**Реферат**

Курсовая работа: 30 страниц, 17 рисунков, 2 таблицы, 6 используемых источников, 2 приложения.

СИМПЛЕКС-МЕТОД, ОПТИМИЗАЦИЯ, ВЫХОД ГОТОВЫХ ИЗДЕЛИЙ, СТРУКТУРЫ ДАННЫХ, ТЕСТИРОВАНИЕ по, ДИЗАЙН ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ, СТРУКТУРНАЯ СХЕМА, ДИАГРАММА СОСТОЯНИЙ ИНТЕРФЕЙСА, ВВОД-ВЫВОД ИНФОРМАЦИИ, ДВИЖЕНИЕ ИНФОРМАЦИОННЫХ ПОТОКОВ

**Содержание**

[Введение 5](#_Toc104287508)

[1 Нормативные ссылки 6](#_Toc104287509)

[2 Анализ технических требований и уточнение спецификаций 7](#_Toc104287510)

[2.1 Анализ задания и выбор технологии, языка и среды разработки 7](#_Toc104287511)

[2.2 Анализ процесса обработки информации и построение функциональных диаграмм 8](#_Toc104287512)

[2.3 Анализ хранимой информации и выбор структур данных для ее представления 10](#_Toc104287513)

[2.4 Выбор методов и алгоритмов решения задачи 11](#_Toc104287514)

[3 Проектирование структуры и компонентов программного продукта 13](#_Toc104287515)

[3.1 Разработка интерфейса пользователя 13](#_Toc104287516)

[3.1.1 Разработка структурной схемы интерфейса 13](#_Toc104287517)

[3.1.2 Построение диаграммы состояний интерфейса 14](#_Toc104287518)

[3.1.3 Разработка форм ввода-вывода информации 15](#_Toc104287519)

[3.2 Разработка алгоритма основной программы и структурной схемы программного продукта 16](#_Toc104287520)

[3.2.1 Описание структуры приложения и схема связности модулей 16](#_Toc104287521)

[3.2.2 Схема движения информационных потоков 17](#_Toc104287522)

[3.3 Разработка основных алгоритмов программного продукта 18](#_Toc104287523)

[4 Тестирование программы 24](#_Toc104287524)

[4.1 Разработка плана тестирования 24](#_Toc104287525)

[4.2 Разработка алгоритма процедуры тестирования 24](#_Toc104287526)

[4.3 Оценка результатов тестирования 25](#_Toc104287527)

[5 Сопровождение 28](#_Toc104287528)

[6 Заключение 30](#_Toc104287529)

[Список использованных источников 31](#_Toc104287530)

[Приложение А 32](#_Toc104287531)

[Приложение Б 33](#_Toc104287533)

# Введение

Целью данный курсовой работы является решение задач на оптимизацию симплекс-методом.

В первой главе описаны основные нормативные ссылки, использованные в пояснительной записке.

Во второй главе происходит анализ технических требований и уточнение спецификации задачи.

В третьей главе разрабатывается дизайн, а также основные алгоритмы работы программного обеспечения.

В четвёртой главе происходит тестирование программного обеспечение и сравнение реальных полученных результатов с ожидаемыми.

В пятой главе приведено руководство пользователя по эксплуатации программного обеспечения.

# 1 Нормативные ссылки

В данной курсовой работе были использованы ссылки на следующие нормативные документы:

* ГОСТ Р7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления;
* ГОСТ Р1.5-2004. Стандарты национальные РФ. Правила построения, изложения, оформления и обозначения;
* ГОСТ 2.301-68 ЕСКД. Форматы;
* ГОСТ 7.82-2001 СИБИД. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления;
* ГОСТ 7.12-93 СИБИД. Библиографическая запись. Сокращения слов на русском языке. Общие требования и правила;
* ГОСТ 7.9-95 СИБИД. Реферат и аннотация. Общие требования.

# 2 Анализ технических требований и уточнение спецификаций

## 2.1 Анализ задания и выбор технологии, языка и среды разработки

В задании к данной курсовой работе говорится о комбинатах, которые выпускают железобетонные панели. Комбинаты производят три типа железобетонных панелей (по одному типу на каждом комбинате) для строительства жилых домов. Для решения задачи необходимо определить оптимальные доли использования различных видов топлива (нефть, уголь, газ) для каждого комбината, чтобы максимизировать производство железобетонных панелей и общую прибыль. Также нужно учитывать доступные ресурсы (топливо, цемент, арматура) и требования к производству каждого типа панелей.

Данная задача относится к оптимизационным задачам, в данном случае – это задача на максимизацию выхода готовых изделий. Такого рода задачи могут быть решены с помощью симплекс-метода. В качестве входных данных будет выступать таблица, в которую будет занесена информация. Выходные данные будут хранить максимальное количество комплектов и максимальную прибыль.

В качестве языка программирования был выбран язык C#, по нескольким причинам:

* С# — это объектно-ориентированный язык, который позволяет использовать ранее написанный код, как встроенный (библиотеки), так и собственный, позволяющий абстрагироваться от написания большого количества вспомогательных функций и сосредоточиться на выполнении основной последовательности действий;
* структуры данных, которые поддерживает язык, например, такие как одномерные и многомерные массивы, массивы, позволяющие хранить в себе элементы управления и списки, дают возможность написать эффективный и лаконичный программный код;
* взаимодействие с элементами управления через программный код: C# имеет множество как методов, так и структур данных, которые позволяют динамически манипулировать интерфейсом программы.

Средой разработки была выбрана Microsoft Visual Studio. Среда позволяет создавать приложения с интерфейсом, который будет интуитивно понятен пользователю. В том числе, Visual Studio поддерживает событийное программирование и позволяет писать программный код на языке программирования C#, который был выбран в качестве языка разработки.

## 2.2 Анализ процесса обработки информации и построение функциональных диаграмм

Для решения задачи был выбран симплекс-метод, вследствие чего обработка информации будет непосредственно связана с работой самого метода [1].

Работа с симплекс-методом начинается с ввода ограничений и целевой функции. Целевая функция – функция, максимизация или минимизация которой, является целью, или степенью, решения задачи. Ограничения определяют условия, в рамках которых задача должна быть решена [6].

Симплекс-метод сводится к построению опорных планов, пока значения переменных не будут максимально оптимизированы согласно целевой функции.

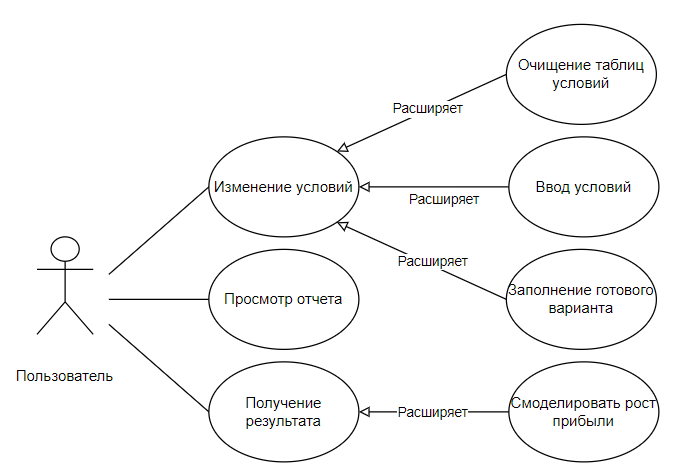


Рисунок 1 – Диаграмма вариантов использования приложения

Первый опорный план подразумевает под собой введение дополнительных переменных. Например, в экономическом плане, такие переменные подразумевают под собой излишки сырья, времени и других ресурсов, остающихся на производстве при использовании данного опорного плана. Свободные переменные при этом приравниваются к нулю.

Далее, симплекс-метод содержит 4 этапа, которые выполняются последовательно до тех пор, пока не будет найден оптимальный план:

1 этап – проверка критерия оптимальности: строка, в данный момент, содержащая коэффициенты целевой функции проверяется на наличие положительных элементов. При наличии таковых план является неоптимальным и вычисления продолжаются, в противном случае, решение найдено: значения переменных, находящихся в базисе, записываются аналогично таблице, остальные переменные обращаются в нуль.

2 этап – определение новой базисной переменной: в качестве ведущего выбирается столбец, который содержит наибольший коэффициент целевой функции.

3 этап – определение новой свободной переменной: для определения строки, содержащий свободный элемент, необходимо выбрать минимальное частное от деления соответствующих элементов базисного столбца и элементов ведущего столбца, выбранного на этапе 2.

4 этап – пересчет симплекс-таблицы. Каждый элемент новой таблицы, высчитывается по формуле: элемент старой таблицы – (элемент разрешающей строки \* элемент разрешающего столбца) / разрешающий элемент.

Далее цикл переходит вновь к этапу 1 [7].

Данный алгоритм можно увидеть на функциональной диаграмме, представленной на рисунках

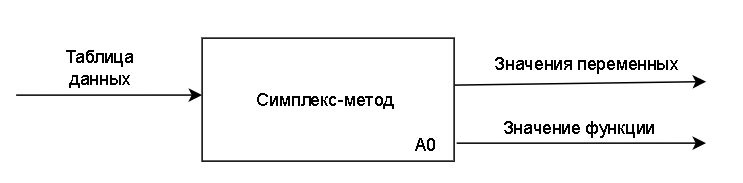


Рисунок 2 – Функциональная диаграмма (уровень А0)

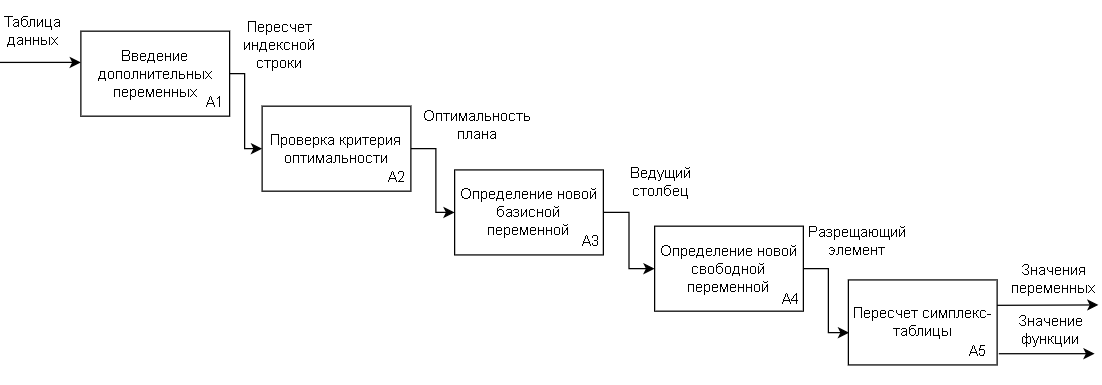


Рисунок 3 – Функциональная диаграмма

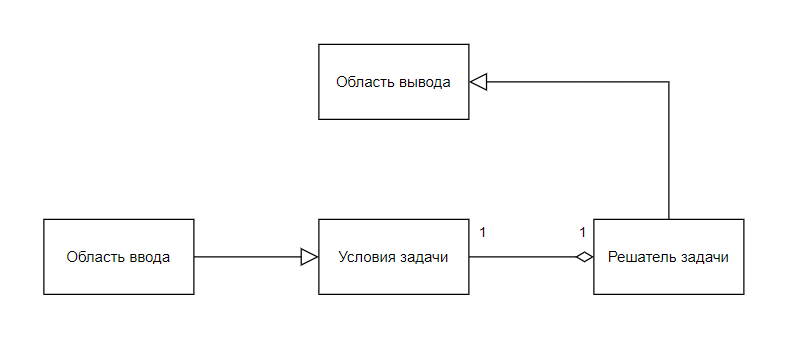


Рисунок 4 – Концептуальная диаграмма классов приложения

## 2.3 Анализ хранимой информации и выбор структур данных для ее представления

Основной информацией, которая необходима для написания программного кода симплекс-метода, является базис, а также сама пересчетная таблица. Информация такого рода будет хранится в течении всего времени работы программы, однако, есть необходимость хранить также промежуточные данные, использующиеся в циклах программы, пока ответ не будет найден.

При выборе структуры данных для представления базиса и пересчетной таблицы, стоит обратить внимание на следующее: структуры имеют четко определённые, заранее заданные размеры, имеется необходимость обращаться к элементам по индексам для использования в циклах, а также для изменения значения элемента. Для таких целей наилучшим выбором будет массив, который имеет все вышеперечисленные функции. Базис содержит лишь одно измерение, в то время как таблица содержит два, и будет хранится в многомерном массиве.

Для определения новой базисной переменной, точнее указания столбца с наибольшим коэффициентом целевой функции, есть необходимость хранить только лишь индекс выбранного столбца. Для таких целей достаточно целочисленной переменной, например, переменной типа int.

Определение свободной переменной проходит в два этапа:

* сохранение частных от деления соответствующих элементов базиса и ведущего столбца: массив данных;
* хранение минимального из них: целочисленная переменная.

При пересчете таблицы данные меняются в том же массиве, в котором хранятся.

## 2.4 Выбор методов и алгоритмов решения задачи

Для каждого из этапов симплекс-метода были использованы собственные алгоритмы:

1. Этап проверки критерия оптимальности: используется единственный метод, проходящий по всем элементам последней строки массива таблицы (доступ к элементам необходимой строки осуществляется с помощью индекса) для описка отрицательного элемента.

2. Поиск ведущего столбца: алгоритм проходит по всем элементам индексной строки и сохраняет ее данные в переменные, хранящие максимальное найденное значение и индекс элемента в строке. Сохраненный индекс и будет является номером ведущего столбца.

3. Перерасчет таблицы: алгоритм преобразует все старые значения из таблицы в новые с помощью формулы. Формула содержит значения, не только найденного на пересечении ведущих столбца и строки разрешающего элемента, но и значения всех элементов этих столбца и строки. После проведенных действий меняется и состав базиса таблицы, содержащийся в отдельно созданном массиве.

# 3 Проектирование структуры и компонентов программного продукта

## 3.1 Разработка интерфейса пользователя

### 3.1.1 Разработка структурной схемы интерфейса

При разработке интерфейса программы должны быть учтены все выполняемые ею функции: визуализация таблицы симплекс-метода, в том числе наличие в таблице раздела для вывода решения, возможность заполнить таблицу и очистить.

Учитывая все функции приложения, структурную схему интерфейса главного окна можно представить в следующем виде:

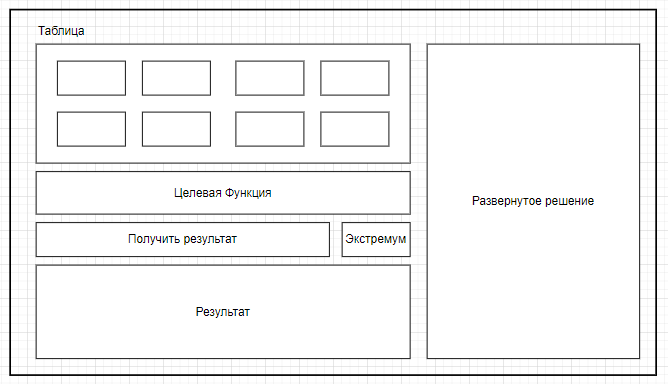


Рисунок 5 – Структурная схема интерфейса главного окна

### **3.1.2 Построение диаграммы состояний интерфейса**

При построении диаграммы состояний интерфейса следует учитывать специфику самого условия задачи: исследование выхода готовых комплектов:

Для реализации данной задачи следует предусмотреть несколько вариантов заполнения таблицы, в том числе тот, в котором таблица может быть полностью очищена. Все вышеперечисленные состояния отражены на следующей диаграмме:

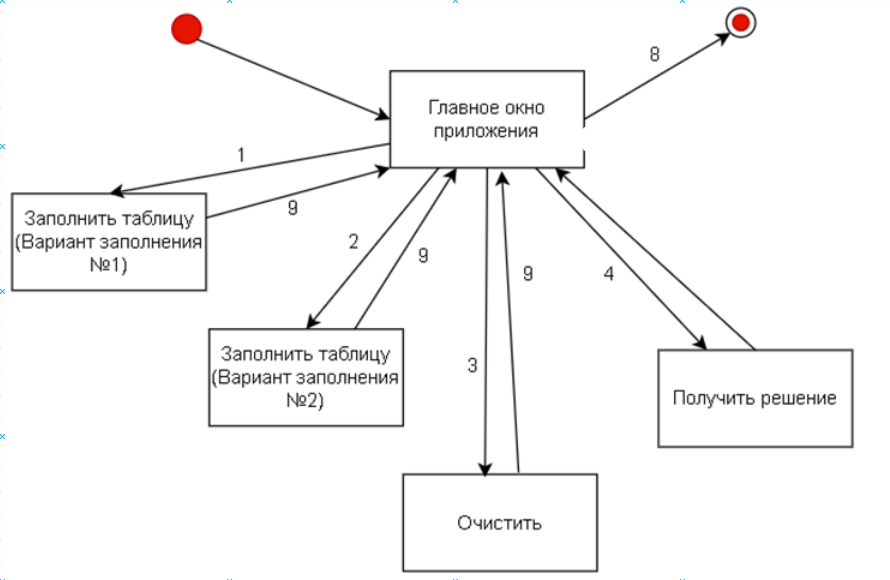


Рисунок 6 **–** Диаграмма состояний интерфейса

### **3.1.3 Разработка форм ввода-вывода информации**

При использовании данного приложения, большое значения имеет ввод-вывод информации, так как работа симплекс-метода без входных табличных значений, на основе которых выдается решение, невозможна. Приложение имеет несколько состояний:

* заполнение таблицы (вариант №1);
* заполнение таблицы (вариант №2);
* вывод решения;
* сообщение об отсутствии данных в таблице.

На приведенной ниже диаграмме, отображена взаимосвязь между состояниями приложения на основе данных, которыми эти состояния оперируют:

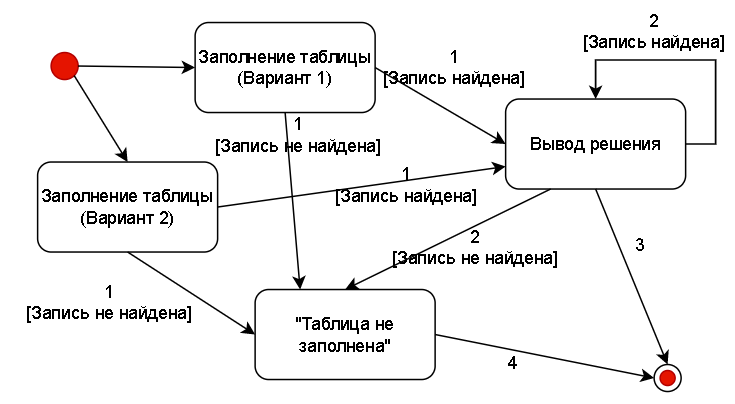


Рисунок 7 – Диаграмма состояний формы интерфейса

## 3.2 Разработка алгоритма основной программы и структурной схемы программного продукта

### **3.2.1 Описание структуры приложения и схема связности модулей**

При проектировании структуры будущего программного продукта стоит описать основные функциональные элементы продукта, а также характер их взаимодействия.

Основные функциональные элементы приложения касаются ввода информации и ее очистки, получении выходных данных. Все вышеперечисленные элементы можно отобразить на структурной схеме, которая в полной мере отражает их взаимодействие:

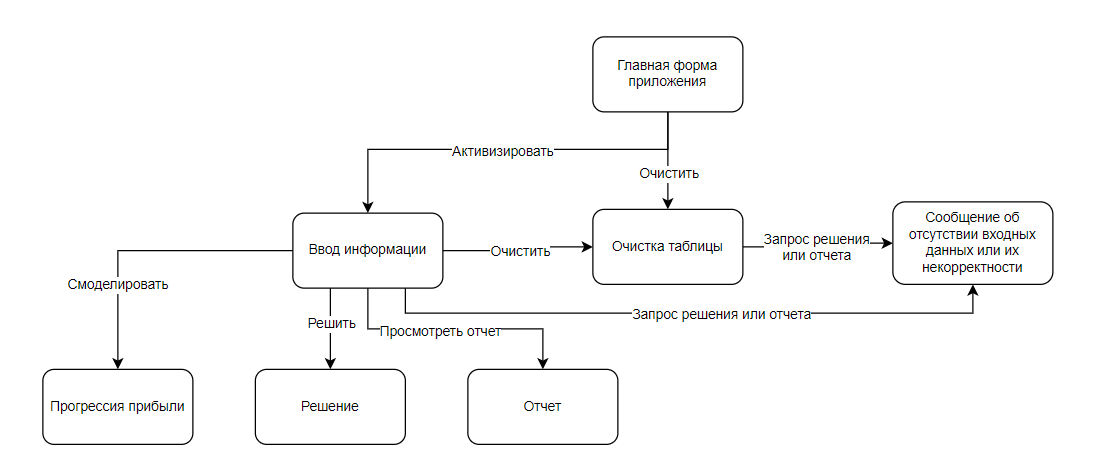


Рисунок 8 – Структурная схема программного продукта

### **3.2.2 Схема движения информационных потоков**

Схема движения информационных поток, содержит несколько уровней детализации. Главный уровень содержит абстракцию, которая заключает в себе весь алгоритм программы, а также входной и выходной потоки. Вследствие того, что абстракция содержит в себе весь программный код, табличные данные – будут являться входными данными, а решение – выходными:

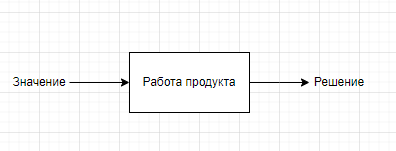


Рисунок 9 – Диаграмма потоков данных (уровень А0)

Далее главный уровень (А0) следует детализировать согласно состояниям программы и структурной схеме программного продукта. Диаграмма, приведенная на рисунке 9, отражает информационные потоки на уровне работы алгоритмов программы. Алгоритм получает входные данные в табличном формате, удобном для пользователя, затем заполняет структуры данных, заранее определенных программой, после чего управление передается симплекс-методу. Результаты работы симплекс-метода являются не полными в контексте задачи, вследствие чего некоторые значения необходимо вычислять дополнительно. Именно эти значения и передаются далее в решение.

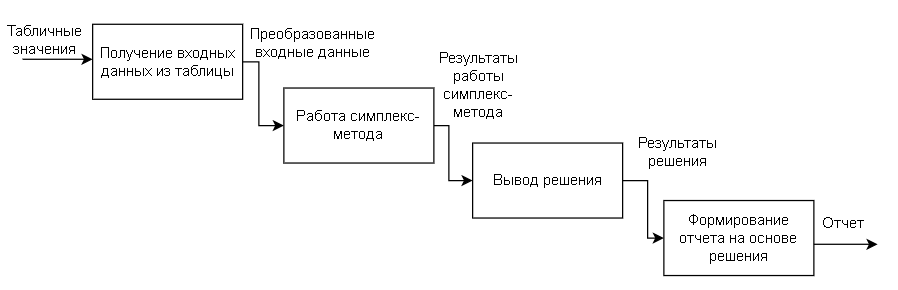


Рисунок 10 – Диаграмма потоков данных

## 3.3 Разработка основных алгоритмов программного продукта

Основной алгоритм программы представлен на следующем рисунке:

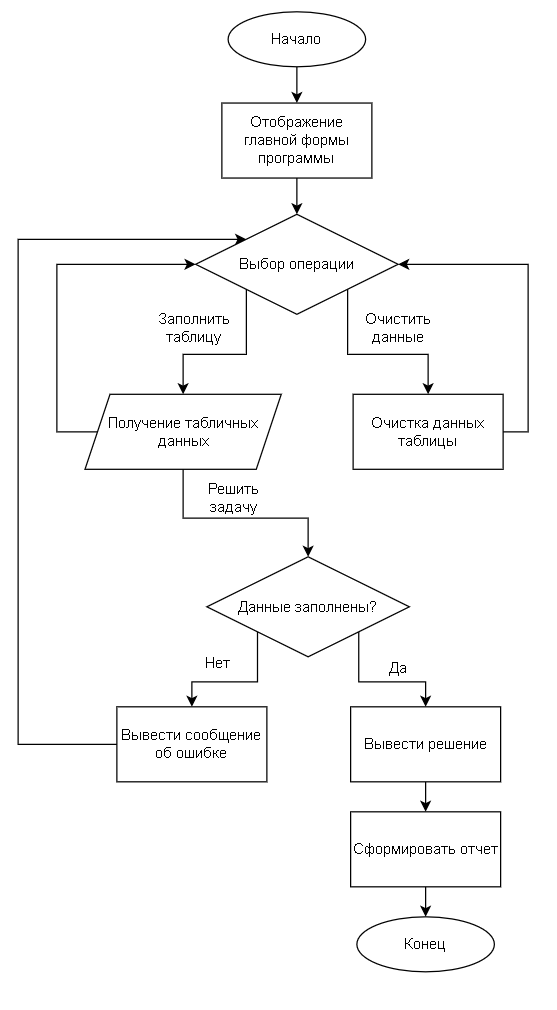


Рисунок 11 – Основной алгоритм программы

Основной алгоритм программы содержит в себе несколько подпрограмм, каждая из которых выполняет одно определенное действие.

1. Алгоритм получения табличных данных.

Используемые переменные: массив вещественного типа данных для хранения непосредственно табличных значений (x), массив целочисленного типа для хранения номеров переменных, входящих в базис (basis).

Схема алгоритма:

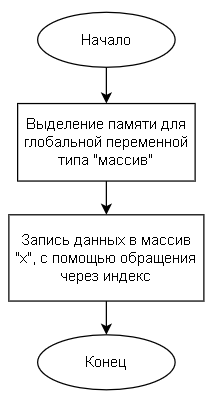


Рисунок 12 – Алгоритм подпрограммы получения данных

Функциональное назначение алгоритма: алгоритм предназначен для инициализации глобальной переменной x типа «двумерный массив», которая будет хранить данные, на протяжение всего времени работы программы, до момента изменения табличных данных, либо их полной очистки.

Входные данные: данные, введенные пользователем через интерфейс программы, либо заполненные автоматически, содержащие непосредственно симплекс-таблицу (x).

Выходные данные: инициализированный двумерный массив, а также одномерный массив, хранящий порядковые номера базисных переменных.

2. Алгоритм очистки данных таблицы.

Используемые переменные: массив вещественного типа данных для хранения непосредственно табличных значений (x), массив целочисленного типа для хранения номеров переменных, входящих в базис (basis).

Схема алгоритма:

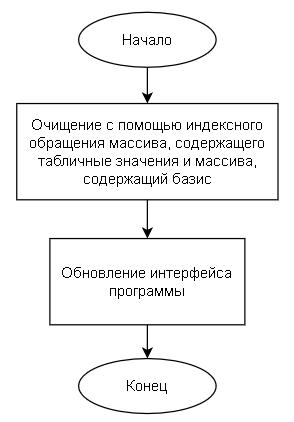


Рисунок 13 – Алгоритм подпрограммы очищения данных

Функциональное назначение алгоритма: алгоритм предназначен для полной очистки данных двух массивов: x и basis. После очищения, предыдущие данные не будут доступны в течении работы программы, до введения новых данных.

Входные данные: не требуются.

Выходные данные: пустые массивы, ранее содержащие табличные значения и базис.

3. Алгоритм получения решения.

Используемые переменные: массив вещественного типа данных для хранения непосредственно табличных значений (x), массив целочисленного типа для хранения номеров переменных, входящих в базис (basis), две целочисленные переменные для хранения индексов разрешающего элемента (index и index2).

Схема алгоритма:



Рисунок 14 – Алгоритм подпрограммы получения решения

Функциональное назначение алгоритма: алгоритм предназначен для решения задачи симплекс-методом, полученные значения также используются для расчета других значений.

Входные данные: двумерный массив табличных значений x, одномерный массив базисных переменных basis.

Выходные данные: двумерный массив x, содержащий готовое решение.

# 4 Тестирование программы

## 4.1 Разработка плана тестирования

При тестировании программного обеспечения, в первую очередь стоит обратить внимание на наличие сбоев в нем, возникающие при невозможности прочитать какие-либо данные, либо работать с пустыми данными, конвертировать один тип переменных в другой, выполнить определенные преобразования. Для предотвращения таких ошибок стоит обратить внимание на следующие пункты:

* если данные, использующиеся в решении не заполнены, программа должна выдавать сообщение об ошибке и не производить далее вычислений;

Вторым этапом тестирования программного продукта будет являться проверка правильности работы программы, в частности работы симплекс-метода.

И затем заключительным этапом будет являться проверка на правильность данных, которые были посчитаны на основе решения.

## 4.2 Разработка алгоритма процедуры тестирования

Учитывая этапы, описанные выше, можно составить следующий алгоритм проверки программы:

1. Проверка пользовательской истории при обращении к решению без входных данных.
2. Проверка работы симплекс-метода на заранее сформированных данных, а также сравнение результатов.
3. Сверка данных, отраженных в решении и полученных на основании решения.

## 4.3 Оценка результатов тестирования

Ниже приведены результаты выполнения разработанных для приложения модульных тестов (рисунок 20).

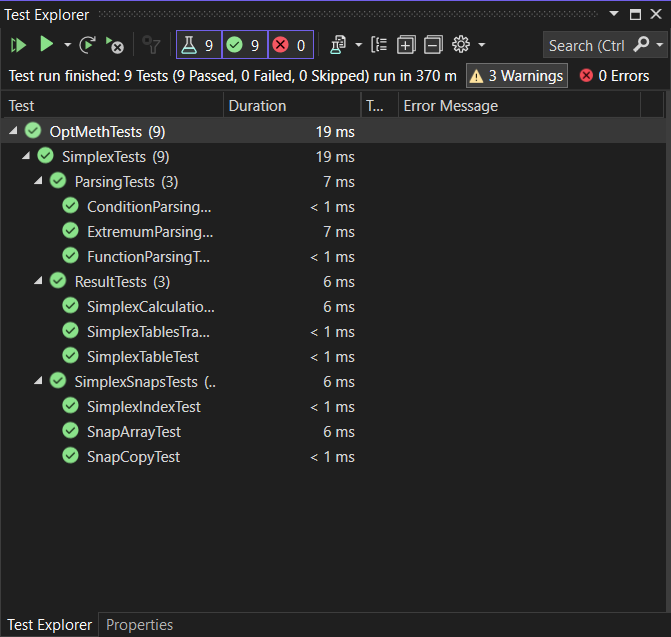


Рисунок 15 – Результат выполнения модульных тестов приложения

Планы и результаты тестирования интерфейса представлены в приложении А.

В данном разделе было проведено разноплановое тестирование программы. Были разработаны планы тестирования, разработан алгоритм процедуры тестирования и проведена оценка результатов. Основываясь на результатах тестирования приложения, можно сделать вывод о том, что оно полностью готово к эксплуатации, а ошибки, которые возможно обнаружатся в будущем будут несущественны и некритичны.

# 5 Сопровождение

1. Вход в приложение. При входе в программу открывается главное окно приложения:

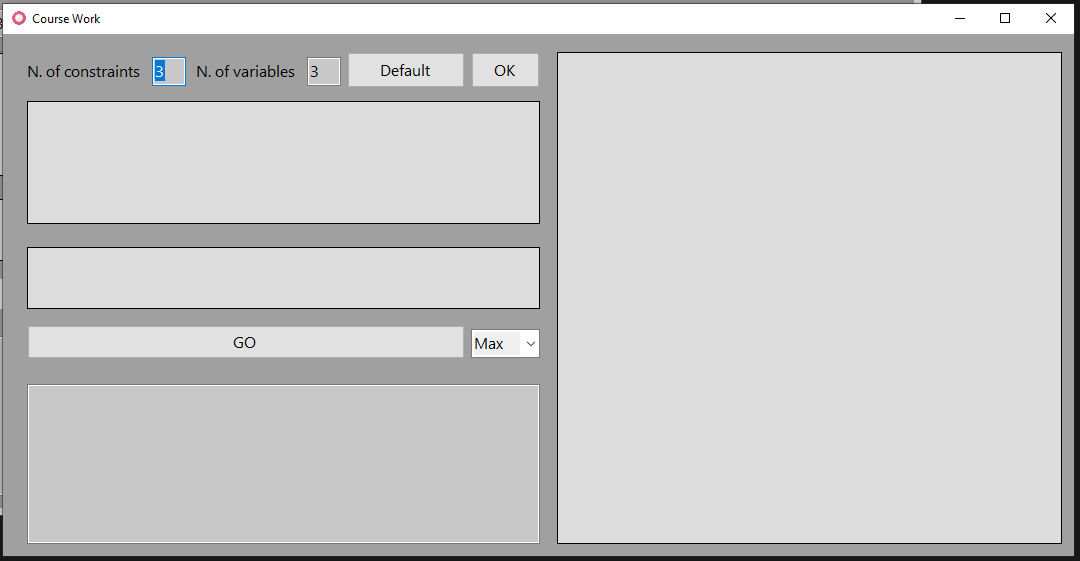


Рисунок 16 – Главное окно приложения

1. Заполнение таблицы. Для удобства использования в программе предоставлено автоматическое заполнение таблицы.
2. Получение решения. При использовании функции “Go” решение отображается в последней строке таблицы.

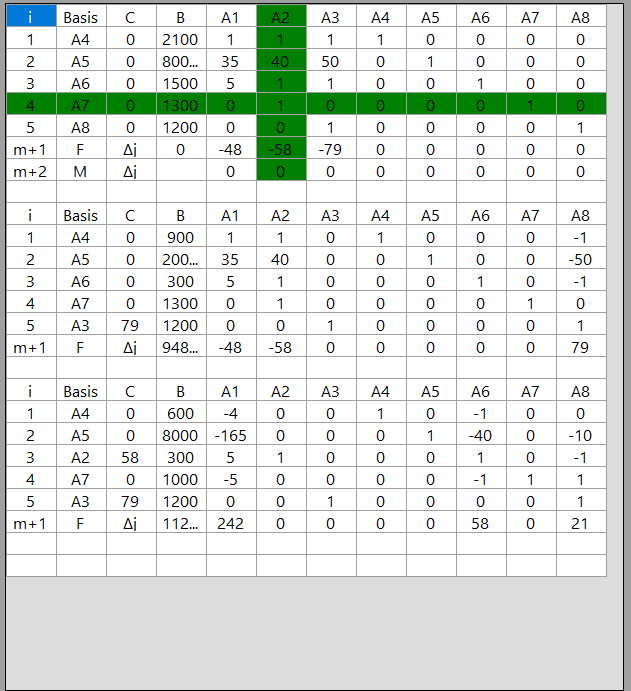


Рисунок 17.1 – Окно с результатами

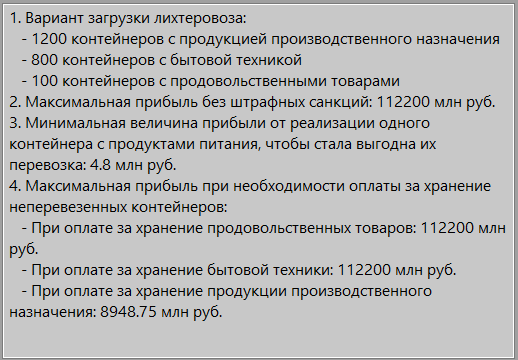


Рисунок 17.2 – Окно с результатами

# 6 Заключение

В результате выполнения курсовой работы было спроектировано и реализовано приложение по решению задачи оптимизации используя симплекс-метод.

В первом разделе был проведён анализ технических требований и уточнение спецификаций продукта. Сначала был проведён анализ задания и выбор технологий, языка программирования и среды разработки. Их результатом стал выбор объектно-ориентированной парадигмы программирования, выбор языка программирования C# и использование среды разработки Visual Studio 2022. Затем был проанализирован процесс обработки информации, на основе которого были построены функциональные диаграммы программы. Далее был проведён анализ хранимой информации и выбор структур данных для её представления – двумерные массивы.

Во втором разделе было проведено проектирование структуры и компонентов программного продукта. Был разработан интерфейс – его структурная схема, диаграмма его состояний и формы ввода-вывода информации. Был разработан алгоритм основной программы и структурная схема продукта. Произведено описание структуры приложения и составлена схема связности модулей. Разработаны основные алгоритмы программного продукта.

В третьем разделе было проведено разноплановое тестирование программы. Были разработаны планы тестирования, разработан алгоритм процедуры тестирования и проведена оценка результатов. Основываясь на результатах тестирования приложения, можно сделать вывод о том, что оно полностью готово к эксплуатации, а ошибки, которые возможно обнаружатся в будущем будут несущественны и некритичны.

В четвертом разделе было составлено руководство пользователя, описывающее и разъясняющее работу с разработанным продуктом. Были определены и описаны процессы обслуживания модели, алгоритма и программы, а также их эксплуатация.

После поэтапного решения задач был получен готовый продукт, соответствующий поставленным требованиям, однако он готов к дальнейшим изменениям и усовершенствованиям при необходимости, например в плане дизайна интерфейса или добавления возможности решения других типов задач оптимизации. Продукт был протестирован в соответствии с методологиями, изученными в процессе обучения. Также важно отметить полученные посредством изучения предметной области в процессе работы знания и опыт разработки.

# Список использованных источников

1. Балдин К. В. Математическое программирование / Балдин К.В., Брызгалов Н.А., Рукосуев А.В., - 2-е изд. - Москва :Дашков и К, 2018. - 218 с.: ISBN 978-5-394-01457-4. - Текст : электронный. - URL: https://znanium.com/bookread2.php?book=415097

2. Аттетков А. В. Методы оптимизации: Учебное пособие / А.В. Аттетков, В.С. Зарубин, А.Н. Канатников. - М.: ИЦ РИОР: НИЦ Инфра-М, 2019. - 270 с.: ил.; - (Высшее образование: Бакалавриат). - ISBN 978-5-16-103309-8. - Текст : электронный. - URL: https://znanium.com/bookread2.php?book=1002733

3. Карманов В. Г. Математическое программирование [Электронный ресурс] : Учебное пособие / В. Г. Карманов. - 6-е изд., испр. - Москва : ФИЗМАТЛИТ, 2018. - 264 с. - ISBN 978-5-9221-0983-3. - Текст : электронный. - URL: https://znanium.com/bookread2.php?book=544747

4. Каштанов В. А. Исследование операций (линейное программирование и стохастические модели) : учебник / В.А. Каштанов, О.Б. Зайцева. — Москва : КУРС, 2017. - 256 с. - ISBN 978-5-906818-78-2. - Текст : электронный. - URL: <https://znanium.com/bookread2.php?book=1017099>

5. Святослав Куликов. Тестирование программного обеспечения. Базовый курс, 2015 г.

6. Фленов М. «Библия C#» БХВ-Петербург, 2019 год, 4-е изд., перераб. и доп., 512 стр.

# Приложение А

**Проверка на уникальность**

Проверка на уникальность курсовой работы представлена на рисунке 20.

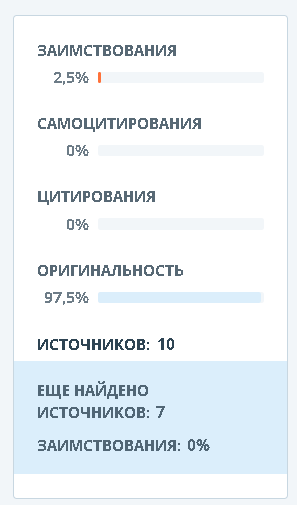


Рисунок 20 – Проверка на уникальность

# Приложение Б

**Листинг программы**

private void label14\_Click(object sender, EventArgs e)

{

if (x11.Text == "") { MessageBox.Show("Не все поля заполнены!"); return; }

Label[] labels = new Label[8];

labels[0] = l1;

labels[1] = l2;

labels[2] = l3;

labels[3] = l4;

labels[4] = l5;

labels[5] = l6;

labels[6] = l7;

labels[7] = l8;

int[] basis = new int[4] { 8, 9, 10, 11 };

double[,] result = GetResultMass(x, basis);

int k = 0;

foreach (int b in basis)

{

labels[b - 1].Text = result[k, 0].ToString();

k++;

}

l8.Text = l7.Text;

}

private double[,] GetResultMass(double[,] x, int[] basis)

{

while(allFIsBigger0(x))

{

//нашли наибольшее значение функции по модулю (посл строка)

int index = BiggestIndInF(x);

//поиск минимальной строки (В / xi)

double[] min = new double[4];

for (int i = 0; i < 4; i++)

{

if (x[i, index] <= 0) min[i] = 10000;

else min[i] = x[i, 0] / x[i, index];

}

int index2 = -1;

double min\_el = 1000;

for (int i = 0; i < min.Length; i++)

{

if(min[i] < min\_el)

{

index2 = i;

min\_el = min[i];

}

}

double razr = x[index2, index];

//пересчет одного из базовых элементов

int index\_baz = basis[index2];

for (int i = 0; i < 5; i++)

{

x[i, index\_baz] = 0;

}

x[index2, index\_baz] = 1;

//пересчет таблицы

double[] razr\_mas = new double[5];

for (int i = 0; i < 5; i++)

{

razr\_mas[i] = x[i, index];

}

for (int i = 0; i < 5; i++)

{

if (i == index2) continue;

for (int j = 0; j < 12; j++)

{

x[i, j] = x[i, j] - (x[index2, j]\*razr\_mas[i]) / razr;

}

}

for (int i = 0; i < 12; i++)

{

x[index2, i] = x[index2, i] / razr;

}

basis[index2] = index;

}

return x;

}

private bool allFIsBigger0(double[,] mas)

{

for (int i = 0; i < 12; i++)

{

if (mas[4, i] < 0) return true;

}

return false;

}

private int BiggestIndInF(double[,] mas)

{

int index = -1;

double max = -1;

for (int i = 1; i < 12; i++)

{

if (i == 11) continue;

if(Math.Abs(mas[4, i]) > max)

{

max = Math.Abs(mas[4, i]);

index = i;

}

}

return index;

}