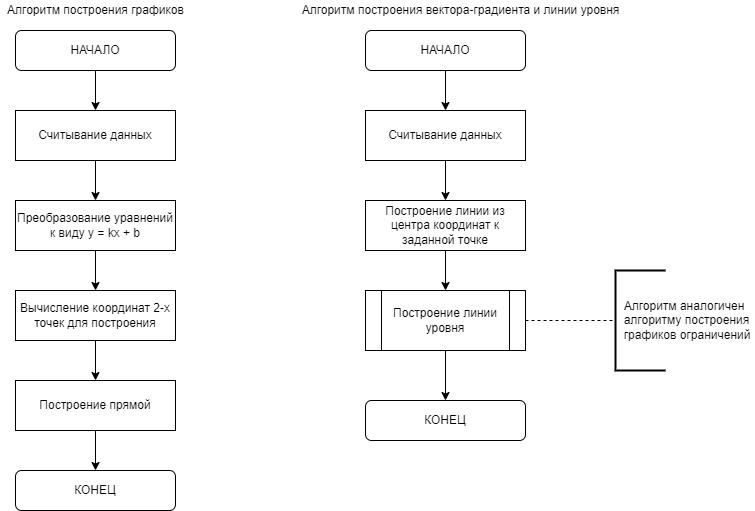


Рисунок А.2 – Схема алгоритмов отрисовки графиков и вектора – градиента

Приложение Б

(Обязательное)

Структура программы

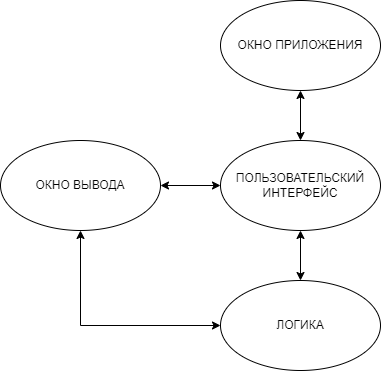


Рисунок Б.1 – Структура программы.

Приложение В

(обязательное)

Исходный код

**procedure** TForm1.shtrixOSI; // процедура штриховки области по осям

**var**

i: integer;

**begin**

Form1.PaintBox1.Canvas.Pen.Color:= ClBlue;

Form1.PaintBox1.Canvas.Pen.Width:=2;

// X>=0 Y>=0

**if** Form1.ComboBox16.ItemIndex = 1 **then**

**begin**

**for** i := 0 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + i\*hh, cy, cx + i\*hh + 5, cy - 5);

**for** i:= 0 **to** 50 **do** Form1.PaintBox1.Canvas.Line(cx, cy - i\*hh, cx+5, cy - i\*hh-5);

**end**;

// X<=0 Y<=0

**if** Form1.ComboBox16.ItemIndex = 2 **then**

**begin**

**for** i := 0 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx - i\*hh, cy, cx - i\*hh - 5, cy + 5);

**for** i:= 0 **to** 50 **do** Form1.PaintBox1.Canvas.Line(cx, cy + i\*hh, cx-5, cy + i\*hh+5);

**end**;

// X>=0 Y<=0

**if** Form1.ComboBox16.ItemIndex = 3 **then**

**begin**

**for** i := 0 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + i\*hh, cy, cx + i\*hh + 5, cy + 5);

**for** i:= 0 **to** 50 **do** Form1.PaintBox1.Canvas.Line(cx, cy + i\*hh, cx+5, cy + i\*hh+5);

**end**;

// X<=0 Y>=0

**if** Form1.ComboBox16.ItemIndex = 4 **then**

**begin**

**for** i := 0 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx - i\*hh, cy, cx - i\*hh - 5, cy - 5);

**for** i:= 0 **to** 50 **do** Form1.PaintBox1.Canvas.Line(cx, cy - i\*hh, cx-5, cy - i\*hh-5);

**end**;

**end**;

// процедура построения графиков и штриховки области

**procedure** TForm1.str\_1;

**var**

x1, x2, x11, x22: real;

ii: integer;

a1,a2,b1,b2: real;

k,x,y,b: real;

**begin**

Form1.PaintBox1.Canvas.Pen.Width:=2;

Form1.PaintBox1.Canvas.Pen.Style:=psSolid; //сплошная линия

Form1.PaintBox1.Canvas.Pen.Color:=clGreen;

**if** (Edit\_X1\_1.Text<>'0') **and** (Edit\_X2\_1.Text<>'0') **and** (Edit\_B1.Text<>'0') **then**

** begin**

mas1[1, 1]:= 20;

mas1[2, 1]:= (strtofloat(Form1.Edit\_B1.Text)-strtofloat(Form1.Edit\_X1\_1.Text)\*20) / strtofloat(Form1.Edit\_X2\_1.Text);

mas1[1, 2]:= -20;

mas1[2, 2]:= (strtofloat(Form1.Edit\_B1.Text)+strtofloat(Form1.Edit\_X1\_1.Text)\*20) / strtofloat(Form1.Edit\_X2\_1.Text);

x1:= mas1[1, 1];

x2:= mas1[2,1];

x11:= mas1[1, 2];

x22:=mas1[2,2];

Form1.PaintBox1.Canvas.Line(cx + round(x1\*h\*m), cy - round(x2\*h\*m), cx + round(x11\*h\*m), cy - round(x22\*h\*m));

// находим уравнение прямой (y=kx+b) по двум точкам A(a1,a2), B(b1,b2)

//присваиваем ранее найденные значения

a1:=cx + round(x1\*h\*m);

a2:=cy - round(x2\*h\*m);

b1:=cx + round(x11\*h\*m);

b2:=cy - round(x22\*h\*m);

k:=(a2-b2)/(a1-b1);

b:=b2-k\*b1;

//решаем в какую сторону будет штриховка, находим точку пересечения координатных прямых(пересекает положительный или отрицательный х),

//x выражаем из уравнения прямой (y=kx+b)при y=0 (475)

x:=(cy-b)/k;

**if** x>cx **then**

**begin**

**if** Form1.ComboBox\_1.ItemIndex = 0 **then** //>=

**begin**

//подставляем 0 вместо x1,x2

**if** strtofloat(Edit\_X1\_1.Text)\*0 + strtofloat(Edit\_X2\_1.Text)\*0 >= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n-10, round(k\*(cx + ii\*n)+b), cx + ii\*n, round(k\*(cx + ii\*n)+b))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n, round(k\*(cx + ii\*n)+b), cx + ii\*n+10, round(k\*(cx + ii\*n)+b));

**end**;

**if** Form1.ComboBox\_1.ItemIndex = 2 **then** //<=

**begin**

**if** strtofloat(Edit\_X1\_1.Text)\*0 + strtofloat(Edit\_X2\_1.Text)\*0 <= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n-10, round(k\*(cx + ii\*n)+b), cx + ii\*n, round(k\*(cx + ii\*n)+b))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n, round(k\*(cx + ii\*n)+b), cx + ii\*n+10, round(k\*(cx + ii\*n)+b));

**end**;

**end**;

**if** x<cx **then**

**begin**

**if** Form1.ComboBox\_1.ItemIndex = 0 **then** //>=

**begin**

//подставляем 0 вместо x1,x2

**if** strtofloat(Edit\_X1\_1.Text)\*0 + strtofloat(Edit\_X2\_1.Text)\*0 >= strtofloat(Edit\_B1.Text) **then**

** for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n, round(k\*(cx + ii\*n)+b), cx + ii\*n+10, round(k\*(cx + ii\*n)+b))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n-10, round(k\*(cx + ii\*n)+b), cx + ii\*n, round(k\*(cx + ii\*n)+b));

**end**;

**if** Form1.ComboBox\_1.ItemIndex = 2 **then** //<=

**begin**

**if** strtofloat(Edit\_X1\_1.Text)\*0 + strtofloat(Edit\_X2\_1.Text)\*0 <= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n, round(k\*(cx + ii\*n)+b), cx + ii\*n+10, round(k\*(cx + ii\*n)+b))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n-10, round(k\*(cx + ii\*n)+b), cx + ii\*n, round(k\*(cx + ii\*n)+b));

**end**;

**end**;

**end**;

**if** (Edit\_X1\_1.Text<>'0') **and** (Edit\_X2\_1.Text='0') **then** //прямая вдоль Оy

**begin**

x1:= strtofloat(Form1.Edit\_B1.Text) / strtofloat(Form1.Edit\_X1\_1.Text);

Form1.PaintBox1.Canvas.Line(cx + round(x1\*h\*m), 1, cx + round(x1\*h\*m), maxy-1);

x:=cx + round(x1\*h\*m);

**if** x > cx **then** // x>0

**begin**

**if** Form1.ComboBox\_1.ItemIndex = 0 **then** //>=

**begin**

//подставляем 0 вместо x1

**if** strtofloat(Edit\_X1\_1.Text)\*0 >= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x-10), cy + ii\*hh, round(x), cy + ii\*hh)

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x), cy + ii\*hh, round(x+10), cy + ii\*hh);

**end**;

**if** Form1.ComboBox\_1.ItemIndex = 2 **then**//<=

**begin**

**if** strtofloat(Edit\_X1\_1.Text)\*0 <= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x-10), cy + ii\*hh, round(x), cy + ii\*hh)

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x), cy + ii\*hh, round(x+10), cy + ii\*hh);

**end**;

**end**;

**if** x<cx **then**

**begin**

**if** Form1.ComboBox\_1.ItemIndex = 0 **then** //>=

**begin**

//подставляем 0 вместо x1

**if** strtofloat(Edit\_X1\_1.Text)\*0 >= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x), cy + ii\*hh, round(x+10), cy + ii\*hh)

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x-10), cy + ii\*hh, round(x), cy + ii\*hh);

**end**;

 rtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x), cy + ii\*hh, round(x+10), cy + ii\*hh)

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x-10), cy + ii\*hh, round(x), cy + ii\*hh);

**end**;

**end**;

**if** x=cx **then**

**begin**

**if** Form1.ComboBox\_1.ItemIndex = 0 **then** //>=

**begin**

//подставляем 0 вместо x1

**if** strtofloat(Edit\_X1\_1.Text)\*1 >= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x), cy + ii\*hh, round(x+10), cy + ii\*hh)

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x-10), cy + ii\*hh, round(x), cy + ii\*hh);

**end**;

**if** Form1.ComboBox\_1.ItemIndex = 2 **then**

**begin**

**if** strtofloat(Edit\_X1\_1.Text)\*1 <= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x), cy + ii\*hh, round(x+10), cy + ii\*hh)

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(round(x-10), cy + ii\*hh, round(x), cy + ii\*hh);

**end**;

**end**;

**end**;

**if** (Edit\_X1\_1.Text='0') **and** (Edit\_X2\_1.Text<>'0') **then** //прямоя вдоль Оx

**begin**

x2:= strtofloat(Form1.Edit\_B1.Text) / strtofloat(Form1.Edit\_X2\_1.Text);

Form1.PaintBox1.Canvas.Line(1, cy - round(x2\*h), maxx-1, cy - round(x2\*h));

y:= cy - round(x2\*h\*m);

**if** y > cy **then** // y<0 в обычной системе координат

**begin**

**if** Form1.ComboBox\_1.ItemIndex = 0 **then** //>=

**begin**

//подставляем 0 вместо x2

**if** strtofloat(Edit\_X2\_1.Text)\*0 >= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y-10), cx + ii\*hh, round(y))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y), cx + ii\*hh, round(y+10));

**end**;

**if** Form1.ComboBox\_1.ItemIndex = 2 **then** //<=

**begin**

**if** strtofloat(Edit\_X2\_1.Text)\*0 <= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y-10), cx + ii\*hh, round(y))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y), cx + ii\*hh, round(y+10));

**end**;

**end**;

 **if** y < cy **then**

**begin**

**if** Form1.ComboBox\_1.ItemIndex = 0 **then** //>=

**begin**

//подставляем 0 вместо x2

**if** strtofloat(Edit\_X2\_1.Text)\*0 >= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y+10), cx + ii\*hh, round(y))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y), cx + ii\*hh, round(y-10));

**end**;

**if** Form1.ComboBox\_1.ItemIndex = 2 **then** //<=

**begin**

**if** strtofloat(Edit\_X2\_1.Text)\*0 <= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y+10), cx + ii\*hh, round(y))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y), cx + ii\*hh, round(y-10));

**end**;

**end**;

**if** y = cy **then**

**begin**

**if** Form1.ComboBox\_1.ItemIndex = 0 **then** //>=

**begin**

//подставляем 0 вместо x2

**if** strtofloat(Edit\_X2\_1.Text)\*1 >= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y-10), cx + ii\*hh, round(y))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y), cx + ii\*hh, round(y+10));

**end**;

**if** Form1.ComboBox\_1.ItemIndex = 2 **then** //<=

**begin**

**if** strtofloat(Edit\_X2\_1.Text)\*1 <= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y-10), cx + ii\*hh, round(y))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*hh, round(y), cx + ii\*hh, round(y+10));

**end**;

**end**;

**end**;

**if** (Edit\_B1.Text='0') **and** (Edit\_X1\_1.Text<>'0') **and** (Edit\_X2\_1.Text<>'0') **then**

**begin**

mas1[1, 1]:= 20;

mas1[2, 1]:= -strtofloat(Form1.Edit\_X1\_1.Text)\*20 / strtofloat(Form1.Edit\_X2\_1.Text);

mas1[1, 2]:= -20;

mas1[2, 2]:= strtofloat(Form1.Edit\_X1\_1.Text)\*20 / strtofloat(Form1.Edit\_X2\_1.Text);

x1:= mas1[1, 1];

x2:= mas1[2,1];

x11:= mas1[1, 2];

x22:=mas1[2,2];

Form1.PaintBox1.Canvas.Line(cx + round(x1\*h\*m), cy - round(x2\*h\*m), cx + round(x11\*h\*m), cy - round(x22\*h\*m));

 a1:=cx + round(x1\*h\*m);

a2:=cy - round(x2\*h\*m);

b1:=cx + round(x11\*h\*m);

b2:=cy - round(x22\*h\*m);

k:=(a2-b2)/(a1-b1);

b:=b2-k\*b1;

x:=(cy-h\*m-b)/k;

**if** x>cx **then**

**begin**

**if** Form1.ComboBox\_1.ItemIndex = 0 **then** //>=

**begin**

//подставляем 0,1 вместо x1,x2

**if** strtofloat(Edit\_X1\_1.Text)\*0 + strtofloat(Edit\_X2\_1.Text)\*1 >= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n-10, round(k\*(cx + ii\*n)+b), cx + ii\*n, round(k\*(cx + ii\*n)+b))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n, round(k\*(cx + ii\*n)+b), cx + ii\*n+10, round(k\*(cx + ii\*n)+b));

**end**;

**if** Form1.ComboBox\_1.ItemIndex = 2 **then** //<=

**begin**

**if** strtofloat(Edit\_X1\_1.Text)\*0 + strtofloat(Edit\_X2\_1.Text)\*1 <= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n-10, round(k\*(cx + ii\*n)+b), cx + ii\*n, round(k\*(cx + ii\*n)+b))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n, round(k\*(cx + ii\*n)+b), cx + ii\*n+10, round(k\*(cx + ii\*n)+b));

**end**;

**end**;

**if** x<cx **then**

**begin**

**if** Form1.ComboBox\_1.ItemIndex = 0 **then** //>=

**begin**

//подставляем 0,1 вместо x1,x2

**if** strtofloat(Edit\_X1\_1.Text)\*0 + strtofloat(Edit\_X2\_1.Text)\*1 >= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n, round(k\*(cx + ii\*n)+b), cx + ii\*n+10, round(k\*(cx + ii\*n)+b))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n-10, round(k\*(cx + ii\*n)+b), cx + ii\*n, round(k\*(cx + ii\*n)+b));

**end**;

**if** Form1.ComboBox\_1.ItemIndex = 2 **then** //<=

**begin**

**if** strtofloat(Edit\_X1\_1.Text)\*0 + strtofloat(Edit\_X2\_1.Text)\*1 <= strtofloat(Edit\_B1.Text) **then**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n, round(k\*(cx + ii\*n)+b), cx + ii\*n+10, round(k\*(cx + ii\*n)+b))

**else**

**for** ii := -100 **to** 100 **do** Form1.PaintBox1.Canvas.Line(cx + ii\*n-10, round(k\*(cx + ii\*n)+b), cx + ii\*n, round(k\*(cx + ii\*n)+b));

**end**;

**end**;

**end**;

**end**;

// для 2-х и более условий алгоритм аналогичный

**procedure** TForm1.OSI; // PainBox, оси, числа на осях, точки

**var**

i, ii, x, y, n: integer;

t: string;

**begin**

// PaintBox

Form1.PaintBox1.Canvas.Brush.Color:= ClWhite;

Form1.Paintbox1.Canvas.FillRect(0, 0, paintbox1.Width, paintbox1.Height);

Form1.PaintBox1.Canvas.Pen.Width:=2;

Form1.PaintBox1.Canvas.Pen.Style:=psSolid;

Form1.PaintBox1.Canvas.Pen.Color:=ClBlack;

Form1.PaintBox1.Canvas.Rectangle(1, 1, paintbox1.Width, paintbox1.Height);

Form1.PaintBox1.Canvas.Line(1,1,2,2);

//Точки(сетка)

**for** i:= -50 **to** 50 **do**

**begin**

y:=round(cy + i\*h\*m);

**for** ii := -50 **to** 50 **do**

**begin**

x:=round(cx + ii\*h\*m);

Form1.PaintBox1.Canvas.Pixels[x, y] := ClBlack;

Form1.PaintBox1.Canvas.Pixels[x-1, y] := ClBlack;

Form1.PaintBox1.Canvas.Pixels[x+1, y] := ClBlack;

Form1.PaintBox1.Canvas.Pixels[x, y-1] := ClBlack;

Form1.PaintBox1.Canvas.Pixels[x, y+1] := ClBlack;

**end**;

**end**;

// Оси

Form1.PaintBox1.Canvas.Pen.Width:=1;

Form1.PaintBox1.Canvas.Pen.Style:=psSolid;

Form1.PaintBox1.Canvas.Line(cx, 0, cx, maxy);

Form1.PaintBox1.Canvas.Line(0, cy, maxx, cy);

**for** i := -50 **to** 50 **do** Form1.PaintBox1.Canvas.Line(round(cx + i\*h\*m), cy - 5, round(cx + i\*h\*m), cy + 5); // OX

**for** i:= -50 **to** 50 **do** Form1.PaintBox1.Canvas.Line(cx-5, round(cy + i\*h\*m), cx+5, round(cy + i\*h\*m)); // OY

//числа на осях Ox

n:=0;

Form1.PaintBox1.Canvas.TextOut(cx-20, cy+5, inttostr(n));

**for** i := -50 **to** -10 **do**

**begin**

n:=i;

**if** i **mod** 2=0 **then**

Form1.PaintBox1.Canvas.TextOut(round(cx + i\*h\*m-12), cy+20, inttostr(n));

**end**;

**for** i := -9 **to** -1 **do**

**begin**

n:=i;

**if** i **mod** 2=0 **then**

Form1.PaintBox1.Canvas.TextOut(round(cx + i\*h\*m-6), cy+20, inttostr(n));

**end**;

**for** i := 1 **to** 9 **do**

**begin**

n:=i;

**if** i **mod** 2=0 **then**

Form1.PaintBox1.Canvas.TextOut(round(cx + i\*h\*m-3), cy+20, inttostr(n));

**end**;

**for** i := 10 **to** 50 **do**

**begin**

n:=i;

**if** i **mod** 2=0 **then**

Form1.PaintBox1.Canvas.TextOut(round(cx + i\*h\*m-6), cy+20, inttostr(n));

**end**;

 //числа на осях Oy

**for** i := -50 **to** -10 **do**

**begin**

n:=i;

**if** i **mod** 2=0 **then**

Form1.PaintBox1.Canvas.TextOut(cx-46, round(cy - i\*h\*m-6), inttostr(n));

**end**;

**for** i := -9 **to** -1 **do**

**begin**

n:=i;

**if** i **mod** 2=0 **then**

Form1.PaintBox1.Canvas.TextOut(cx-40, round(cy - i\*h\*m-6), inttostr(n));

**end**;

**for** i := 1 **to** 9 **do**

**begin**

n:=i;

**if** i **mod** 2=0 **then**

Form1.PaintBox1.Canvas.TextOut(cx-34, round(cy - i\*h\*m-6), inttostr(n));

**end**;

**for** i := 10 **to** 50 **do**

**begin**

n:=i;

**if** i **mod** 2=0 **then**

Form1.PaintBox1.Canvas.TextOut(cx-40, round(cy - i\*h\*m-6), inttostr(n));

**end**;

**end**;

**procedure** TForm1.shtrixOSI; // процедура штриховки области по осям

**var**

i: integer;

**begin**

Form1.PaintBox1.Canvas.Pen.Color:= ClBlue;

Form1.PaintBox1.Canvas.Pen.Width:=2;

// X>=0 Y>=0

**if** Form1.ComboBox\_dop\_usl.ItemIndex = 1 **then**

**begin**

**for** i := 0 **to** 150 **do** Form1.PaintBox1.Canvas.Line(cx + i\*hh, cy, cx + i\*hh + 5, cy - 5);

**for** i:= 0 **to** 150 **do** Form1.PaintBox1.Canvas.Line(cx, cy - i\*hh, cx+5, cy - i\*hh-5);

**end**;

// X<=0 Y<=0

**if** Form1.ComboBox\_dop\_usl.ItemIndex = 2 **then**

**begin**

**for** i := 0 **to** 150 **do** Form1.PaintBox1.Canvas.Line(cx - i\*hh, cy, cx - i\*hh - 5, cy + 5);

**for** i:= 0 **to** 150 **do** Form1.PaintBox1.Canvas.Line(cx, cy + i\*hh, cx-5, cy + i\*hh+5);

**end**;

// X>=0 Y<=0

**if** Form1.ComboBox\_dop\_usl.ItemIndex = 3 **then**

**begin**

**for** i := 0 **to** 150 **do** Form1.PaintBox1.Canvas.Line(cx + i\*hh, cy, cx + i\*hh + 5, cy + 5);

**for** i:= 0 **to** 150 **do** Form1.PaintBox1.Canvas.Line(cx, cy + i\*hh, cx+5, cy + i\*hh+5);

**end**;

// X<=0 Y>=0

**if** Form1.ComboBox\_dop\_usl.ItemIndex = 4 **then**

**begin**

**for** i := 0 **to** 150 **do** Form1.PaintBox1.Canvas.Line(cx - i\*hh, cy, cx - i\*hh - 5, cy - 5);

**for** i:= 0 **to** 150 **do** Form1.PaintBox1.Canvas.Line(cx, cy - i\*hh, cx-5, cy - i\*hh-5);

**end**;

**end**;

//вектор-градиент и линия уровня

**procedure** TForm1.Grad;

**var**

x1, x2, x11, x22: real;

g1, g2, cx1, cy1: real;

xt, yt: integer;

**begin**

//вектор-градиент

g1:=strtoint(Form1.Edit\_X1.Text);

g2:=strtoint(Form1.Edit\_X2.Text);

//Продолжение вектора

Form1.PaintBox1.Canvas.Pen.Color:=clSilver;

Form1.PaintBox1.Canvas.Pen.Style:=psDash;

Form1.PaintBox1.Canvas.Pen.Width:=2;

Form1.PaintBox1.Canvas.Line(cx,cy,cx+round(g1\*h\*m)\*10,cy-round(g2\*h\*m)\*10);

Form1.PaintBox1.Canvas.Line(cx,cy,cx-round(g1\*h\*m)\*10,cy+round(g2\*h\*m)\*10);

//Вектор

Form1.PaintBox1.Canvas.Pen.Color:= clRed;

Form1.PaintBox1.Canvas.Pen.Style:=psSolid;

Form1.PaintBox1.Canvas.Pen.Width:=2;

Form1.PaintBox1.Canvas.Line(cx,cy,cx+round(g1\*h\*m),cy-round(g2\*h\*m));

//Линия уровня(перпендикуляр к вектору)

**if** (Edit\_X1.Text<>'0') **and** (Edit\_X2.Text<>'0') **then**

**begin**

mas1[1, 1]:= 20;

mas1[2, 1]:= (0-strtofloat(Form1.Edit\_X1.Text)\*20) / strtofloat(Form1.Edit\_X2.Text);

mas1[1, 2]:= -20;

mas1[2, 2]:= (0+strtofloat(Form1.Edit\_X1.Text)\*20) / strtofloat(Form1.Edit\_X2.Text);

Form1.PaintBox1.Canvas.Pen.Width:=2;

Form1.PaintBox1.Canvas.Pen.Style:=psSolid;

Form1.PaintBox1.Canvas.Pen.Color:=clGray;

x1:= mas1[1, 1];

x2:= mas1[2,1];

x11:= mas1[1, 2];

x22:=mas1[2,2];

Form1.PaintBox1.Canvas.Line(cx + round(x1\*h\*m), cy - round(x2\*h\*m), cx + round(x11\*h\*m), cy - round(x22\*h\*m));

//Линии к точке

Form1.PaintBox1.Canvas.Pen.Width:=2;

Form1.PaintBox1.Canvas.Pen.Style:=psDash;

Form1.PaintBox1.Canvas.Pen.Color:=clOlive;

cx1:= strtofloat(Form1.Edit\_X1\_otvet.Text);

cy1:= strtofloat(Form1.Edit\_X2\_otvet.Text);

Form1.PaintBox1.Canvas.Line(cx + round(cx1\*h\*m), cy, cx+round(cx1\*h\*m), cy - round(cy1\*h\*m));

Form1.PaintBox1.Canvas.Line(cx, cy - round(cy1\*h\*m),cx+round(cx1\*h\*m), cy - round(cy1\*h\*m));

//выделение точки

xt:=cx + round(cx1\*h\*m);

yt:=cy - round(cy1\*h\*m);

Form1.PaintBox1.Canvas.Pen.Width:=2;

Form1.PaintBox1.Canvas.Pen.Style:=psSolid;

Form1.PaintBox1.Canvas.Pen.Color:=clRed;

 Form1.PaintBox1.Canvas.Ellipse(xt-5, yt-5, xt+5,yt+5);

Form1.PaintBox1.Canvas.Pixels[xt, yt] := ClRed;

Form1.PaintBox1.Canvas.Pixels[xt-1, yt] := ClRed;

Form1.PaintBox1.Canvas.Pixels[xt+1, yt] := ClRed;

Form1.PaintBox1.Canvas.Pixels[xt, yt-1] := ClRed;

Form1.PaintBox1.Canvas.Pixels[xt, yt+1] := ClRed;

**end**;

**end**;

Приложение Г

(справочное)

Список используемой литературы

1. https://ru.wikipedia.org/wiki/Линейное программирование
2. Банди Б. Б23 Основы линейного программирования: Пер. с англ. - М.: Радио и связь, 1989. – 176 с.: ил.
3. Гасс С.Линейное программирование. - М.: Физматгиз, 2001.
4. Ланских В. Г. Математическое программирование. В 2-частях. Ч.1. Линейное и нелинейное программирование : учебное пособие / В. Г. Ланских, Ю. В. Ланских. – Киров : ВятГУ, 2019. – 196 с.
5. https://ru.wikipedia.org/wiki/Графический метод решения задач линейного программирования