МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіотехнічних систем літальних апаратів

Кафедра комп’ютерних систем та мереж

**Курсова робота**

|  |  |
| --- | --- |
| з | *Об’єктно-орієнтованого програмування* |
|  | (назва дисципліни) |
| на тему | *«Генератор документiв про вищу освiту*» |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Виконав: студент | | | *2* | курсу групи № | | | | *525a* |
| напряму підготовки (спеціальності) | | | | | | | | |
| *150102 – комп’ютерна інженерія* | | | | | | | | |
| (шифр і назва напряму підготовки (спеціальності)) | | | | | | | | |
| *Маркевич В.* | | | | | | | | |
| (прізвище й ініціали студента) | | | | | | | | |
| Керівник: | *старший викладач* | | | | | | | |
| *Бабешко Євген Васильович* | | | | | | | | |
| (посада, науковий ступінь, прізвище й ініціали) | | | | | | | | |
| Національна шкала: | | | | |  | | | |
| Кількість балів: | | | | |  | | | |
| Оцінка ECTS: | | | | |  | | | |
| Члени комісії: | |  | | | |  |  | |
|  | | (підпис) | | | |  | (прізвище й ініціали) | |
|  | |  | | | |  |  | |
|  | | (підпис) | | | |  | (прізвище й ініціали) | |
|  | |  | | | |  |  | |
|  | | (підпис) | | | |  | (прізвище й ініціали) | |

Харків – 2017

Міністерство освіти і науки України

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

|  |  |  |
| --- | --- | --- |
| Факультет | *радіотехнічних систем літальних апаратів* | |
|  | (повне найменування) | |
| Кафедра | *комп’ютерних систем та мереж* | |
|  | (повне найменування) | |
| Рівень вищої освіти | | *перший (бакалаврський)* |
|  | | (назва) |
| Напрям підготовки | | *150102 – комп’ютерна інженерія* |
|  | | (код та найменування) |

**ЗАВДАННЯ**

**НА КУРСОВИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ**

|  |  |  |
| --- | --- | --- |
| *Маркевич* | | |
| (прізвище, ім’я, по батькові) | | |
| 1. Тема проекту (роботи) | *«Генератор документiв про вищу освiту»* | |
|  | | |
|  | | |
| Керівник проекту (роботи) | | *Бабешко Євген Васильович старший викладач* |
|  | | ( прізвище, ім’я, по батькові, науковий ступінь, вчене звання) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2. Строк подання студентом проекту (роботи) | | | *25.05.2017* | |
| 3. Вихідні дані до проекту (роботи) | | *Фамілія, ім’я та по-батькові студенту,* | | |
| *дані про спеціальність, спеціалізацію, отриману кваліфікацію, пройдену учбову* | | | | |
| *програму.* | | | | |
|  | | | | |
|  | | | | |
| 4. Зміст пояснювальної записки (перелік завдань, які потрібно розв’язати) | | | |  |
| *Реферат, Постановка задачі, Проектування, Розробка, Верифікація та* | | | | |
| *тестування, Висновок, Перелік посилань, Додаток А. Технічне завдання,* | | | | |
| *Додаток Б. Вихідні тексти програми, Додаток В. Керівництво Оператора,* | | | | |
| *Додаток Г. Презентація* | | | | |
|  | | | | |
|  | | | | |
| 5. Перелік графічного матеріалу | *діаграма варіантів використання,* | | | |
| *діаграма послідовності, діаграма пакетів, діаграма класів,* | | | | |
| *алгоритм гомеморфного стиснення,алгоритм перевірки графа на ізоморфізм.* | | | | |
| *.* | | | | |
|  | | | | |

6. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
| *Основний* | *Бабешко Є. В. старший викладач* | *24.01.2017* | *24.01.2017* |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Нормоконтроль |  | *Бабешко Є. В.* | *«* | *25* | *»* | *05* | *20* | *17* | *р* |
|  | (підпис) | (ініціали та прізвище) |  |  |  |  |  |  |  |

7. Дата видачі завдання « 24 » 01 2017 р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів курсового  проекту (роботи) | Строк виконання етапів проекту  ( роботи ) | Примітка |
| *1* | *Реферат* | *24.01.2017* |  |
| *2* | *Постанова задачі* | *07.02.2017* |  |
| *3* | *Розробка ТЗ* | *21.02.2017* |  |
| *4* | *Проектування* | *07.03.2017* |  |
| *5* | *Розробка* | *21.03.2017* |  |
| *6* | *Тестування та верифікація* | *04.04.2017* |  |
| *7* | *Висновки* | *18.04.2017* |  |
| *8* | *Розробка керівництва оператора* | *02.04.2017* |  |
| *9* | *Оформлення пояснювальної записки* | *16.04.2017* |  |
| *10* | *Розробка презентації* | *16.04.2017* |  |

|  |  |  |
| --- | --- | --- |
| **Студент** |  | *Маркевич В.* |
|  | (підпис) | (прізвище та ініціали) |
| **Керівник проекту (роботи)** |  | *Бабешко Є.В.* |
|  | (підпис) | (прізвище та ініціали) |

РЕФЕРАТ

Пояснительная записка на курсовой проект: 67 с., 65 ил., 12 табл., 4 приложения, 10 источников.

Данная работа посвящена задаче по созданию генератора документов о высшем образовании.

Целью данной работы является разработка приложения, которое может быть использовано для генерации документов о высшем образовании..

Приложение было реализовано с помощью языка программирования С#. За графический интерфейс отвечал Windows Forms. Для доступа к объектам файла Word использовалась библиотека Microsoft.Interop.Word.

Содержание

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc484445558)

[1.1 Анализ предметной области 5](#_Toc484445559)

[1.2 Анализ существующих решений 9](#_Toc484445560)

[1.3 Постановка задачи 9](#_Toc484445561)

[2 ПРОЕКТИРОВАНИЕ 10](#_Toc484445562)

[2.1 Выбор и обоснование архитектуры 10](#_Toc484445563)

[2.2 Диаграмма прецедентов 11](#_Toc484445564)

[2.3 Диаграмма последовательности 13](#_Toc484445565)

[2.4 Проектирование графического интерфейса 14](#_Toc484445566)

[3 РАЗРАБОТКА 19](#_Toc484445567)

[3.1 Диаграмма пакетов 19](#_Toc484445568)

[3.2 Диаграмма классов 20](#_Toc484445569)

[3.3 Разработка и описание ключевых методов классов 22](#_Toc484445570)

[4 ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММЫ 29](#_Toc484445571)

[4.1 Верификация 29](#_Toc484445572)

[4.2 Тестирование программы 33](#_Toc484445573)

[ЗАКЛЮЧЕНИЕ 43](#_Toc484445574)

[ПЕРЕЧЕНЬ ССЫЛОК 44](#_Toc484445575)

[ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 45](#_Toc484445576)

[ПРИЛОЖЕНИЕ Б. ИСХОДНЫЕ КОДЫ ПРОГРАММЫ 48](#_Toc484445577)

[ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 66](#_Toc484445578)

[ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ 74](#_Toc484445579)

# ПОСТАНОВКА ЗАДАЧИ

## 1.1 Анализ предметной области

Текстовый файл — компьютерный [файл](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), содержащий [текстовые данные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5). Текстовым файлам противопоставляются [двоичные (бинарные) файлы](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB), в которых содержатся данные, не рассчитанные на интерпретацию в качестве текстовых (например, файлы, хранящие текст в закодированном или сжатом виде, или хранящие не текст, а звук, изображение или иные данные).

В отличие от термина «[текстовые данные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5)» («текстовый формат данных»), характеризующего содержимое данных, термин «текстовый файл» относится к файлу и характеризует его как контейнер, хранящий такие данные.

Текстовый файл содержит последовательность символов (в основном печатных знаков, принадлежащих тому или иному набору символов). Эти символы обычно сгруппированы в строки. В современных системах строки разделяются разделителями строк, в прошлом же применялось хранение строк в виде записей постоянной или переменной длины. Иногда конец текстового файла (особенно если в файловой системе не хранится информация о размере файла) также отмечается одним или более специальными знаками, известными как маркеры конца файла. [1].

Microsoft Word (часто — MS Word, WinWord или просто Word) — [текстовый процессор](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80), предназначенный для создания, просмотра и редактирования [текстовых документов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB), с локальным применением простейших форм [таблично](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0)-[матричных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))алгоритмов. Выпускается [корпорацией Microsoft](https://ru.wikipedia.org/wiki/Microsoft) в составе [пакета](https://ru.wikipedia.org/wiki/%D0%9E%D1%84%D0%B8%D1%81%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82) [Microsoft Office](https://ru.wikipedia.org/wiki/Microsoft_Office). Первая версия была написана [Ричардом Броди](https://ru.wikipedia.org/w/index.php?title=%D0%91%D1%80%D0%BE%D0%B4%D0%B8,_%D0%A0%D0%B8%D1%87%D0%B0%D1%80%D0%B4&action=edit&redlink=1) ([Richard Brodie](https://en.wikipedia.org/wiki/Richard_Brodie)) для [IBM PC](https://ru.wikipedia.org/wiki/IBM_PC), использующих [DOS](https://ru.wikipedia.org/wiki/DOS), в 1983 году. Позднее выпускались версии для [Apple Macintosh](https://ru.wikipedia.org/wiki/Macintosh)(1984), [SCO UNIX](https://ru.wikipedia.org/wiki/SCO_UNIX) и [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows) (1989). Текущей версией является Microsoft Office Word 2016 для [Windows](https://ru.wikipedia.org/wiki/Windows) и [macOS](https://ru.wikipedia.org/wiki/MacOS).

Microsoft Word является наиболее популярным из используемых в данный момент текстовых процессоров, что сделало его бинарный формат документа стандартом де-факто, и многие конкурирующие программы имеют поддержку совместимости с данным форматом. Расширение «.doc» на платформе IBM PC стало синонимом двоичного формата Word 97—2000. .doc или .DOC (от англ. document) — расширение имени файла, используемое для файлов, представляющих текст, с разметкой или без. Расширение «.doc» часто использовалось для обозначения текстовых файлов, содержащих текст без форматирования, позже стало использоваться для двоичных форматов с разметкой. Фильтры экспорта и импорта в данный формат присутствуют в большинстве текстовых процессоров. Формат документа разных версий Word меняется, различия бывают довольно тонкими [2].

При разработке решений Word в Visual Studio выполняется взаимодействие с объектной моделью Word. Эта объектная модель состоит из классов и интерфейсов, которые предоставляются в основной сборке взаимодействия для Word и определены в пространстве имен Microsoft.Office.Interop.Word.

## Анализ существующих решений

Поскольку данная задача является строго прикладной, то аналогов у неё не существует, т.к. разрабатывается она под специальное техническое задание.

## Постановка задачи

Используя динамическую библиотеку, следует разработать программу, которая будет генерировать по государственному образцу .doc-файл-форму диплома о получении степени бакалавра, сохраняя уникальное значение номера сертификата.

# ПРОЕКТИРОВАНИЕ

## 2.1 Выбор и обоснование архитектуры

Качественная архитектура является основой разработки любого проекта. Таковой можно считать только выгодную архитектуру, которая упрощает процесс создания программы и делает его более понятным. Таким образом можно сказать, что чем проще становится разработчику понять его задачу, тем лучше организована программа. Также результатом хорошо продуманной архитектуры является простота отладки программы, возможность вносить поправки и расширять возможности, не ломая при этом того, что было создано ранее. Архитектура программного обеспечения начинается с набора требований. Они могут быть выражены в форме диаграмм, блок-схем процесса, моделей или документированных списков задач эксплуатации, которые должно выполнять программное обеспечение. Обычно клиент или партнер также выражает менее точные требования, такие как внешний вид или способ работы определенных пользовательских интерфейсов для часто встречающихся задач. Требования также должны включать в себя информацию о существующем программном обеспечении, системах, оборудовании и сетях, с которыми будет взаимодействовать новое программное обеспечение; и другие факторы, такие как план развертывания и обслуживания, и, конечно же, доступный бюджет проекта[8].

В данном проекте будет использоваться трехуровневая архитектура. Разрабатываемая система разделается на три уровня, уровень представления или же интерфейс, уровень логики, которая включает в себя реализацию алгоритмов и уровень файловой системы.



Рисунок 2.1 – Архитектура данного проекта

## Диаграмма прецедентов

Функции, которые реализованные в проекте, можно визуализировать с помощью диаграммы прецедентов.



Рисунок 2.2 – Диаграмма прецедентов

В таблицах 2.1 – 2.7 описаны прецеденты представленные на рисунке 2.2

Таблица 2.1 – Прецедент ввода данных с клавиатуры

|  |  |
| --- | --- |
| Название прецедента | «Ввод данных с клавиатуры» |
| Описание | Данный прецедент позволяет вводить данные с клавиатуры. |

Таблица 2.2 – Прецедент проверки

|  |  |
| --- | --- |
| Название прецедента | «Проверка на корректность» |
| Описание | Прецедент проверяет входные данные на корректность. |

Таблица 2.3 – Прецедент копирования

|  |  |
| --- | --- |
| Название прецедента | «Копирование документа-образца» |
| Описание | Прецедент осуществляет копирование документа с сохраненными стилями в указанное место для последующей редакции. |

Таблица 2.4 – Прецедент корректирования данных

|  |  |
| --- | --- |
| Название прецедента | «Корректирование данных» |
| Описание | Прецедент реализует замену кодовых слов в документе на данные о студенте. |

Таблица 2.5 – Прецедент сохранения

|  |  |
| --- | --- |
| Название прецедента | «Сохранение документа» |
| Описание | Прецедент сохраняет .doc файл с изменениями |

## Диаграмма последовательности



Рисунок 2.3 – Диаграмма последовательности

## Проектирование графического интерфейса

В ходе разработки пользовательского интерфейса было реализовано оконное приложение с использованием WindowForms [9], которое соответствует всем описанным в техническом задании требованиям.

Приложение состоит из главной формы:

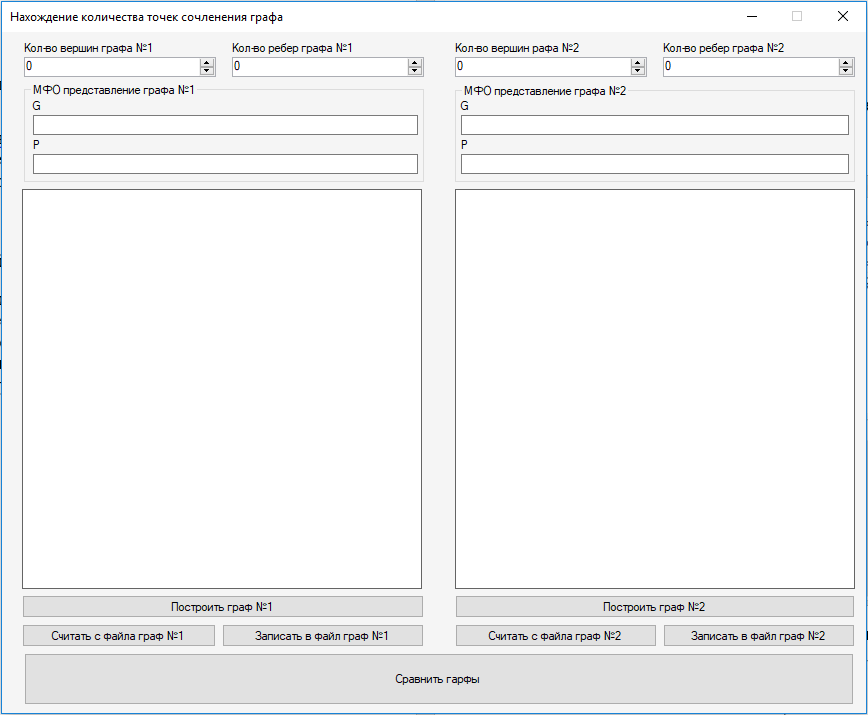


Рисунок 2.4 – Внешний вид главного окна приложения

Рабочая область содержит такие кнопки:

1. «Построить граф №1» – построение графа №1 по исходным массивам.
2. «Считать с файла граф №1» - считывает данные из файла для графа №1.
3. «Записать в файл граф №1» - запись MFO представления графа №1 в файл формата json.
4. «Построить граф №2» – построение графа №2 по исходным массивам.
5. «Записать в файл граф №2» - запись MFO представления графа №2 в файл формата json.
6. «Сравнить графы» - вывод результатов нахождения числа точек сочленения графов и проверка этих графов на эквивалентность.

Пример работы:

1. Ввод осуществляется при помощи клавиш либо чтением с файла формата json. 2.5.

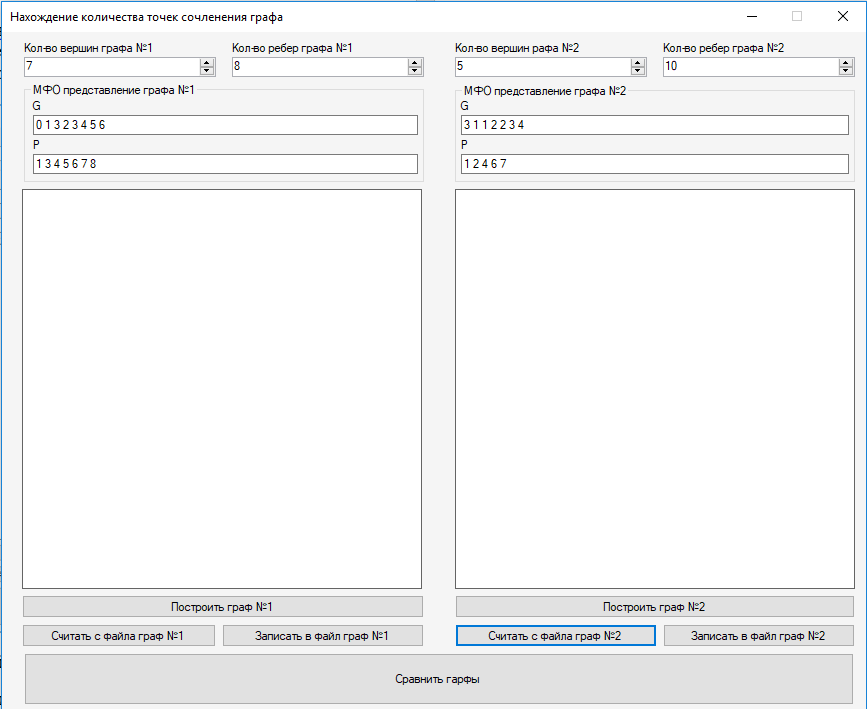


Рисунок 2.5 – Полученные результаты после загрузки файлов

1. Операции по поиску числа точек сочленения графа и сравнение графов на эквивалентность приведены на рисунке 2.7 и 2.8.

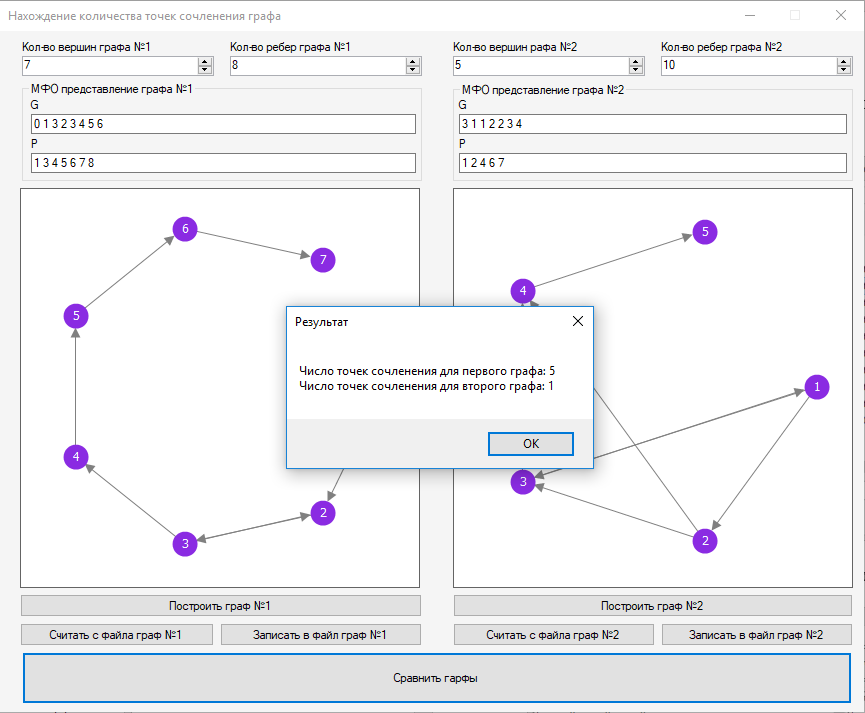


Рисунок 2.7 – Полученные результаты после выполнения операции поиска числа точек сочленения

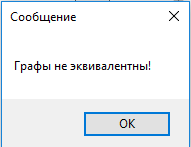


Рисунок 2.8 – Результат проверки на эквивалентность

1. Ввод данных из файла и результат ввода представлен на рисунке 2.9 и 2.10 соответственно.

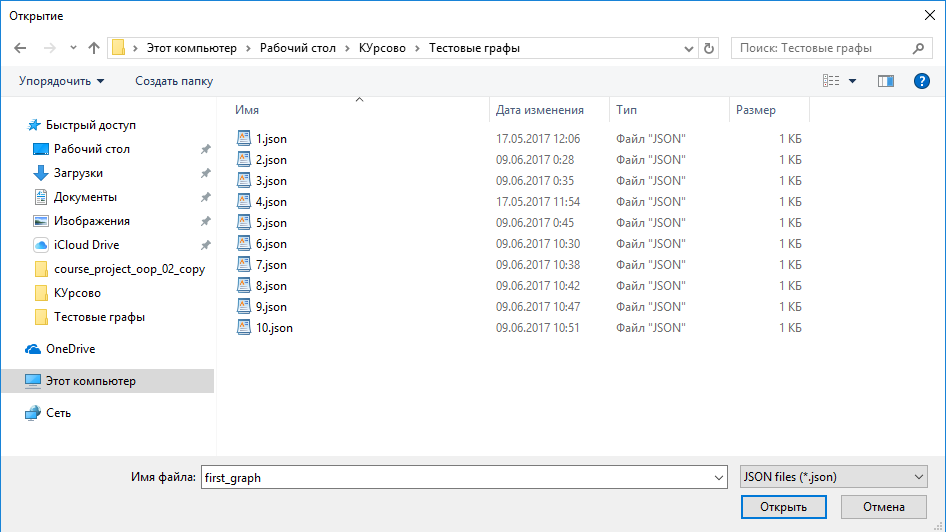


Рисунок 2.9 Выбор файла, из которого будет введен граф

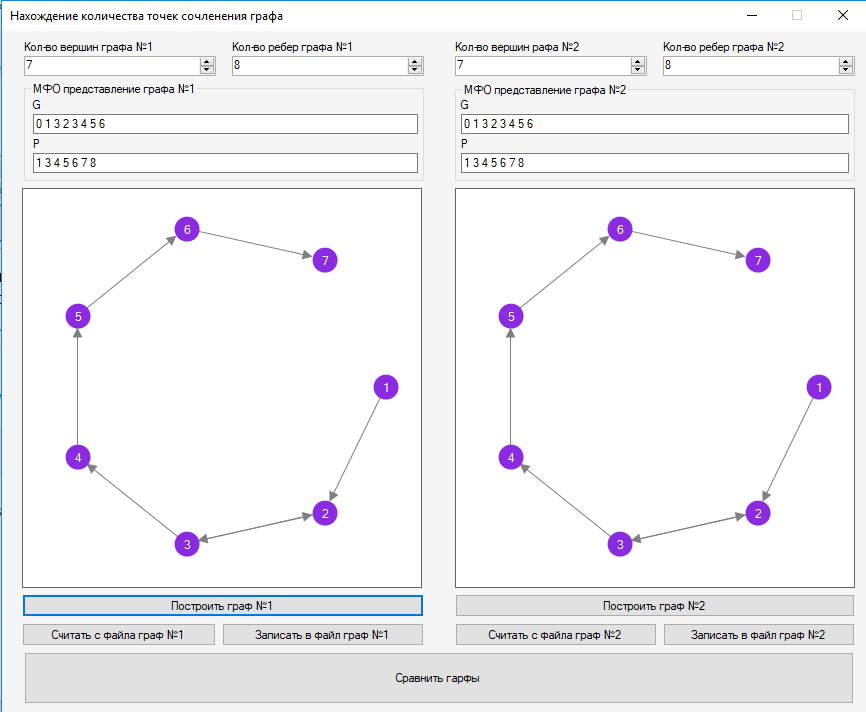


Рисунок 2.10 Графическое представление графов

# РАЗРАБОТКА

## Диаграмма пакетов



Рисунок 3.1 – Диаграмма пакетов

На диаграмме пакетов, изображеной на рисунке 3.1 показана трехурованевая организация архитектуры. Класс MainForm относится к уровню интерфейса. Класс ModelingOfGraph реализует логику по нахожденю числа точек сочленения графа. FileSystem – реализует работу чтения и записи в файл.

## 3.2 Диаграмма классов

В ходе выполнения данной работы для решения поставленной задачи были разработаны классы DataModifiing, GraphBuilder, GraphReadingAndWriting, IReadingAndWritingGraph, MainForm, ModelingOfGraph которые представлены на рисунке 3.2.

Описание:

1. DataModifiing – класс для преобразования и проверки матриц;
2. GraphBuilder – класс для визуализации графов;
3. GraphReadingAndWriting – класс для чтения и записи данных;
4. IReadingAndWritingGraph – обобщенный интерфейс для чтения и записи данных в формате JSON;
5. MainForm – класс формы;
6. ModelingOfGraph – класс для решения задачи нахождения числа точек сочленения в графе.

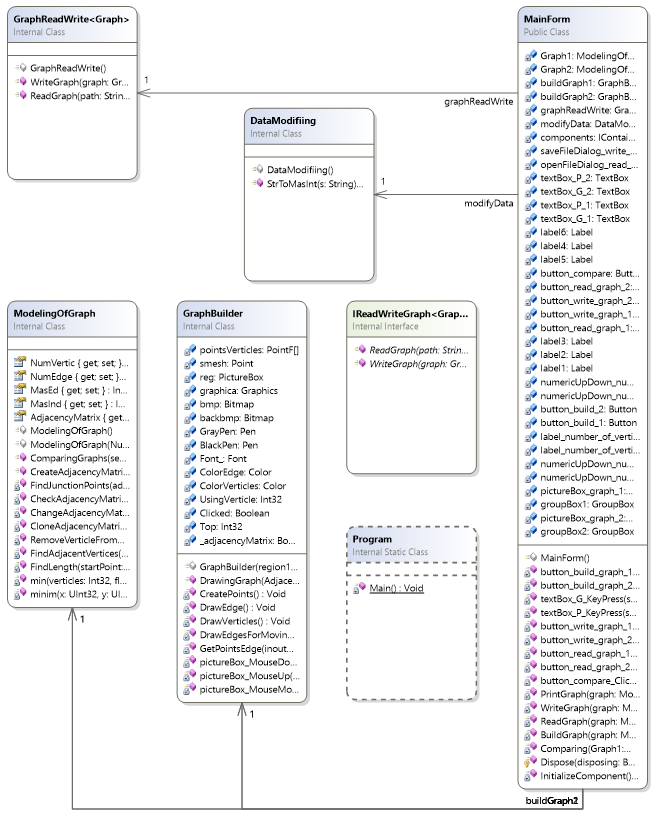


Рисунок 3.2 – Диаграмма классов

## Разработка и описание ключевых методов классов



Рисунок 3.3 – Общий алгоритм работы программы



Рисунок 3.4 – Общий алгоритм нахождения числа точек сочленения графа

В таблице 3.1 представлены поля и методы класса DataModifiing

Таблица 3.1 – Таблица со списком полей и методов класса DataModifiing.

|  |  |  |
| --- | --- | --- |
|  | *Методы* | *Назначение* |
| 1. | StrToMasInt | Преобразовать строку в целочисленный массив, где числа в строке разделены пробелами |

В таблице 3.2 представлены поля и методы класса GraphBuilder

Таблица 3.2 – Таблица со списком полей и методов класса GraphBuilder.

|  |  |  |
| --- | --- | --- |
|  | *Поля* | *Назначение* |
| 1. | pointsVerticles | Массив точек для хранения координат вершин графа |
| 2. | smesh | Переменная для хранения смещения вершины |
| 3. | reg | Область для рисования графа |
| 4. | graphica | Ссылка на объект класса Graphics - предоставление средств для рисования |
| 5. | bmp | Ссылка на объект класса Bitmap - хранение изображения полного графа |
| 6. | backbmp | Ссылка на объект класса Bitmap - хранения изображения не полного графа(только тех частей, которые являются неподвижными во время передвижения вершины графа) |
| 7. | GrayPen | Карандаш для рисования ребер, которые не двигаются |
| 8. | BlackPen | Карандаш для рисования ребер, которые двигаются |
| 9. | Font\_ | Шрифт для подписи номеров вершин |
| 10. | ColorEdge | Цвет для рисования подвижных ребер |
| 11. | ColorVerticles | Цвет для рисования вершин |
| 12. | UsingVerticle | Индекс подвижной вершины |
| 13. | Clicked | Переменная для хранения результата попадания точки в область - true, если есть попадание |
| 14. | Top | Количество вершин |
| 15. | \_adjacencyMatrix | Матрица смежности |
|  | *Методы* | *Назначение* |
| 1. | GraphBuilder | Конструктор с параметрами - задание области для рисования |
| 2. | DrawingGraph | Нарисовать граф |
| 3. | CreatePoints | Создать массив координат вершин |
| 4. | DrawEdge | Нарисовать ребра |
| 5. | DrawVerticles | Нарисовать вершины |
| 6. | DrawEdgesForMovingVerticle | Нарисовать подвижные ребра(те, которые связаны с подвижной вершиной) |
| 7. | GetPointsEdge | Получить координаты точки для построения ребра |
| 8. | pictureBox\_MouseDown | Обработчик для проверки попадания точки в одну из вершин |
| 9. | pictureBox\_MouseUp | Обработчик для перерисовки графа после отпускания клавиши мыши |
| 10. | pictureBox\_MouseMove | Обработчик для рисования графа во время передвижения вершины |

В таблице 3.3 представлены поля и методы класса GraphReadingAndWriting

Таблица 3.3 – Таблица со списком полей и методов класса GraphReadingAndWriting

|  |  |  |
| --- | --- | --- |
|  | *Методы* | *Назначение* |
| 1. | WriteGraph | Записать данные в файл |
| 2. | ReadGraph | Прочитать данные из файла |

В таблице 3.4 представлены поля и методы класса ModelingOfGraph

Таблица 3.4 – Таблица со списком полей и методов класса ModelingOfGraph

|  |  |  |
| --- | --- | --- |
|  | *Поля* | *Назначение* |
| 1. | NumVertic | Количество вершин |
| 2. | NumEdge | Количество ребер |
| 3. | MasEd | Массив MasEd - массив ребер |
| 4. | MasInd | Массив MasInd - массив индексов для обращения в первый массив |
| 5. | AdjacencyMatrix | Матрица смежности |
|  | *Методы* | *Назначение* |
| 1. | ModelingOfGraph | Конструктор по умолчанию |
| 2. | ModelingOfGraph | Конструктор с параметрами - задание количества вершин NumVertic, к-ва ребер NumEdge и MFO |
| 3. | ComparingGraphs | Сравнить графы на эквивалентность |
| 4. | CreateAdjacencyMatrix | Создать матрицу смежности из полученного MFO |
| 5. | FindJunctionPoints | Найти точки сочленения |
| 6. | CheckAdjacencyMatrix | Проверить матрицу на первом шаге - если граф не связный в самом начале, то точки сочленения больше искать не нужно |
| 7. | ChangeAdjacencyMatrix | Дополнить матрицу связями для корректной работы поиска пути между вершинами |
| 8. | CloneAdjacencyMatrix | Клонировать матрицу смежности |
| 9. | RemoveVerticleFromAdjacencyMatrix | Удалить вершину из матрицы смежности |
| 10. | FindAdjacentVertices | Найти смежные вершины для заданной вершины |
| 11. | FindLength | Найти длину пути из начальной точки в конечную |
| 12. | min | Метод для поддержки работы метода FindLength |
| 13. | minimum | Метод для поддержки работы метода FindLength |

В таблице 3.5 представлены поля и методы класса GraphForm

Таблица 3.5 – Таблица со списком полей и методов класса GraphForm

|  |  |  |
| --- | --- | --- |
|  | *Поля* | *Назначение* |
| 1. | Graph1 | Ссылка на объект класса ModelingOfGraph - первый граф |
| 2. | Graph2 | Ссылка на объект класса ModelingOfGraph - второй граф |
| 3. | buildGraph1 | Ссылка на объект класса GraphBuilder - построить первый граф |
| 4. | buildGraph2 | Ссылка на объект класса GraphBuilder - построить второй граф |
| 5. | graphReadWrite | Ссылка на объект класса GraphReadWrite - чтение и запись данных |
| 6. | modifyData | Ссылка на объект класса ModifiingData - проверка и модифицирование данных |
|  | *Методы* | *Назначение* |
| 1. | GraphForm | Конструктор по умолчанию |
| 2. | button\_build\_graph\_1\_Click | Обработчик для построения первого графа |
| 3. | button\_build\_graph\_2\_Click | Обработчик для построения второго графа |
| 4. | textBox\_G\_KeyPress | Обработчик для проверки данных в массиве MasEd |
| 5. | textBox\_P\_KeyPress | Обработчик для проверки данных в массиве MasInd |
| 6. | button\_write\_graph\_1\_Click | Обработчик для записи первого графа в файл |
| 7. | button\_write\_graph\_2\_Click | Обработчик для записи второго графа в файл |
| 8. | button\_read\_graph\_1\_Click | Обработчик для чтения первого графа из файла |
| 9. | button\_read\_graph\_2\_Click | Обработчик для чтения второго графа из файла |
| 10. | button\_compare\_Click | Обработчик для сравнения графов |
| 11. | PrintGraph | Вывести граф в поля редактирования |
| 12. | WriteGraph | Записать граф в файл |
| 13. | ReadGraph | Прочитать граф из файла |
| 14. | BuildGraph | Построить граф |
| 15. | Comparing | Сравнить графы |

# ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММЫ

## 4.1 Верификация

Верификацией называется проверка соответствия результатов отдельных этапов разработки программной системы требованиям и ограничениям, сформулированным для них на предыдущих этапах. [10].

Согласно техническому заданию для реализации поставленной задачи был разработан проект, который позволяет:

1. Осуществлять выбор ввода данных (с клавиатуры или из файла формата json).
2. Задавать количество ребер, вершин и MFO-представление.
3. Проверять на корректность введенные данные.
4. Выполнение операции поиска числа точек сочленения графа.
5. Визуализация графов.

Задачу 1 выполняет прецеденты: «Ввод данных с клавиатуры» и «Чтение данных с файла», результат представлен на рисунке 4.1

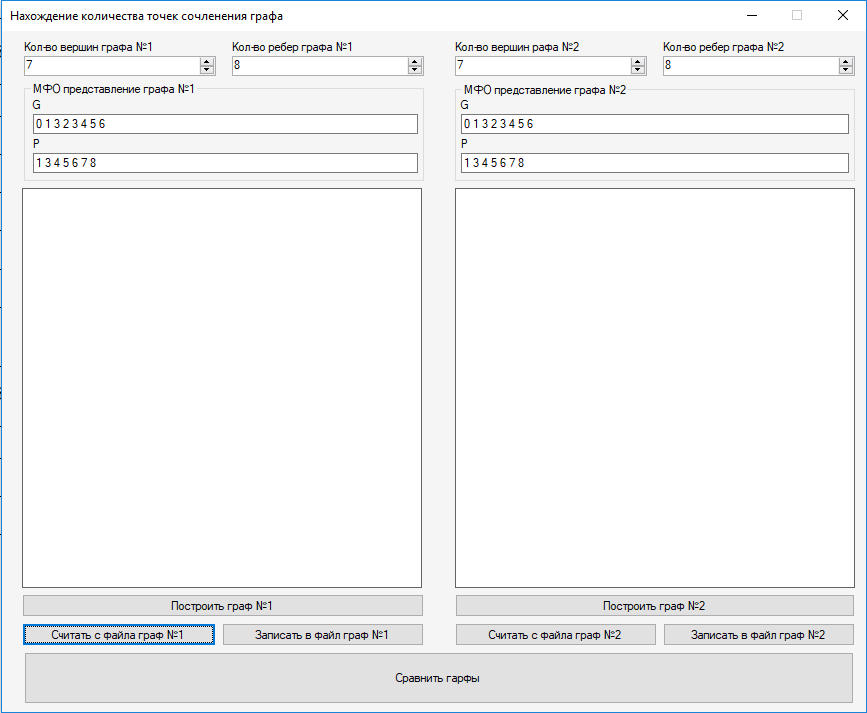


Рисунок 4.1 – Иллюстрация выполнения задачи 1

Задачу №2 выполняет прецедент «Ввод данных с клавиатуры», результат представлен на рисунке 4.2

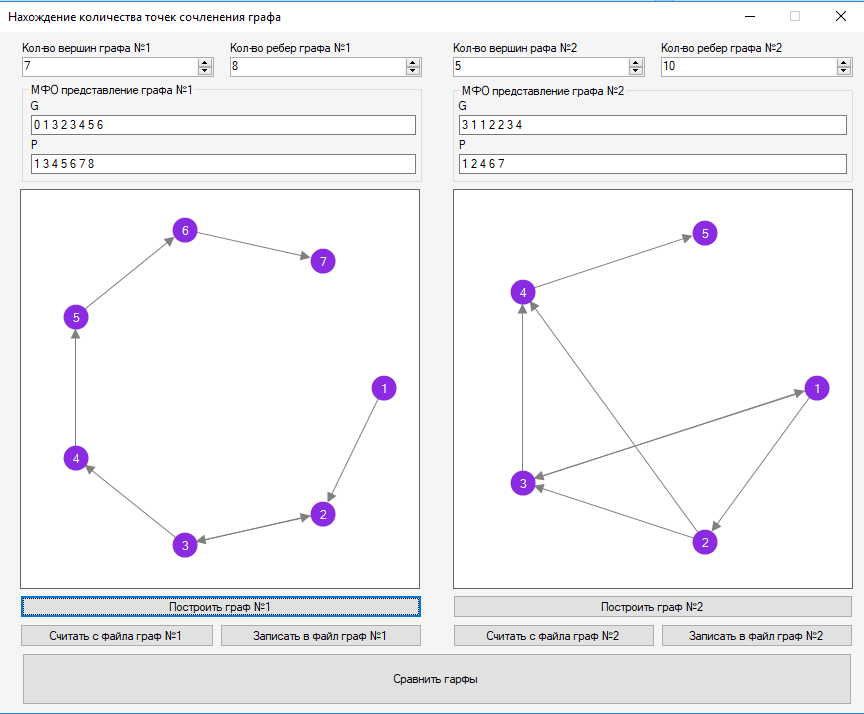


Рисунок 4.2 – Иллюстрация выполнения задачи 2

Задачу №3 реализует прецедент «Проверка на корректность», результат представлен на рисунке 4.3

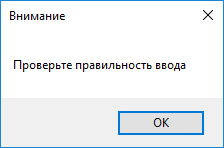


Рисунок 4.3 – Иллюстрация выполнения задачи 3

1. Задачу №4 реализует прецедент «Выполнение операции поиска числа точек сочленения графа», результат представлен на рисунке 4.4

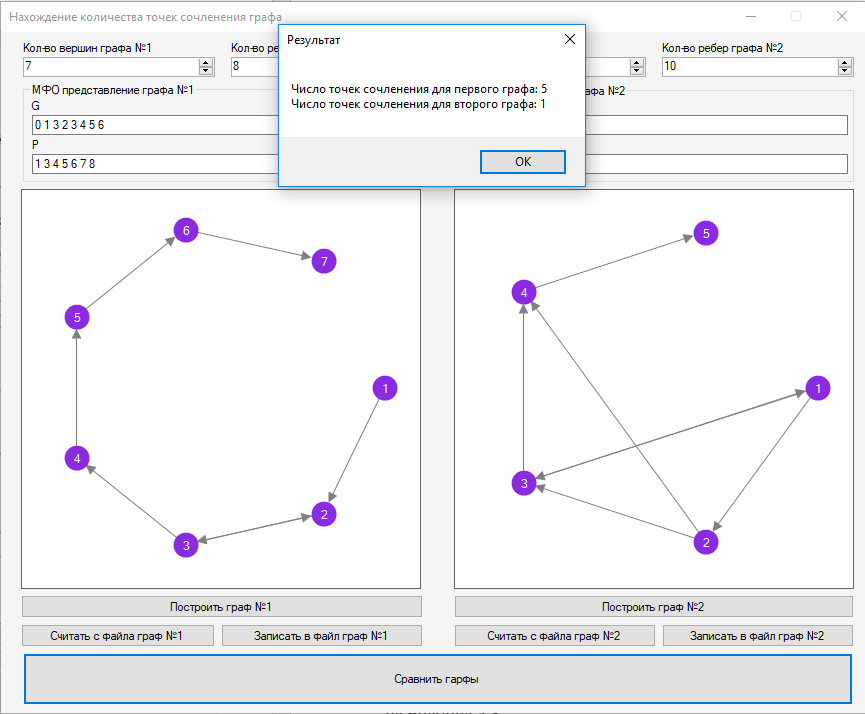


Рисунок 4.4 – Иллюстрация выполнения задачи 4

Задачу номер 5 реализует прецедент «Визуализация графов», результат представлен на рисунке 4.5

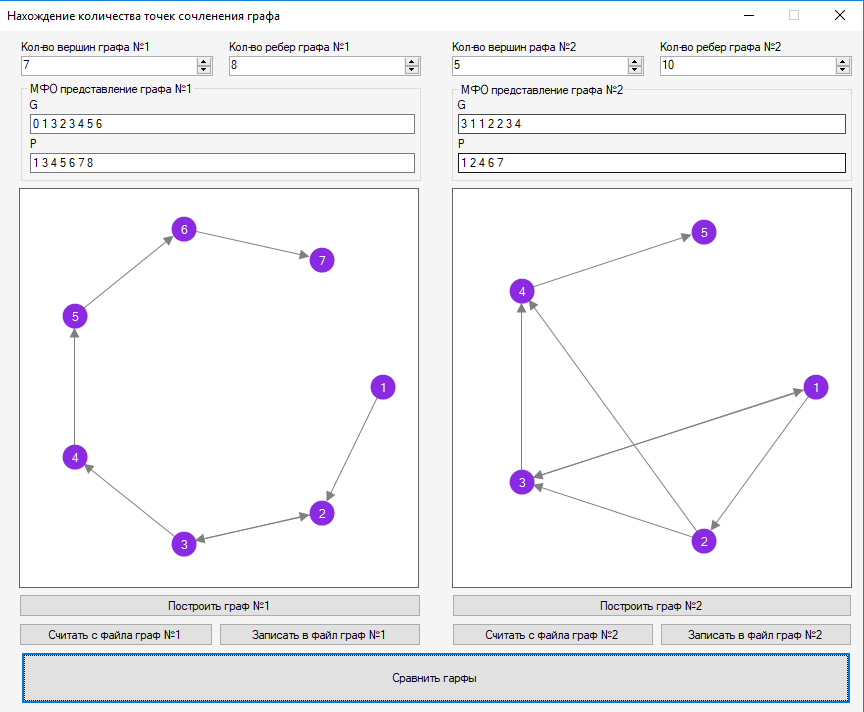


Рисунок 4.5 – Визуализация графов

## 4.2 Тестирование программы

Тестирование программного обеспечения (Software Testing) - проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом

Графы, созданные для тестирования изображены на рисунках 4.6 – 4.29:



Рисунок 4.6 – Первый граф

MFO-представление графа:

G { 0 1 2 3 4 2 7 5 6 }; P { 1 2 3 4 6 7 8 }; Ожидаемый результат тестирования: 3 вершин;



Рисунок 4.7 – Второй граф

MFO-представление графа:

G { 3 1 1 2 2 3 4 }; P { 1 2 4 6 7}; Ожидаемый результат тестирования: 1 вершина.



Рисунок 4.8 – Третий граф

MFO-представление графа:

G { 0 1 3 1 5 2 3 4}; P { 1 3 5 6 7 8}; Ожидаемые результаты тестирования: 3 точки.



Рисунок 4.9 – Четвертый граф

MFO-представление графа:

G { 0 1 2 5 2 3 1 2 }; P { 1 2 3 6 8}; Ожидаемые результаты тестирования: 0 точек;



Рисунок 4.10 – Пятый граф

MFO-представление графа:

G { 0 1 6 3 4 4 2}; P { 1 2 3 4 5 7}; Ожидаемый результат тестирования: 0 точек.



Рисунок 4.11 – Шестой граф

MFO-представление графа:

G { 4 1 1 2 3}; P { 1 2 4 5}; Ожидаемый результат тестирования: 3 точек.



Рисунок 4.12 – Седьмой граф

MFO-представление графа:

G { 0 1 5 1 1 1}; P { 1 3 4 5 6}; Ожидаемый результат тестирования: 1 точек.



Рисунок 4.13 – Восьмой граф

MFO-представление графа:

G { 2 4 2 1 3 }; P { 1 2 3 4 5 }; Ожидаемый результат тестирования: 2 точек.



Рисунок 4.14– Девятый граф

MFO-представление графа:

G { 0 1 4 2 3 4}; P { 1 3 4 5 6}; Ожидаемый результат тестирования: 2 точек.



Рисунок 4.15 – Десятый граф

MFO-представление графа:

G { 0 1 4 2 3 4}; P { 1 3 4 5 6}; Ожидаемый результат тестирования: 0 точек.



Рисунок 4.16 – Одиннадцатый граф

MFO-представление графа:

G { 8 1 1 2 2 8 7 5 2 4 3}; P {1 2 2 4 7 8 10 11}; Ожидаемый результат тестирования: 2 точек.



Рисунок 4.17 – Двенадцатый граф

MFO-представление графа:

G { 5 1 2 3 7 6 2 4 1 6 2 3 7}; P { 1 2 3 5 7 9 12 13}; Ожидаемый результат тестирования: 1 точек.

Тест 1. Результат теста по нахождению точек сочленения графов №1 и №2 представлен на рисунке 4.16 соответсвенно. Исходные графы представлены на рисунках 4.6 и 4.7

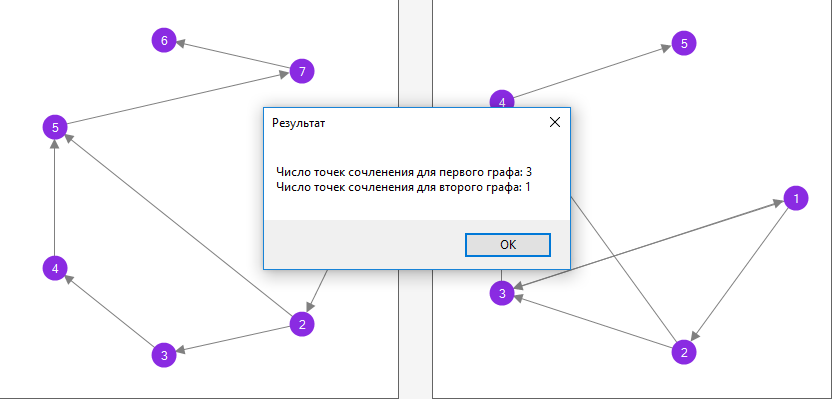


Рисунок 4.16 – Результат теста по нахождению точек сочленения графов №1 и №2.

Тест 2. Результат теста по нахождению точек сочленения графов №3 и №4 представлен на рисунке 4.17 соответсвенно. Исходные графы представлены на рисунках 4.8 и 4.9

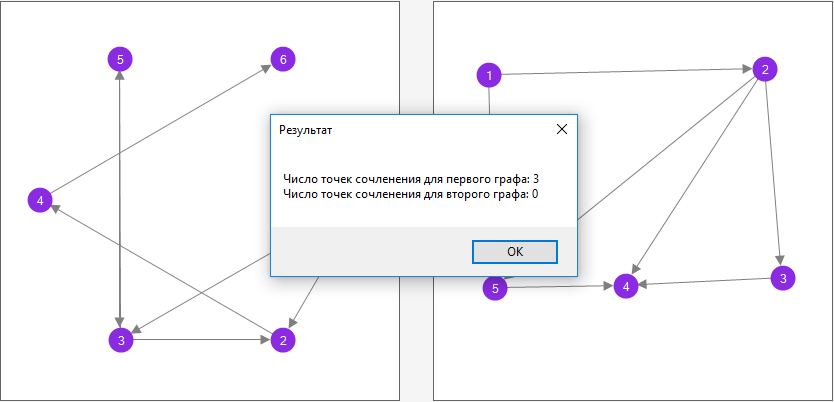


Рисунок 4.17 – Результат теста по нахождению точек сочленения графов №3 и №4.

Тест3. Результат теста по нахождению точек сочленения графов №5 и №6 представлен на рисунке 4.18 соответсвенно. Исходные графы представлены на рисунках 4.10 и 4.11

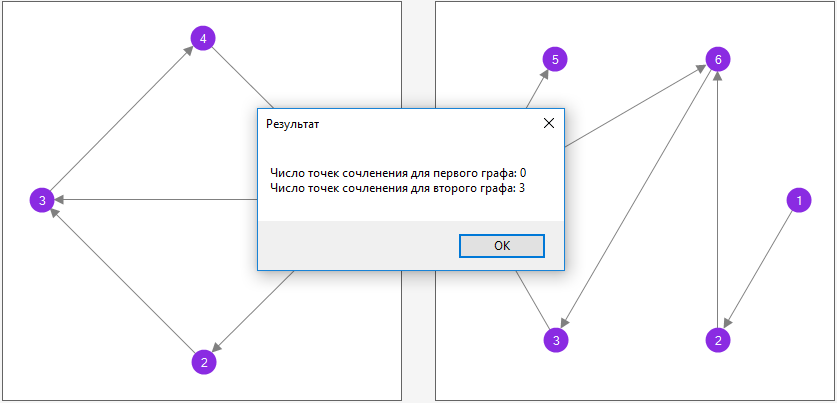


Рисунок 4.18 – Результат теста по нахождению точек сочленения графов №5 и №6.

Тест 4. Результат теста по нахождению точек сочленения графов №7 и №8 представлен на рисунке 4.19 соответсвенно. Исходные графы представлены на рисунках 4.12 и 4.13

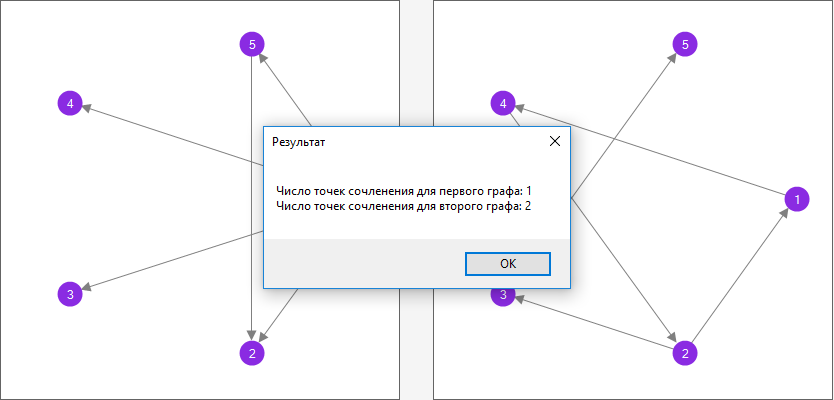


Рисунок 4.19 – Результат теста по нахождению точек сочленения графов №7 и №8.

Тест 5. Результат теста по нахождению точек сочленения графов №9 и №10 представлен на рисунке 4.20 соответсвенно. Исходные графы представлены на рисунках 4.14 и 4.15

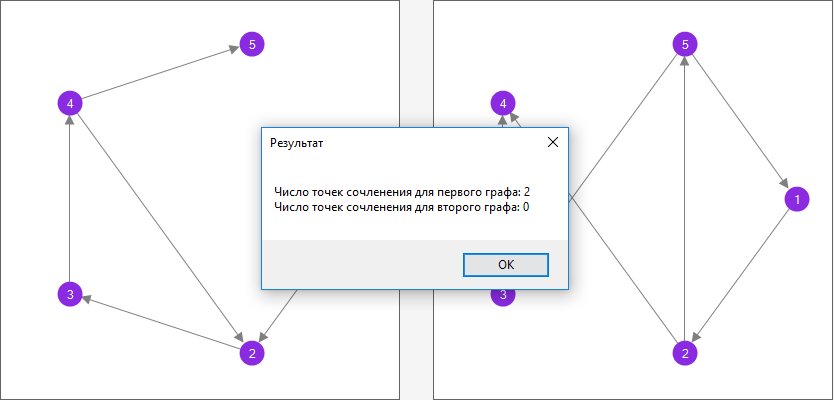


Рисунок 4.20 – Результат теста по нахождению точек сочленения графов №9 и №10.

Тест 5. Результат теста по нахождению точек сочленения графов №11 и №12 представлен на рисунке 4.20 соответсвенно. Исходные графы представлены на рисунках 4.16 и 4.17

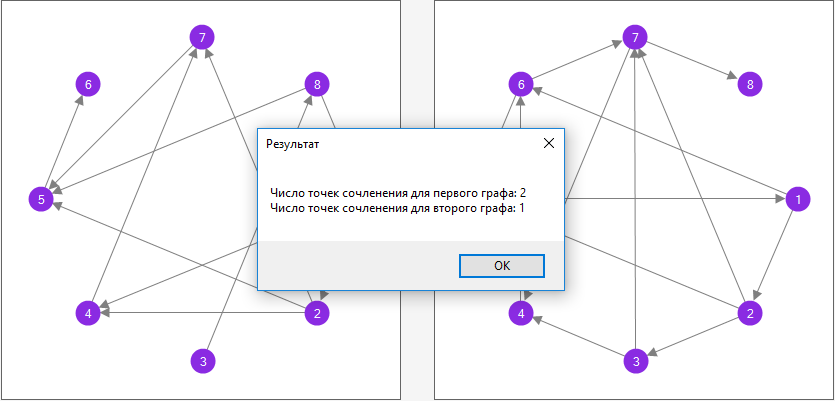


Рисунок 4.21 – Результат теста по нахождению точек сочленения графов №11 и №12.

Тестирование на поиск аномалий.

Тест 1. Результат некорректного ввода MFO массивов представлен на рисунке 4.21

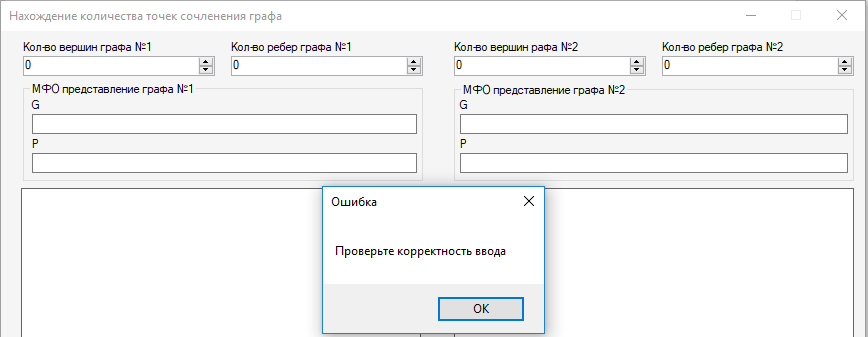


Рисунок 4.21 – Некорректный ввод графа

Тест 2. Результат некорректного ввода количества вершин или ребер представлен на рисунке 4.22

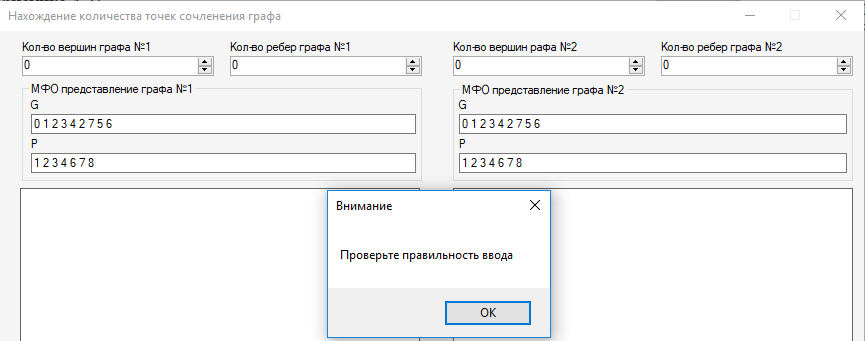


Рисунок 4.22 – Неполный ввод

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были изучены основные принципы работы с графами и способы их представления. Была проведена работа по нахождению числа точек сочленения графов. Программа верифицирована и протестирована. Тесты пройдены успешно.

В дальнейшем возможно усовершенствование программы:

1. Улучшение интерфейса;
2. Работа с графами большей размерности;
3. Улучшение и оптимизация алгоритма по нахождению числа точек сочленения графа;

# ПЕРЕЧЕНЬ ССЫЛОК

1. Текстовый файл [Электронный ресурс] / Режим доступа https://ru.wikipedia.org/wiki/Текстовый\_файл
2. Microsoft Word [Электронный ресурс] / Режим доступа https://ru.wikipedia.org/wiki/Microsoft\_Word
3. Уилсон Р. Введение в теорию графов. 1977. 207 с.
4. Оре О. Графы и их применение: Пер. с англ. 1965. 176 с.
5. Б.И. Иванов Дискретная Математика алгоритмы и программы
6. Мельников О.И. Теория графов в занимательных задачах. Изд.3, испр. и доп. 2009. 232 с.
7. Интернет источник: msdn.microsoft.com/hh\_144976.aspx
8. С# 2005 и платформа .NET 3.0 для профессионалов, Кристиан Нейгел, Киев: «Диалектика», 2008. – 1790 с.
9. Кулямин В.В. Методы верификации программного обеспечения. – М:РАН, 2013. – 12 с.

# ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

**1 Введение**

**1.1 Наименование программы**

Наименование программы – Нахождение количества точек сочленения графа.

**1.2 Краткая характеристика области применения**

Данная программа предназначена для исследования графов на наличие точек сочленения. Данный продукт может применяться как учебный материал для работы с графами.

**2 Основание для разработки**

**2.1 Основание для проведения разработки**

Основания для разработки – задание на курсовой проект по дисциплине «Объектно-ориентированное программирование» на тему «Нахождение числа точек сочленения графа.», выдано 24.01.2017.

**2.2 Наименование и условное обозначение темы разработки**

Наименование темы разработки – «Разработка программы для нахождения числа точек сочленения графа.».

**3 Назначение разработки**

**3.1 Функциональное назначение программы**

Данная программа предоставляет возможность нахождения числа точек сочленения графа. По введенным данным, необходимых для MFO-представления, производится нахождение числа точек сочленения графа.

**4** **Эксплуатационное назначение**

Программный продукт был разработан средствами Windows Forms в среде разработки Visual Studio 2017. Программа предназначена для работы на ОС Windows и не является кроссплатформенной, изменения в графическом отображении возможны.  
Программа не нуждается в облуживании. Программа распространяется как откомпилированный файл \*.exe. Сбой или отказ работы программы может привести к зависанию самой программы и ОС.

**5 Требования к программе или программному изделию**

**5.1 Требования к функциональным характеристикам**

**5.1.1 Требования к составу выполняемых функций**

Программа должна выполнять следующие функции:

а) осуществлять выбор ввода данных (с клавиатуры или из файла);

б) задавать количество ребер, вершин и MFO-представление;

в) проверять на корректность введенные данные;

г) выполнять поиск точек сочленения графа;

е) проверять сжатые графы на изоморфизм;

**5.1.2 Требования к организации входных данных**

Входные данные могут быть представлены как в виде текстовых файлов, так и могут быть введены с клавиатуры в виде массива MFO-представления. Текстовые файлы должны быть представлены в json-формате. Место расположения файлов: папка с исполняемой программой. Данные поступают в программу путем чтения из файлов или вводятся пользователем с помощью клавиатуры.

**5.1.3 Требования к организации выходных данных**

В зависимости от полученного результата поиска количества точек сочленения графов пользователю будет выдано сообщение о результатах.

**5.2 Требования к составу и параметрам технических средств**

В состав технических средств должен входить x86-совместимый компьютер, используемый в качестве рабочей станции и включающий в себя:

а) процессор с тактовой частотой, МГц - 400, не менее;

б) материнскую плату с FSB, МГц - 330, не менее;

в) оперативную память объемом, Мб -1, не менее;

г) свободное дисковое пространство для хранения программы, Мб – 1, не менее;

е) дисплей и видеоадаптер любого типа, позволяющий отображать информацию в текстовом режиме;

ж) клавиатуру;

з) мышь.

**5.3 Требования к информационной и программной совместимости**

**5.3.1 Требования к информационным структурам и методам решения**

Требования к информационным структурам на входе и выходе, а также к методам решения не предъявляются.

**5.3.2 Требования к исходным кодам и языкам программирования**

Для разработки программы в качестве языка программирования нужно использовать язык C# поддерживающий спецификацию компании производителя Microsoft, а за графический интерфейс отвечает Windows Forms. Для проектирования необходимо использовать среду разработки Microsoft Visual Studio версии 2010 и выше.

Исходные тексты программ должны содержать достаточное количество комментариев и быть доступными для любого читающего. Необходимо использовать осмысленные и корректные названия, объясняющие свое назначение.

**5.3.3 Требования к программным средствам, используемым программой**

Для функционирования программы на компьютере должно быть установлено следующее ПО:

* ОС Windows 8.1 и выше;
* платформа .NET Framework 4.5 и выше;
* Microsoft Visual Studio версии 2015 и выше.

**5.3.4 Требования к защите информации и программ**

Программа не имеет доступ к конфиденциальной информации, поэтому никакие требования к защите информации и программы не предъявляются.

**6 Требования к программной документации**

**6.1 Предварительный состав программной документации**

В результате разработки программы должна быть представлена следующая программная документация:

а) техническое задание;

б) схемы алгоритмов и данных;

в) текст программы;

г) результаты тестирования;

д) пояснительная записка;

е) руководство пользователя.

Кроме программного обеспечения на диске обязательно должна находиться пояснительная записка в электронном виде, включающая в себя весь комплект документации, предусмотренный в п. 5.1 настоящего технического задания, а также слайды презентации.

**7 Стадии и этапы разработки**

**7.1 Стадии разработки**

Разработка должны быть проведена в 4 этапа:

1. Проектирование.

2. Разработка пользовательского интерфейса.

3. Разработка проекта в целом.

4. Разработка руководства пользователя.

**7.2 Этапы разработки**

На этапе проектирования необходимо определить тему работы и разработать техническое задание. На этапе разработки пользовательского интерфейса необходимо разработать интерфейс программы. На этапе разработки проекта в целом необходимо разработать проект программы и провести ее тестирование. В заключительном этапе необходимо разработать руководство пользователя и подготовить программу к использованию.

# ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММЫ

MainForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace course\_project

{

/// <summary>

/// Класс формы

/// </summary>

public partial class MainForm : Form

{

/// <summary>

/// Ссылка на объект класса ModelingOfGraph - первый граф

/// </summary>

private ModelingOfGraph Graph1;

/// <summary>

/// Ссылка на объект класса ModelingOfGraph - второй граф

/// </summary>

private ModelingOfGraph Graph2;

/// <summary>

/// Ссылка на объект класса GraphBuilder - построить первый граф

/// </summary>

private GraphBuilder buildGraph1;

/// <summary>

/// Ссылка на объект класса GraphBuilder - построить второй граф

/// </summary>

private GraphBuilder buildGraph2;

/// <summary>

/// Ссылка на объект класса GraphReadWrite - чтение и запись данных

/// </summary>

private GraphReadWrite<ModelingOfGraph> graphReadWrite;

/// <summary>

/// Ссылка на объект класса ModifyData - проверка и модифицирование данных

/// </summary>

private DataModifiing modifyData;

/// <summary>

/// Конструктор по умолчанию

/// </summary>

public MainForm()

{

InitializeComponent();

Graph1 = new ModelingOfGraph();

Graph2 = new ModelingOfGraph();

buildGraph1 = new GraphBuilder(pictureBox\_graph\_1);

buildGraph2 = new GraphBuilder(pictureBox\_graph\_2);

graphReadWrite = new GraphReadWrite<ModelingOfGraph>();

modifyData = new DataModifiing();

}

/// <summary>

/// Обработчик для построения первого графа

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_build\_graph\_1\_Click(object sender, EventArgs e)

{

try

{

BuildGraph(Graph1, buildGraph1, (int)numericUpDown\_number\_of\_vertices\_1.Value, (int)numericUpDown\_number\_of\_edges\_1.Value, modifyData.StrToMasInt(textBox\_G\_1.Text), modifyData.StrToMasInt(textBox\_P\_1.Text));

}

catch (Exception n)

{

MessageBox.Show("Проверьте правильность ввода", "Внимание");

}

}

/// <summary>

/// Обработчик для построения второго графа

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_build\_graph\_2\_Click(object sender, EventArgs e)

{

try

{

BuildGraph(Graph2, buildGraph2, (int)numericUpDown\_number\_of\_vertices\_2.Value, (int)numericUpDown\_number\_of\_edges\_2.Value, modifyData.StrToMasInt(textBox\_G\_2.Text), modifyData.StrToMasInt(textBox\_P\_2.Text));

}

catch (Exception n)

{

MessageBox.Show("Проверьте правильность ввода", "Внимание");

}

}

/// <summary>

/// Обработчик для проверки данных в массиве MasEd

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void textBox\_G\_KeyPress(object sender, KeyPressEventArgs e)

{

// Empty

}

/// <summary>

/// Обработчик для проверки данных в массиве MasInd

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void textBox\_P\_KeyPress(object sender, KeyPressEventArgs e)

{

// Empty

}

/// <summary>

/// Обработчик для записи первого графа в файл

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_write\_graph\_1\_Click(object sender, EventArgs e)

{

WriteGraph(Graph1, (int)numericUpDown\_number\_of\_vertices\_1.Value, (int)numericUpDown\_number\_of\_edges\_1.Value, modifyData.StrToMasInt(textBox\_G\_1.Text), modifyData.StrToMasInt(textBox\_P\_1.Text), "first\_graph");

}

/// <summary>

/// Обработчик для записи второго графа в файл

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_write\_graph\_2\_Click(object sender, EventArgs e)

{

WriteGraph(Graph2, (int)numericUpDown\_number\_of\_vertices\_2.Value, (int)numericUpDown\_number\_of\_edges\_2.Value, modifyData.StrToMasInt(textBox\_G\_2.Text), modifyData.StrToMasInt(textBox\_P\_2.Text), "second\_graph");

}

/// <summary>

/// Обработчик для чтения первого графа из файла

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_read\_graph\_1\_Click(object sender, EventArgs e)

{

ReadGraph(Graph1, textBox\_G\_1, textBox\_P\_1, numericUpDown\_number\_of\_vertices\_1, numericUpDown\_number\_of\_edges\_1, "first\_graph");

}

/// <summary>

/// Обработчик для чтения второго графа из файла

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_read\_graph\_2\_Click(object sender, EventArgs e)

{

ReadGraph(Graph1, textBox\_G\_2, textBox\_P\_2, numericUpDown\_number\_of\_vertices\_2, numericUpDown\_number\_of\_edges\_2, "second\_graph");

}

/// <summary>

/// Обработчик для сравнения графов

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_compare\_Click(object sender, EventArgs e)

{

if (textBox\_G\_1.Text != "" && textBox\_G\_2.Text != "" && textBox\_P\_1.Text != "" && textBox\_P\_2.Text != "")

{

Comparing(Graph1, Graph2);

}

else

MessageBox.Show("Проверьте корректность ввода", "Ошибка");

}

/// <summary>

/// Вывести граф в поля редактирования

/// </summary>

/// <param name="graph"></param>

/// <param name="MasEd"></param>

/// <param name="MasInd"></param>

/// <param name="NumVertic"></param>

/// <param name="NumEdge"></param>

private void PrintGraph(ModelingOfGraph graph, TextBox MasEd, TextBox MasInd, NumericUpDown NumVertic, NumericUpDown NumEdge)

{

if (graph.MasEd != null && graph.MasInd != null)

{

MasEd.Text = graph.MasEd[0].ToString();

MasInd.Text = graph.MasInd[0].ToString();

for (int i = 1; i < graph.MasEd.Length; i++)

{

MasEd.Text += " " + graph.MasEd[i].ToString();

}

for (int i = 1; i < graph.MasInd.Length; i++)

{

MasInd.Text += " " + graph.MasInd[i].ToString();

}

NumVertic.Value = graph.NumVertic;

NumEdge.Value = graph.NumEdge;

}

}

/// <summary>

/// Записать граф в файл

/// </summary>

/// <param name="graph"></param>

/// <param name="NumVertic"></param>

/// <param name="NumEdge"></param>

/// <param name="MasEd"></param>

/// <param name="MasInd"></param>

/// <param name="FileName"></param>

private void WriteGraph(ModelingOfGraph graph, int NumVertic, int NumEdge, int[] MasEd, int[] MasInd, string FileName)

{

saveFileDialog\_write\_graph.FileName = FileName;

saveFileDialog\_write\_graph.Filter = "JSON files (\*.json)|\*.json";

if (saveFileDialog\_write\_graph.ShowDialog() == DialogResult.OK)

{

graph.NumVertic = NumVertic;

graph.NumEdge = NumEdge;

graph.MasEd = MasEd;

graph.MasInd = MasInd;

graphReadWrite.WriteGraph(graph, saveFileDialog\_write\_graph.FileName);

}

}

/// <summary>

/// Прочитать граф из файла

/// </summary>

/// <param name="graph"></param>

/// <param name="MasEd"></param>

/// <param name="MasInd"></param>

/// <param name="NumVertic"></param>

/// <param name="NumEdge"></param>

/// <param name="FileName"></param>

private void ReadGraph(ModelingOfGraph graph, TextBox MasEd, TextBox MasInd, NumericUpDown NumVertic, NumericUpDown NumEdge, string FileName)

{

openFileDialog\_read\_graph.FileName = FileName;

openFileDialog\_read\_graph.Filter = "JSON files (\*.json)|\*.json";

if (openFileDialog\_read\_graph.ShowDialog() == DialogResult.OK)

{

graph = graphReadWrite.ReadGraph(openFileDialog\_read\_graph.FileName);

PrintGraph(graph, MasEd, MasInd, NumVertic, NumEdge);

}

}

/// <summary>

/// Построить граф

/// </summary>

/// <param name="graph"></param>

/// <param name="buildGraph"></param>

/// <param name="NumVertic"></param>

/// <param name="NumEdge"></param>

/// <param name="MasEd"></param>

/// <param name="MasInd"></param>

private void BuildGraph(ModelingOfGraph graph, GraphBuilder buildGraph, int NumVertic, int NumEdge, int[] MasEd, int[] MasInd)

{

graph.NumVertic = NumVertic;

graph.NumEdge = NumEdge;

graph.MasEd = MasEd;

graph.MasInd = MasInd;

graph.CreateAdjacencyMatrix();

buildGraph.DrawingGraph(graph.AdjacencyMatrix);

}

/// <summary>

/// Сравнить графы

/// </summary>

/// <param name="Graph1"></param>

/// <param name="Graph2"></param>

private void Comparing(ModelingOfGraph Graph1, ModelingOfGraph Graph2)

{

int result = Graph1.ComparingGraphs(Graph2);

if (result == 1)

{

MessageBox.Show("Графы эквивалентны!", "Сообщение");

}

else if (result == 0)

{

MessageBox.Show("Графы не эвивалентны!", "Сообщение");

}

}

}

}

DataModifing.cs

namespace course\_project\_oop\_02

{

/// <summary>

/// Класс для преобразования и проверки данных

/// </summary>

class DataModifiing

{

/// <summary>

/// Преобразовать строку в целочисленный массив, где числа в строке разделены пробелами

/// </summary>

/// <param name="s"></param>

/// <returns>Массив целых чисел</returns>

public int[] StrToMasInt(string s)

{

if (s != null && s != "")

{

string data = s;

string[] S = data.Split(' ');

int[] mas = new int[S.Length];

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

{

mas[i] = Convert.ToInt32(S[i]);

}

return mas;

}

else

{

return null;

}

}

}

}

GraphBuilder.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace course\_project

{

/// <summary>

/// Класс для визуализации графа

/// </summary>

class GraphBuilder

{

/// <summary>

/// Массив точек для хранения координат вершин графа

/// </summary>

private PointF[] pointsVerticles;

/// <summary>

/// Переменная для хранения смещения вершины

/// </summary>

private Point smesh;

/// <summary>

/// Область для рисования графа

/// </summary>

private PictureBox reg;

/// <summary>

/// Ссылка на объект класса Graphics - предоставление средств для рисования

/// </summary>

private Graphics graphica;

/// <summary>

/// Ссылка на объект класса Bitmap - хранение изображения полного графа

/// </summary>

private Bitmap bmp;

/// <summary>

/// Ссылка на объект класса Bitmap - хранения изображения не полного графа(только тех частей, которые являются неподвижными во время передвижения вершины графа)

/// </summary>

private Bitmap backbmp;

/// <summary>

/// Карандаш для рисования ребер, которые не двигаются

/// </summary>

private Pen GrayPen;

/// <summary>

/// Карандаш для рисования ребер, которые двигаются

/// </summary>

private Pen BlackPen;

/// <summary>

/// Шрифт для подписи номеров вершин

/// </summary>

private Font Font\_;

/// <summary>

/// Цвет для рисования подвижных ребер

/// </summary>

private Color ColorEdge;

/// <summary>

/// Цвет для рисования вершин

/// </summary>

private Color ColorVerticles;

/// <summary>

/// Индекс подвижной вершины

/// </summary>

private int UsingVerticle;

/// <summary>

/// Переменная для хранения результата попадания точки в область - true, если есть попадание

/// </summary>

private bool Clicked;

/// <summary>

/// Количество вершин

/// </summary>

private int Top;

/// <summary>

/// Матрица смежности

/// </summary>

private bool[][] \_adjacencyMatrix;

/// <summary>

/// Конструктор с параметрами - задание области для рисования

/// </summary>

/// <param name="region">Область для рисования</param>

public GraphBuilder(PictureBox region1)

{

this.reg = region1;

this.bmp = new Bitmap(this.reg.Width, this.reg.Height);

this.backbmp = new Bitmap(this.reg.Width, this.reg.Height);

this.graphica = Graphics.FromImage(bmp);

this.graphica.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

this.reg.MouseMove += pictureBox\_MouseMove;

this.reg.MouseDown += pictureBox\_MouseDown;

this.reg.MouseUp += pictureBox\_MouseUp;

this.ColorEdge = Color.Black;

this.BlackPen = new Pen(ColorEdge, 1f) { CustomEndCap = new AdjustableArrowCap(5, 5) };

this.GrayPen = new Pen(Color.Gray, 1f) { CustomEndCap = new AdjustableArrowCap(5, 5) };

this.Font\_ = new Font("TimesNewRoman", 9f);

}

/// <summary>

/// Нарисовать граф

/// </summary>

/// <param name="AdjacencyMatrix">Матрица смежности</param>

public void DrawingGraph(bool[][] AdjacencyMatrix)

{

if (AdjacencyMatrix != null)

{

this.Top = AdjacencyMatrix.Length;

this.\_adjacencyMatrix = AdjacencyMatrix;

graphica.Clear(reg.BackColor);

CreatePoints();

DrawEdge();

DrawVerticles();

reg.Image = bmp;

}

}

/// <summary>

/// Создать массив координат вершин

/// </summary>

private void CreatePoints()

{

pointsVerticles = new PointF[Top];

double degr = 360.0 / Top;

double d = 0;

Point xy = new Point(reg.Width / 2 - 12, reg.Height / 2 - 15);

Point x0y0 = new Point(reg.Width - 50, reg.Height / 2 - 15);

Point rxry = new Point((x0y0.X - xy.X), (x0y0.Y - xy.Y));

for (int i = 0; i < Top; i++, d += degr)

{

double cosinus = Math.Cos((Math.PI / 180) \* d);

double sinus = Math.Sin((Math.PI / 180) \* d);

double tmps = (xy.X + rxry.X \* cosinus - rxry.Y \* sinus);

double tmp1 = (xy.Y + rxry.X \* sinus + rxry.Y \* cosinus);

Point x1y1 = new Point((int)tmps, (int)tmp1);

pointsVerticles[i] = new Point(x1y1.X, x1y1.Y);

}

}

/// <summary>

/// Нарисовать ребра

/// </summary>

private void DrawEdge()

{

if (pointsVerticles != null && \_adjacencyMatrix != null)

{

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

{

for (int j = 0; j < \_adjacencyMatrix[i].Length; j++)

{

if (!(Clicked && (UsingVerticle == i || UsingVerticle == j)))

{

if (i == j && \_adjacencyMatrix[i][j])

{

graphica.DrawEllipse(Pens.Gray, pointsVerticles[i].X - 3, pointsVerticles[i].Y - 18, 30, 30);

}

else if (\_adjacencyMatrix[i][j])

{

double Xc = 0;

double Yc = 0;

GetPointsEdge(ref Xc, ref Yc, pointsVerticles[j], pointsVerticles[i]);

graphica.DrawLine(GrayPen, pointsVerticles[i].X + 12, pointsVerticles[i].Y + 12, (int)Xc + 12, (int)Yc + 12);

}

}

}

}

}

}

/// <summary>

/// Нарисовать вершины

/// </summary>

private void DrawVerticles()

{

if (pointsVerticles != null && \_adjacencyMatrix != null)

{

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

{

ColorVerticles = Clicked && i == UsingVerticle ? Color.LightGreen : Color.BlueViolet;

graphica.FillEllipse(new SolidBrush(ColorVerticles), pointsVerticles[i].X, pointsVerticles[i].Y, 25, 25);

if (i < 9)

{

graphica.DrawString((i + 1).ToString(), Font\_, Brushes.GhostWhite, pointsVerticles[i].X + 8, pointsVerticles[i].Y + 5);

}

else

{

graphica.DrawString((i + 1).ToString(), Font\_, Brushes.GhostWhite, pointsVerticles[i].X + 4, pointsVerticles[i].Y + 5);

}

}

}

}

/// <summary>

/// Нарисовать подвижные ребра(те, которые связаны с подвижной вершиной)

/// </summary>

private void DrawEdgesForMovingVerticle()

{

for (int i = 0, index = UsingVerticle; i < \_adjacencyMatrix.Length; i++)

{

Pen p = new Pen(ColorEdge);

if (i == index && \_adjacencyMatrix[i][index])

{

graphica.DrawEllipse(p, pointsVerticles[i].X - 3, pointsVerticles[i].Y - 18, 30, 30);

}

else if (\_adjacencyMatrix[i][index] || \_adjacencyMatrix[index][i])

{

double Xc = 0;

double Yc = 0;

if (\_adjacencyMatrix[i][index])

{

GetPointsEdge(ref Xc, ref Yc, pointsVerticles[index], pointsVerticles[i]);

graphica.DrawLine(BlackPen, pointsVerticles[i].X + 12, pointsVerticles[i].Y + 12, (int)Xc + 12, (int)Yc + 12);

}

if (\_adjacencyMatrix[index][i])

{

GetPointsEdge(ref Xc, ref Yc, pointsVerticles[i], pointsVerticles[index]);

graphica.DrawLine(BlackPen, pointsVerticles[index].X + 12, pointsVerticles[index].Y + 12, (int)Xc + 12, (int)Yc + 12);

}

}

}

}

/// <summary>

/// Получить координаты точки для построения ребра

/// </summary>

/// <param name="Xc">X</param>

/// <param name="Yc">Y</param>

/// <param name="firstPoint">Первая точка</param>

/// <param name="secondPoint">Вторая точка</param>

private void GetPointsEdge(ref double Xc, ref double Yc, PointF firstPoint, PointF secondPoint)

{

double Rab = Math.Sqrt((firstPoint.X - secondPoint.X) \* (firstPoint.X - secondPoint.X) +

(firstPoint.Y - secondPoint.Y) \* (firstPoint.Y - secondPoint.Y));

double k = Rab == 0 ? 0 : (Rab - 12) / Rab;

Xc = secondPoint.X + (firstPoint.X - secondPoint.X) \* k;

Yc = secondPoint.Y + (firstPoint.Y - secondPoint.Y) \* k;

}

/// <summary>

/// Обработчик для проверки попадания точки в одну из вершин

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox\_MouseDown(object sender, MouseEventArgs e)

{

if (pointsVerticles != null)

{

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

{

PointF ab = new PointF(25 / 2 + pointsVerticles[i].X, 25 / 2 + pointsVerticles[i].Y);

PointF xy = new PointF(e.X, e.Y);

int R = 25 / 2;

if (((ab.X - xy.X) \* (ab.X - xy.X) + (ab.Y - xy.Y) \* (ab.Y - xy.Y)) < R \* R)

{

Clicked = true;

smesh.X = (int)(e.X - pointsVerticles[i].X);

smesh.Y = (int)(e.Y - pointsVerticles[i].Y);

UsingVerticle = i;

graphica.Clear(reg.BackColor);

DrawEdge();

backbmp = bmp.Clone(new Rectangle(0,0,reg.Width,reg.Height), bmp.PixelFormat);

}

}

}

}

/// <summary>

/// Обработчик для перерисовки графа после отпускания клавиши мыши

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox\_MouseUp(object sender, MouseEventArgs e)

{

graphica.Clear(reg.BackColor);

Clicked = false;

DrawEdge();

DrawVerticles();

reg.Image = bmp;

}

/// <summary>

/// Обработчик для рисования графа во время передвижения вершины

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

if (Clicked && pointsVerticles != null)

{

if ((e.X - smesh.X) < reg.Width - 25 && (e.X - smesh.X) >= 0)

{

pointsVerticles[UsingVerticle].X = e.X - smesh.X;

}

if ((e.Y - smesh.Y) < reg.Height - 25 && (e.Y - smesh.Y) >= 0)

{

pointsVerticles[UsingVerticle].Y = e.Y - smesh.Y;

}

graphica.Clear(reg.BackColor);

graphica.DrawImage(backbmp, 0, 0, reg.Width, reg.Height);

DrawEdgesForMovingVerticle();

DrawVerticles();

reg.Image = bmp;

}

}

}

}

GraphReadingAndWriting.cs

namespace course\_project

{

/// <summary>

/// Класс для чтения и записи данных

/// </summary>

/// <typeparam name="Graph"></typeparam>

class GraphReadWrite<Graph>: IReadWriteGraph<Graph>

{

/// <summary>

/// Записать данные в файл

/// </summary>

/// <param name="graph">Обобщенный тип, указанный при создании интерфейсной ссылки</param>

/// <param name="path">Путь к файлу</param>

public void WriteGraph(Graph graph, string path)

{

using (StreamWriter swp = new StreamWriter(path))

{

JavaScriptSerializer serial = new JavaScriptSerializer();

swp.WriteLine(serial.Serialize(graph));

}

}

/// <summary>

/// Прочитать данные из файла

/// </summary>

/// <param name="path">Путь к файлу</param>

/// <returns>Обобщенный тип, указанный при создании интерфейсной ссылки</returns>

public Graph ReadGraph(string path)

{

using (StreamReader srq = new StreamReader(path))

{

JavaScriptSerializer serial = new JavaScriptSerializer();

Graph tGraph = serial.Deserialize<Graph>(srq.ReadLine());

return tGraph;

}

}

}

}

IReadingAndWritingGraph.cs

namespace course\_project

{

/// <summary>

/// Обобщенный интерфейс для чтения и записи данных в формате JSON

/// </summary>

/// <typeparam name="Graph"></typeparam>

interface IReadWriteGraph<Graph>

{

/// <summary>

/// Прочитать данные из файла

/// </summary>

/// <param name="path">Путь к файлу</param>

/// <returns>Обобщенный тип, указанный при создании интерфейсной ссылки</returns>

Graph ReadGraph(string path);

/// <summary>

/// Записать данные в файл

/// </summary>

/// <param name="graph">Обобщенный тип, указанный при создании интерфейсной ссылки</param>

/// <param name="path">Путь к файлу</param>

void WriteGraph(Graph graph, string path);

}

}

ModelingOfGraph.cs

namespace course\_project

{

/// <summary>

/// Класс для решения задачи нахождения числа точек сочленения в графе

/// </summary>

[Serializable]

class ModelingOfGraph

{

/// <summary>

/// Количество вершин

/// </summary>

public int NumVertic { get; set; }

/// <summary>

/// Количество ребер

/// </summary>

public int NumEdge { get; set; }

/// <summary>

/// Массив MasEd - массив ребер

/// </summary>

public int[] MasEd { get; set; }

/// <summary>

/// Массив MasInd - массив индексов для обращения в первый массив

/// </summary>

public int[] MasInd { get; set; }

/// <summary>

/// Матрица смежности

/// </summary>

[ScriptIgnore]

public bool[][] AdjacencyMatrix { get; private set; }

/// <summary>

/// Конструктор по умолчанию

/// </summary>

public ModelingOfGraph()

{

}

/// <summary>

/// Конструктор с параметрами - задание количества вершин NumVertic, к-ва ребер NumEdge и MFO

/// </summary>

/// <param name="NumVertic">Количество вершин</param>

/// <param name="NumEdge">Количество ребер</param>

/// <param name="MasEd">Массив ребер</param>

/// <param name="MasInd">Массив индексов</param>

public ModelingOfGraph(int NumVertic, int NumEdge, int[] MasEd, int[] MasInd)

{

this.NumVertic = NumVertic;

this.NumEdge = NumEdge;

this.MasEd = MasEd;

this.MasInd = MasInd;

}

/// <summary>

/// Сравнить графы на эквивалентность

/// </summary>

/// <param name="secondGraph">Граф для сравнения</param>

/// <returns>1 - если графы эквивалентны, 0 - если не эквивалентны, -1 - ошибка</returns>

public int ComparingGraphs(ModelingOfGraph secondGraph)

{

if (AdjacencyMatrix != null && secondGraph.AdjacencyMatrix != null)

{

int numberOfJunctionPointsForFirstGraph = FindJunctionPoints(this.AdjacencyMatrix);

int numberOfJunctionPointsForSecondGraph = FindJunctionPoints(secondGraph.AdjacencyMatrix);

MessageBox.Show($"Число точек сочленения для первого графа: {numberOfJunctionPointsForFirstGraph}\nЧисло точек сочленения для второго графа: {numberOfJunctionPointsForSecondGraph}", "Результат");

if (numberOfJunctionPointsForFirstGraph == numberOfJunctionPointsForSecondGraph)

{

return 1;

}

else

{

return 0;

}

}

else

{

return -1;

}

}

/// <summary>

/// Создать матрицу смежности из полученного MFO

/// </summary>

public void CreateAdjacencyMatrix()

{

if (MasInd != null)

{

AdjacencyMatrix = new bool[NumVertic][];

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

{

AdjacencyMatrix[i] = new bool[NumVertic];

}

int index = 0;

int prev\_index = 0;

int current\_index = 0;

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

{

current\_index = Convert.ToInt32(MasInd[i]);

index = current\_index - prev\_index;

for (int j = index, k = current\_index; j > 0; j--, k--)

{

int l = Convert.ToInt32(MasEd[k - 1]) - 1;

if (l >= 0)

{

AdjacencyMatrix[l][i] = true;

}

}

prev\_index = current\_index;

}

}

}

/// <summary>

/// Найти точки сочленения

/// </summary>

/// <param name="adjacencyMatrix">Матрица смежности</param>

/// <returns>Количество точек сочленения</returns>

private int FindJunctionPoints(bool[][] adjacencyMatrix)

{

int numberOfJunctionPoints = 0;

bool[][] copyOfAdjacencyMatrix = CloneAdjacencyMatrix(adjacencyMatrix);

ChangeAdjacencyMatrix(ref copyOfAdjacencyMatrix);

CheckAdjacencyMatrix(copyOfAdjacencyMatrix);

if (!CheckAdjacencyMatrix(copyOfAdjacencyMatrix))

{

return 0;

}

for (int k = 0; k < adjacencyMatrix.Length; k++)

{

copyOfAdjacencyMatrix = CloneAdjacencyMatrix(adjacencyMatrix);

ChangeAdjacencyMatrix(ref copyOfAdjacencyMatrix);

List<int> adjacentVertices = FindAdjacentVertices(k, copyOfAdjacencyMatrix);

int index = adjacentVertices.IndexOf(k);

if (index != -1)

{

adjacentVertices.Remove(index);

}

RemoveVerticleFromAdjacencyMatrix(k, ref copyOfAdjacencyMatrix);

bool flag = false;

for (int i = 0; i < adjacentVertices.Count; i++)

{

for (int j = 0; j < copyOfAdjacencyMatrix.Length; j++)

{

if (j != k && adjacentVertices[i] != j && FindLength(adjacentVertices[i], j, copyOfAdjacencyMatrix) == 0)

{

numberOfJunctionPoints++;

flag = true;

break;

}

}

if (flag)

{

break;

}

}

}

return numberOfJunctionPoints;

}

/// <summary>

/// Проверить матрицу на первом шаге - если граф не связный в самом начале, то точки сочленения больше искать не нужно

/// </summary>

/// <param name="adjacencyMatrix">Матрица смежности</param>

/// <returns>true - если граф связный, false - если не связный</returns>

private bool CheckAdjacencyMatrix(bool[][] adjacencyMatrix)

{

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

{

for (int j = 0; j < adjacencyMatrix.Length; j++)

{

int result = (int)FindLength(i, j, adjacencyMatrix);

if (i != j && result == 0)

{

return false;

}

}

}

return true;

}

/// <summary>

/// Дополнить матрицу связями для корректной работы поиска пути между вершинами

/// </summary>

/// <param name="adjacencyMatrix"></param>

private void ChangeAdjacencyMatrix(ref bool[][] adjacencyMatrix)

{

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

{

for (int j = 0; j < adjacencyMatrix.Length; j++)

{

if (adjacencyMatrix[i][j] || adjacencyMatrix[j][i])

{

adjacencyMatrix[j][i] = true;

adjacencyMatrix[i][j] = true;

}

if (i == j)

{

adjacencyMatrix[i][j] = false;

}

}

}

}

/// <summary>

/// Клонировать матрицу смежности

/// </summary>

/// <param name="sourceAdjacencyMatrix">Исходная матрица</param>

/// <returns>Результирующая матрица</returns>

private bool[][] CloneAdjacencyMatrix(bool[][] sourceAdjacencyMatrix)

{

if (sourceAdjacencyMatrix != null)

{

bool[][] destinationAdjacencyMatrix = new bool[sourceAdjacencyMatrix.Length][];

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

{

destinationAdjacencyMatrix[i] = new bool[sourceAdjacencyMatrix.Length];

Array.Copy(sourceAdjacencyMatrix[i], destinationAdjacencyMatrix[i], sourceAdjacencyMatrix.Length);

}

return destinationAdjacencyMatrix;

}

else

{

return null;

}

}

/// <summary>

/// Удалить вершину из матрицы смежности

/// </summary>

/// <param name="numberOfVerticle">Номер вершины</param>

/// <param name="adjacencyMatrix">Матрица смежности</param>

private void RemoveVerticleFromAdjacencyMatrix(int numberOfVerticle, ref bool[][] adjacencyMatrix)

{

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

{

adjacencyMatrix[numberOfVerticle][i] = false;

adjacencyMatrix[i][numberOfVerticle] = false;

}

}

/// <summary>

/// Найти смежные вершины для заданной вершины

/// </summary>

/// <param name="numberOfVerticle">Номер вершины</param>

/// <param name="adjacencyMatrix">Матрица смежности</param>

/// <returns>Обобщенная коллекция List<int> - номера смежных вершин; null - если номер вершины является не действительным</returns>

private List<int> FindAdjacentVertices(int numberOfVerticle, bool[][] adjacencyMatrix)

{

if (numberOfVerticle >= 0 && numberOfVerticle < adjacencyMatrix.Length)

{

List<int> adjacentVertices = new List<int>();

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

{

if (adjacencyMatrix[numberOfVerticle][i])

{

adjacentVertices.Add(i);

}

if (adjacencyMatrix[i][numberOfVerticle])

{

adjacentVertices.Add(i);

}

}

return adjacentVertices.Distinct().ToList();

}

else

{

return null;

}

}

/// <summary>

/// Найти длину пути из начальной точки в конечную

/// </summary>

/// <param name="startPoint">Начальная точка</param>

/// <param name="endPoint">Конечная точка</param>

/// <param name="adjacencyMatrix">Матрица смежности</param>

/// <returns>Длина пути</returns>

private uint FindLength(int startPoint, int endPoint, bool[][] adjacencyMatrix)

{

int i, p, xn, xk;

bool[] flag = new bool[adjacencyMatrix.Length];

uint[][] c = new uint[adjacencyMatrix.Length][];

uint[] l = new uint[adjacencyMatrix.Length];

for (int k = 0; k < c.Length; k++)

{

c[k] = new uint[adjacencyMatrix.Length];

}

for (int k = 0; k < c.Length; k++)

{

for (int t = 0; t < c.Length; t++)

{

if (adjacencyMatrix[k][t])

{

c[k][t] = 1;

}

if (c[k][t] == 0)

{

c[k][t] = 65535;

}

}

}

xn = startPoint;

xk = endPoint;

if (xn == xk)

{

return 0;

}

for (i = 0; i < adjacencyMatrix.Length; i++)

{

l[i] = 65535;

}

l[xn] = 0;

flag[xn] = true;

p = xn;

do

{

for (i = 0; i < adjacencyMatrix.Length; i++)

{

if ((c[p][i] != 65535) && (!flag[i]) && (i != p))

{

l[i] = minim(l[i], l[p] + c[p][i]);

}

}

p = min(adjacencyMatrix.Length, flag, l);

flag[p] = true;

}

while (p != xk);

if (l[p] != 65535)

{

return l[p];

}

else

{

return 0;

}

}

/// <summary>

/// Метод для поддержки работы метода FindLength

/// </summary>

/// <param name="verticles"></param>

/// <param name="flag"></param>

/// <param name="l"></param>

/// <returns></returns>

private int min(int verticles, bool[] flag, uint[] l)

{

int i = 0;

int result = 0;

for (i = 0; i < verticles; i++)

{

if (!(flag[i]))

{

result = i;

}

}

for (i = 0; i < verticles; i++)

{

if ((l[result] > l[i]) && (!flag[i]))

{

result = i;

}

}

return result;

}

/// <summary>

/// Метод для поддержки работы метода FindLength

/// </summary>

private uint minimum(uint x, uint y)

{

if (x < y)

{

return x;

}

return y;

}

}

}

# ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. Назначение программы

Программа представляет собой удобный и интуитивно понятный пользовательский интерфейс для нахождения числа точек сочленения графа.

1. Системные требования

В состав технических средств должен входить x86-совместимый компьютер, используемый в качестве рабочей станции и включающий в себя:

а) процессор с тактовой частотой, МГц - 400, не менее;

б) материнскую плату с FSB, МГц - 330, не менее;

в) оперативную память объемом, Мб -1, не менее;

г) свободное дисковое пространство для хранения программы, Мб – 1, не менее;

е) дисплей и видеоадаптер любого типа, позволяющий отображать информацию в текстовом режиме;

ж) клавиатуру;

з) мышь.

1. Выполнение программы
2. Запустите приложение «course\_project\_oop\_02.exe».

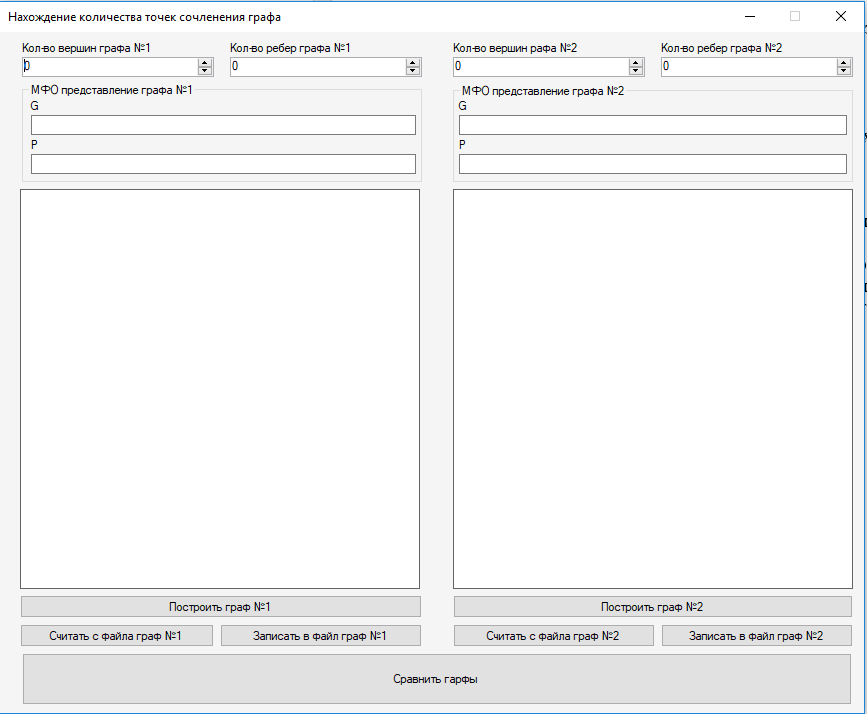


Рисунок Р.П.1 – Внешний вид приложения.

1. Для ввода MFO представления графа в соответствующие поля G и P воспользуйтесь кнопкой «Считать с файла граф №1»/ «Считать с файла граф №2» (файлы должны соответствовать формату json) либо же воспользуйтесь ручным вводом с клавиатуры.

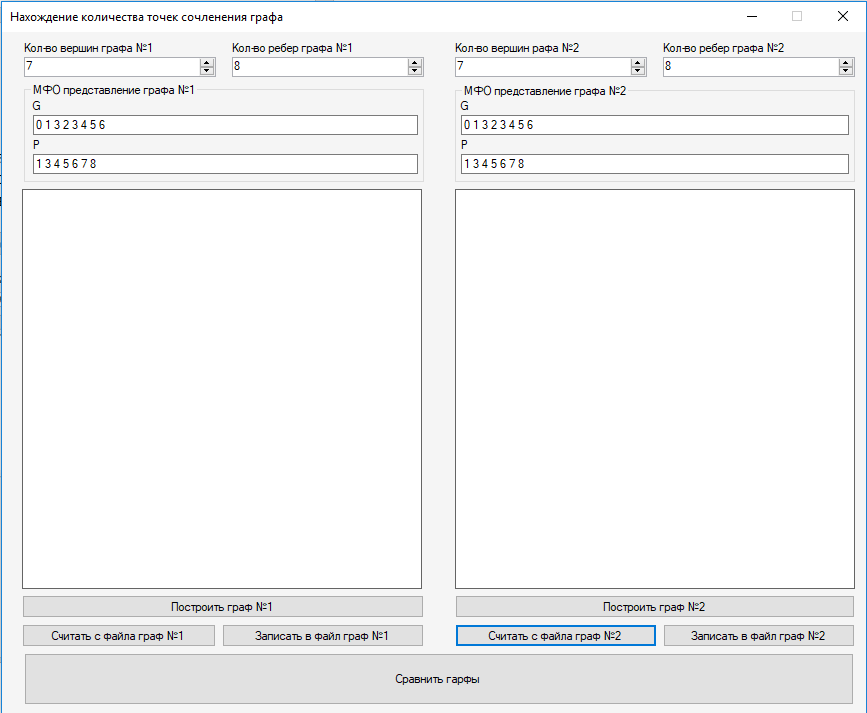


Рисунок Р.П.2 – Ввод MFO представления графа

1. При нажатии кнопки «Построить граф №1(2)» на экран будет выведен рисунок с графом.

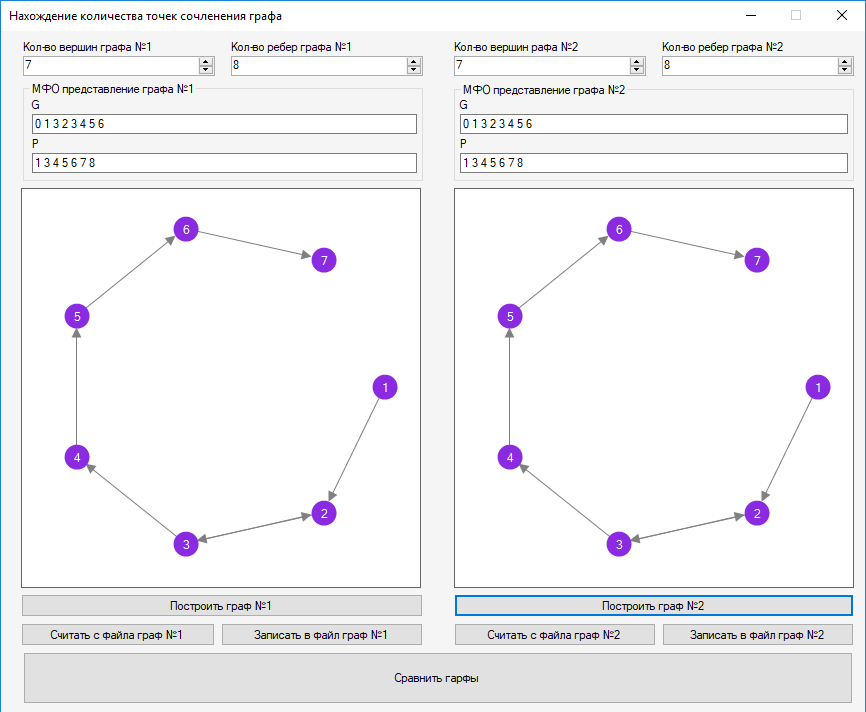


Рисунок Р.П.3 – Вывод картинки с графом после нажатия на кнопку «Построить граф №1(2)»

1. Нажмите на кнопку «Сравнить графы» для вывода на экран число точек сочленения графа и результаты проверки на эквивалентность.

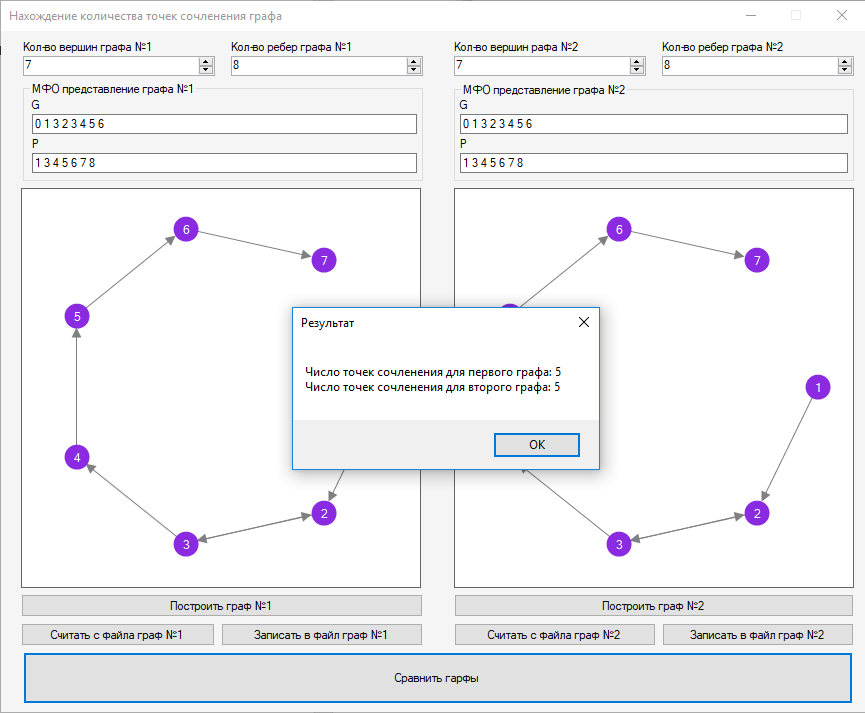


Рисунок Р.П.4 – Вывод результата по поиску числа точек сочленения графа

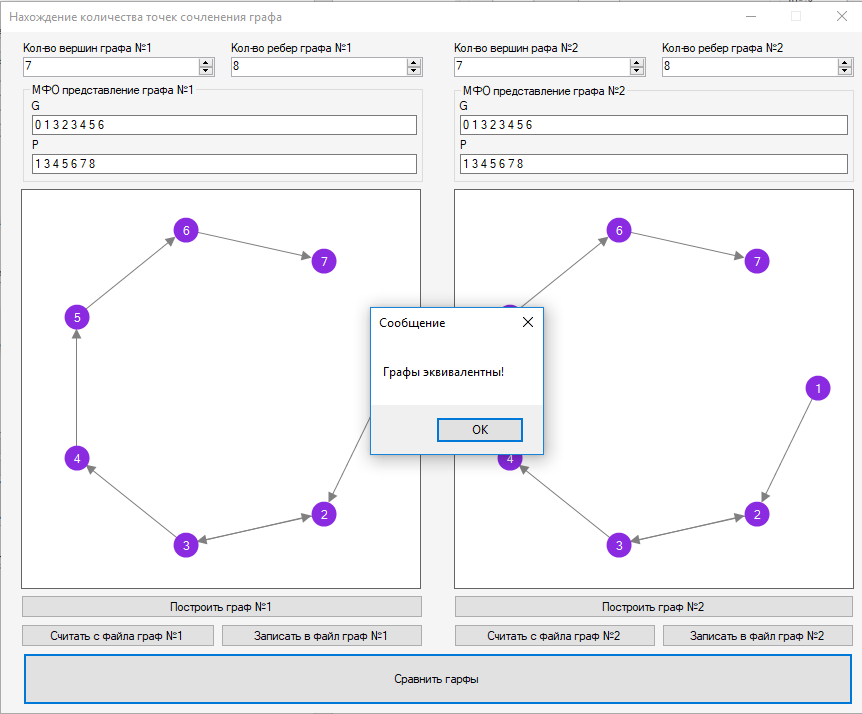


Рисунок Р.П.5 – Вывод результата проверки графов на эквивалентность

1. Закрыть программу можно нажав крестик в правом верхнем углу.

Сообщения пользователю.

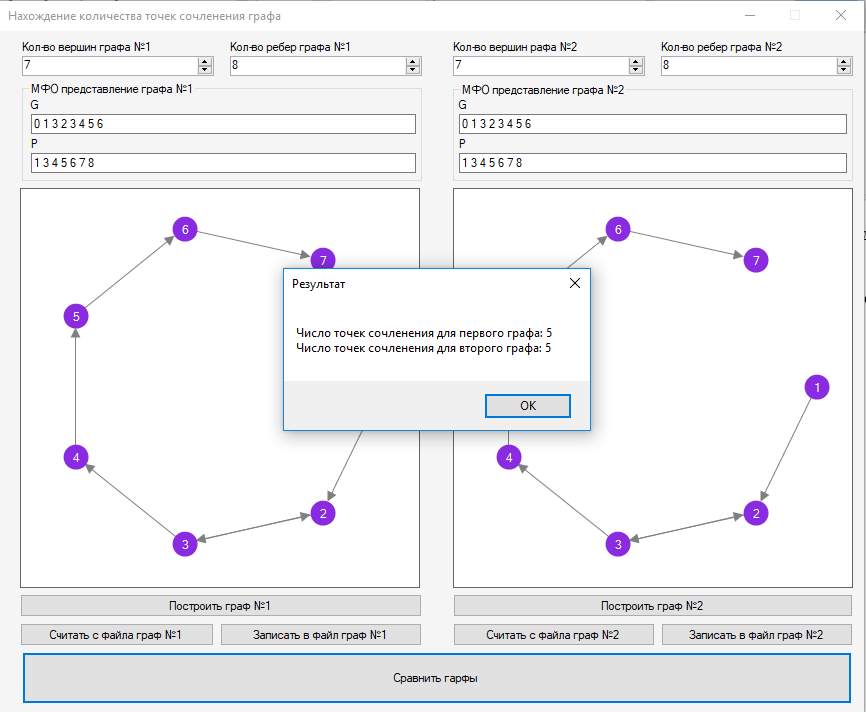


Рисунок П.B.7 - Результат нахождения числа точек сочленения графов

Что бы закрыть данное окно и дальше продолжить работу с программой нажмите кнопку «ОК».

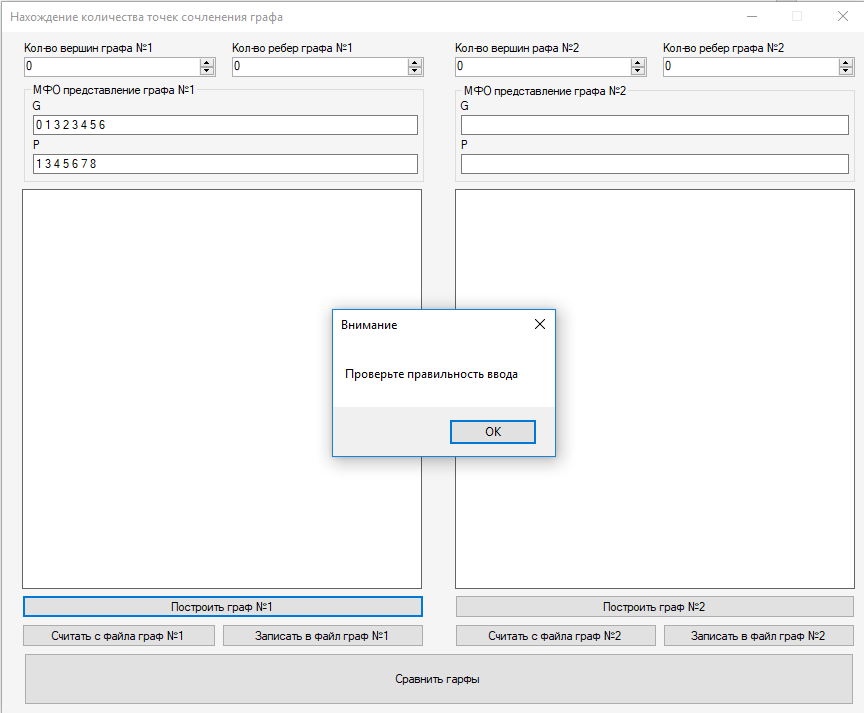


Рисунок Р.П.9 - Ошибка при вводе данных

В случае возникновения ошибки, необходимо нажать кнопку «Ок» и ввести данные заново, используя только цифры и пробел.

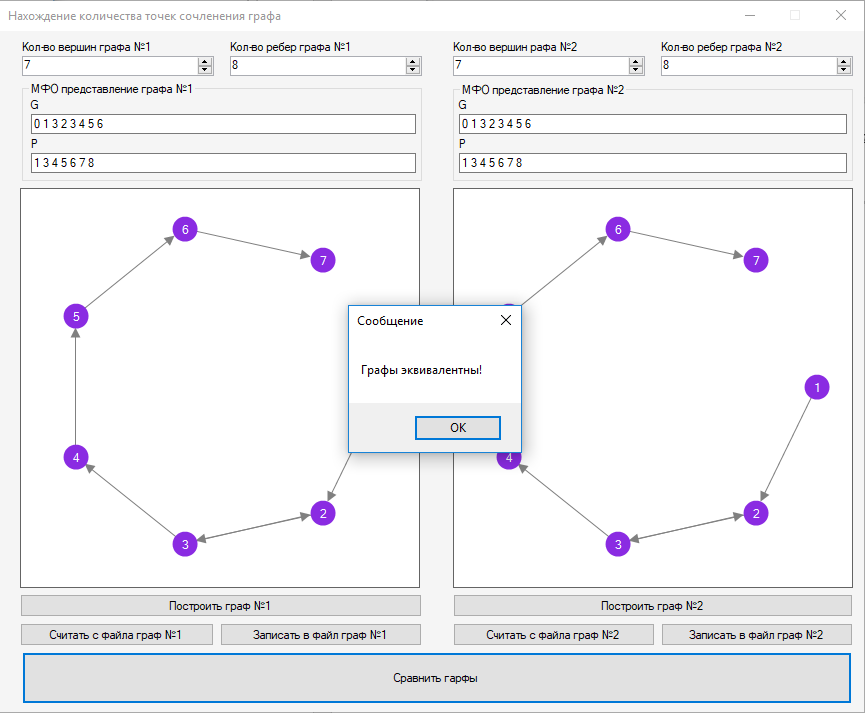


Рисунок Р.П.9 - Результат проверки графов на эквивалентность

Что бы закрыть данное окно и дальше продолжить работу с программой нажмите кнопку «ОК».

# ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ

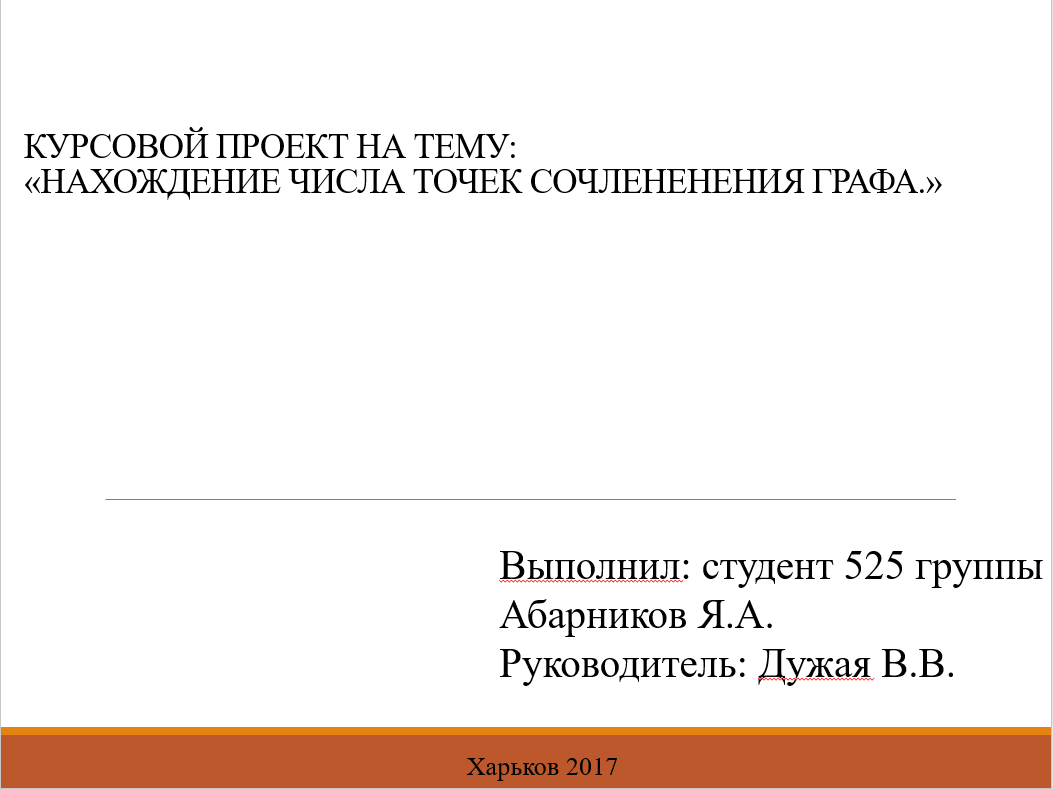


Рисунок П.1 – Слайд 1

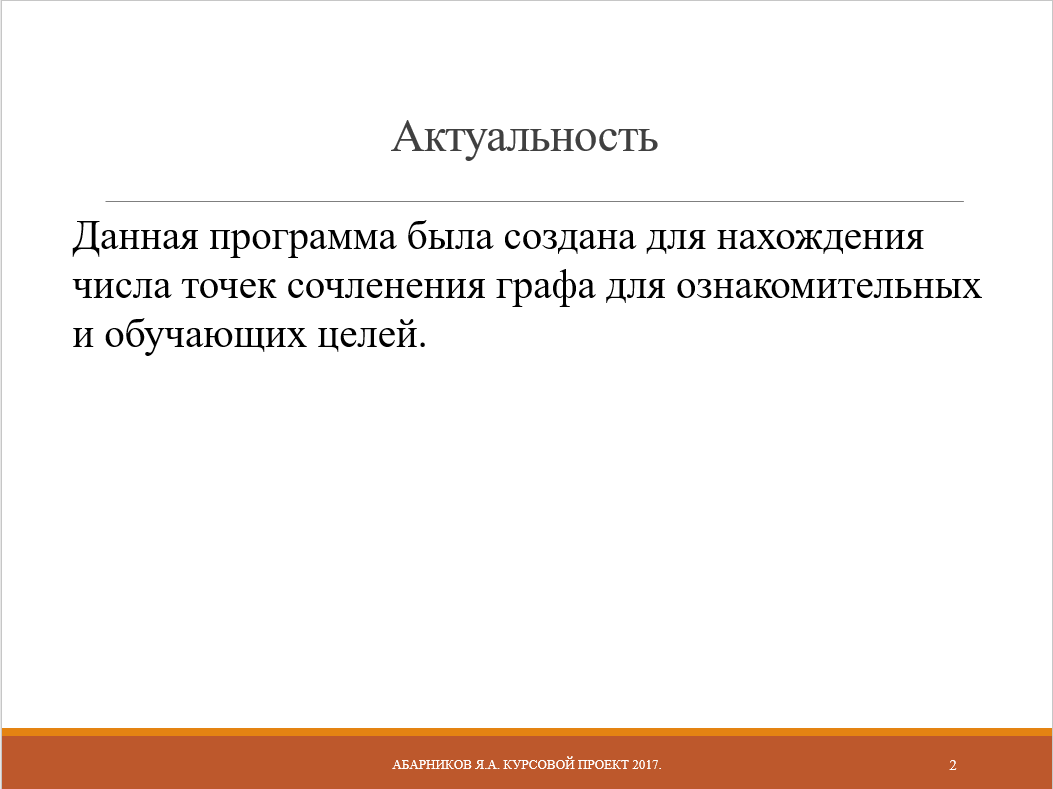


Рисунок П.2 – Слайд 2

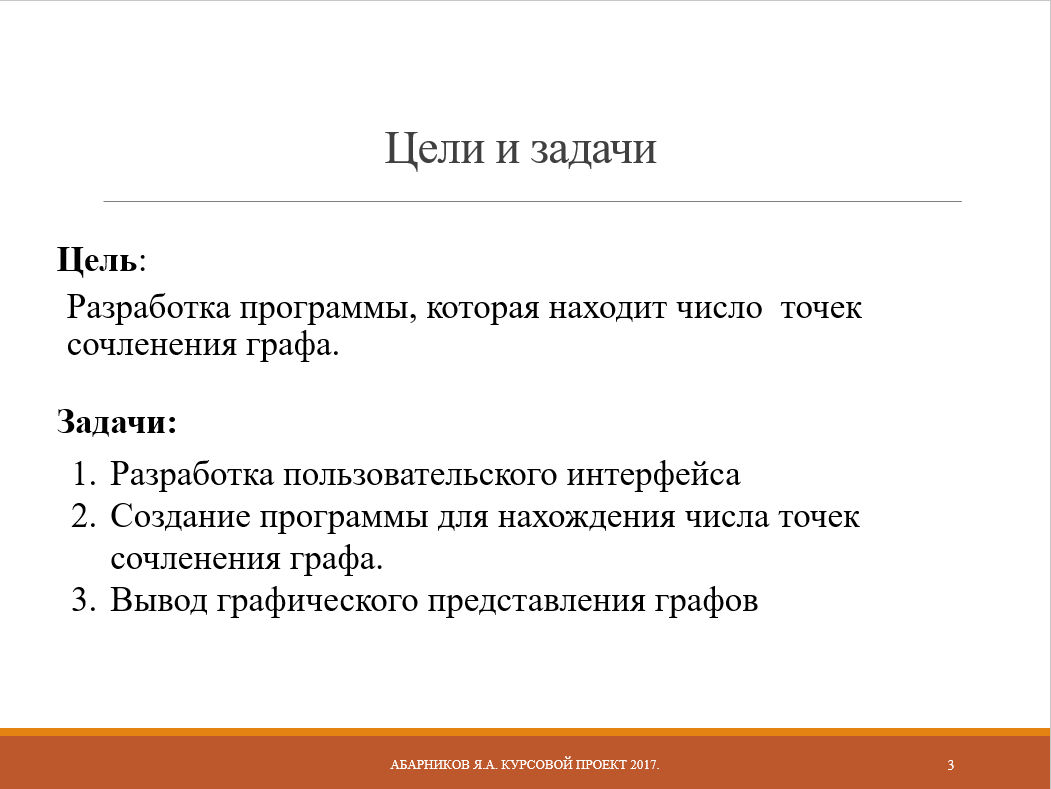


Рисунок П.3 – Слайд 3



Рисунок П.4 – Слайд 4



Рисунок П.5 – Слайд 5

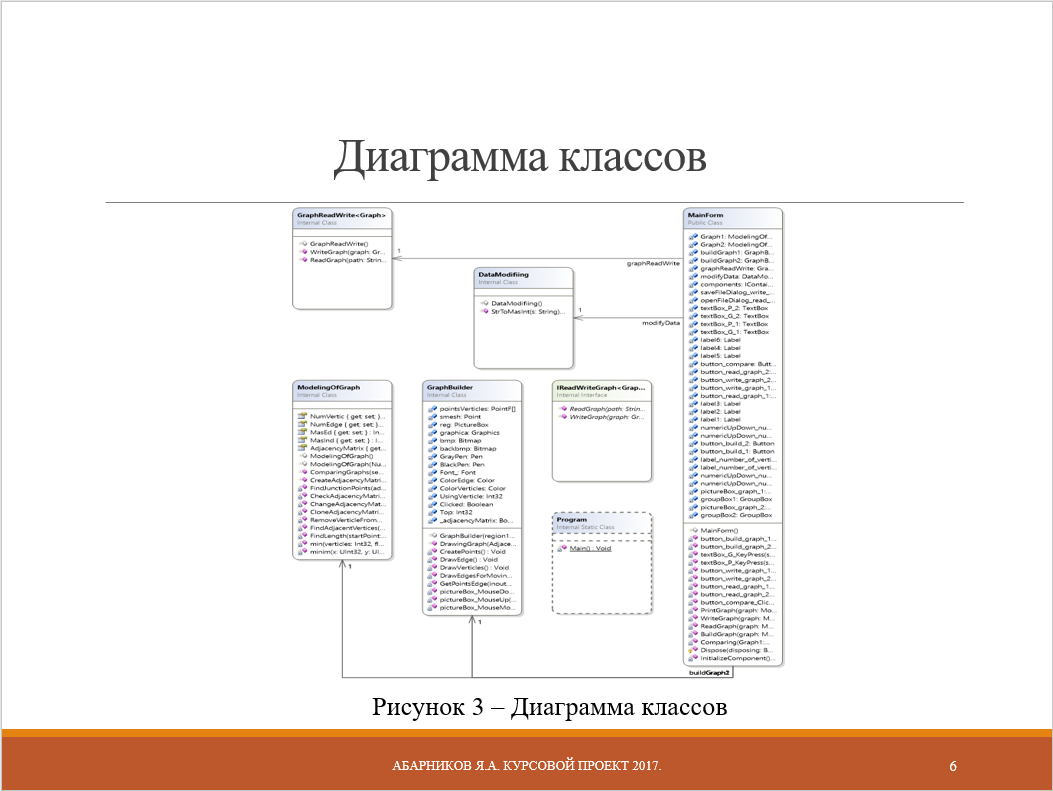


Рисунок П.6 – Слайд 6



Рисунок П.7 – Слайд 7

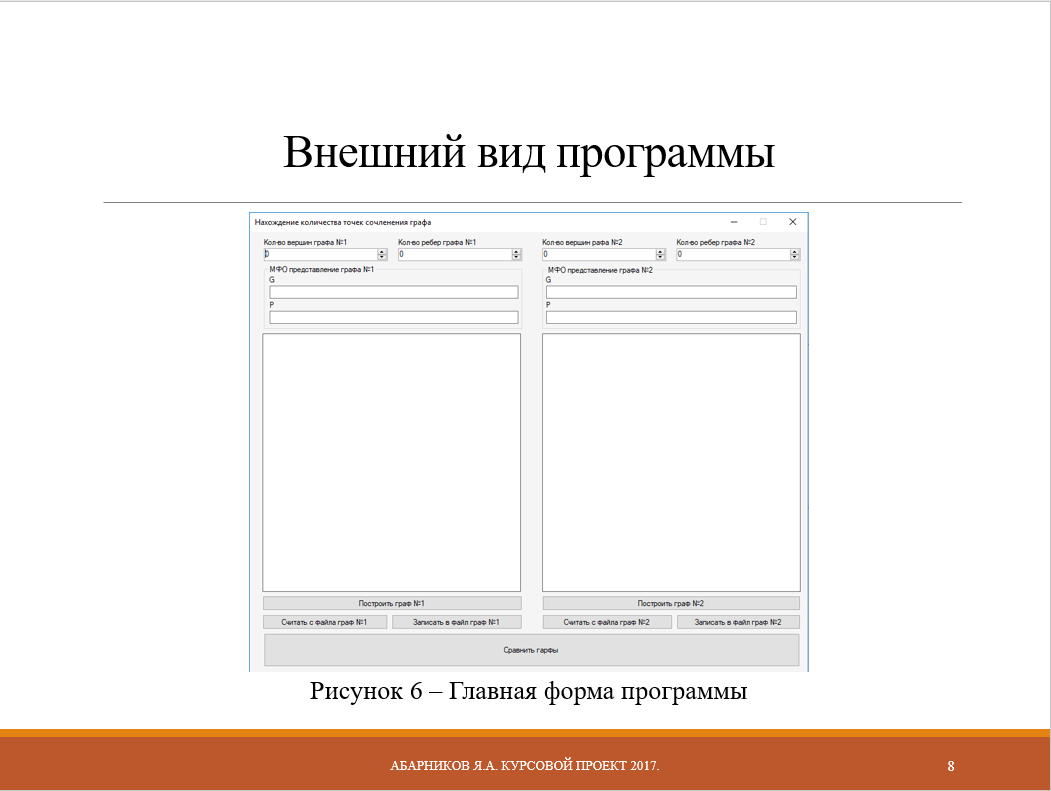


Рисунок П.8 – Слайд 8

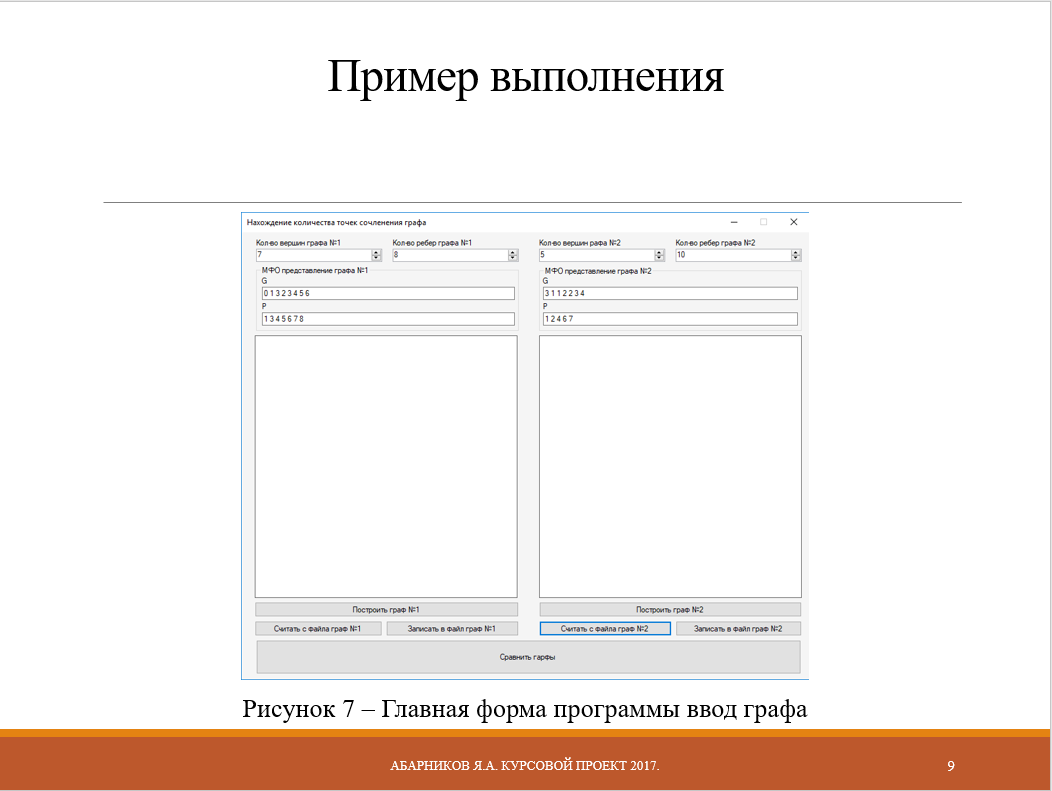


Рисунок П.9 – Слайд 9



Рисунок П.10 – Слайд 10

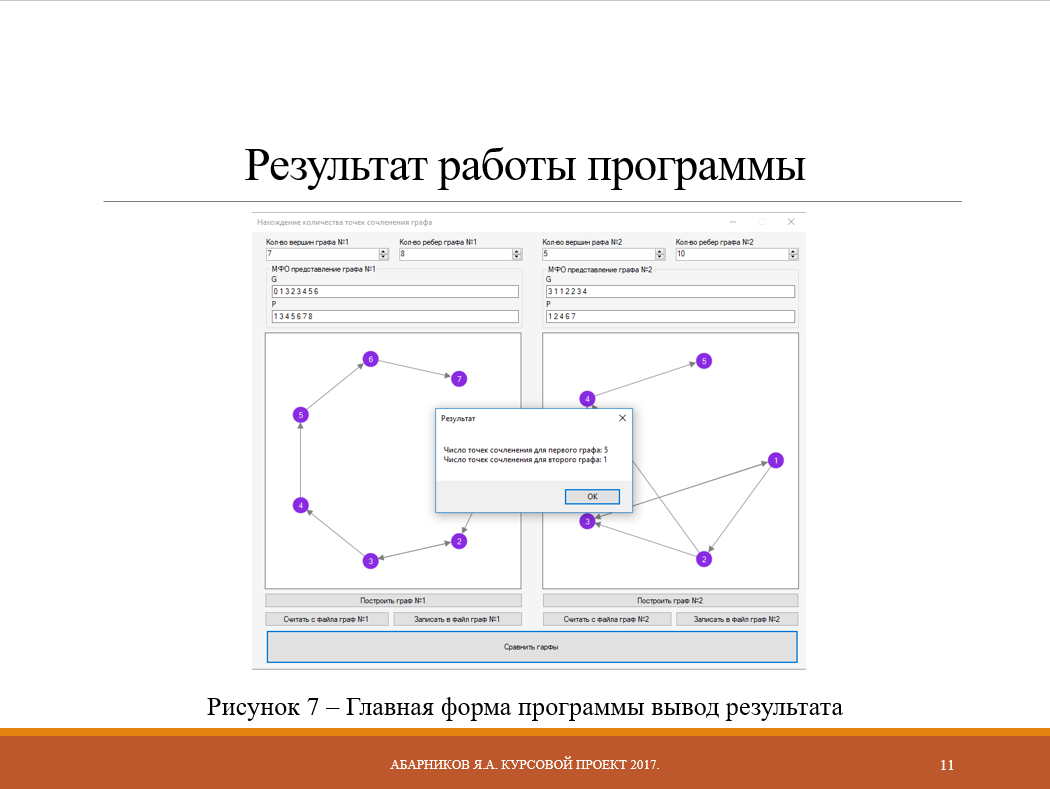


Рисунок П.11 – Слайд 11

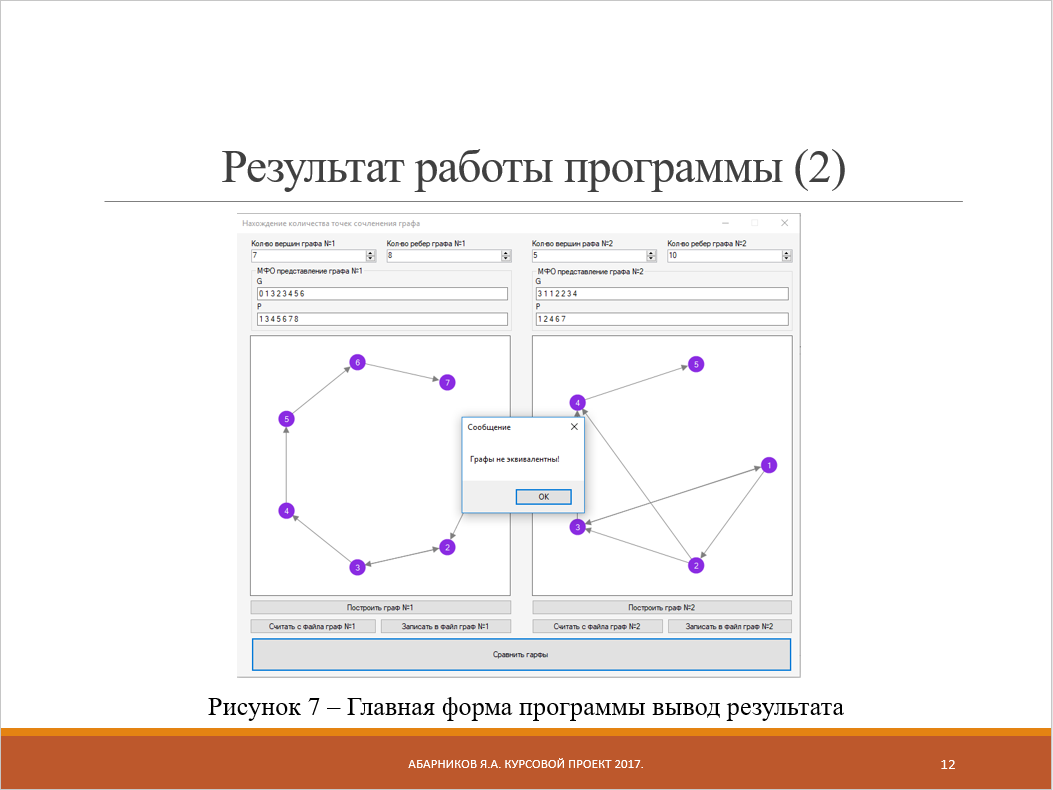


Рисунок П.12 – Слайд 12

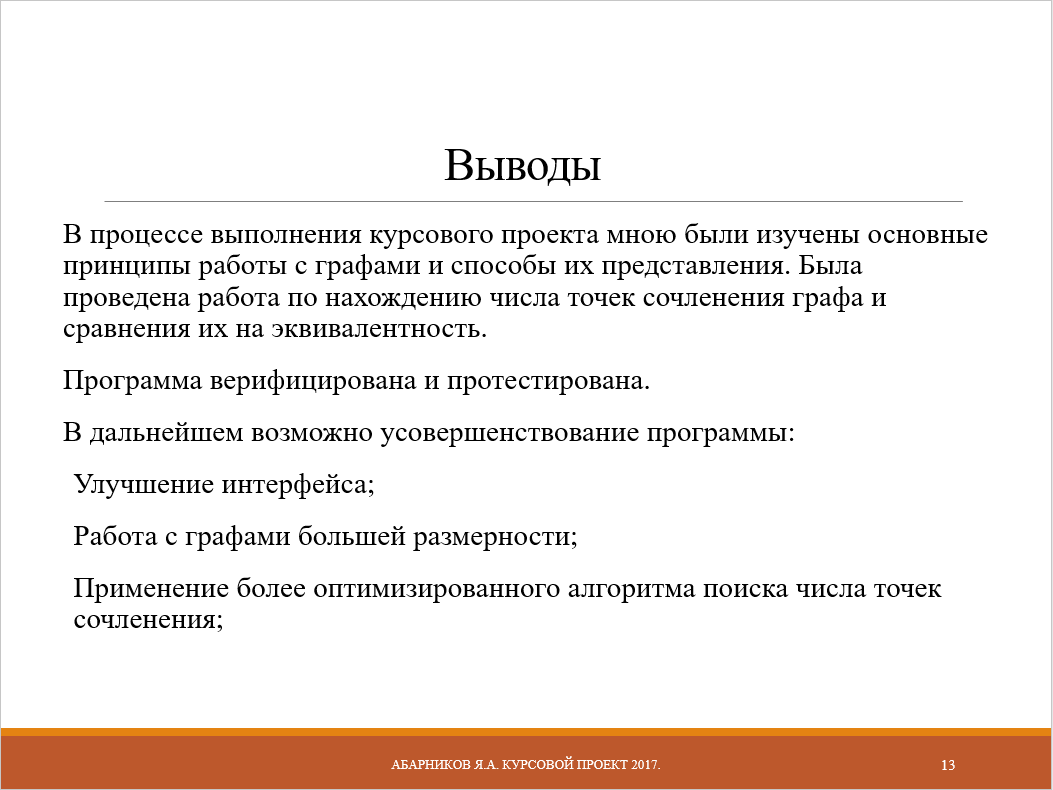


Рисунок П.13 – Слайд 13

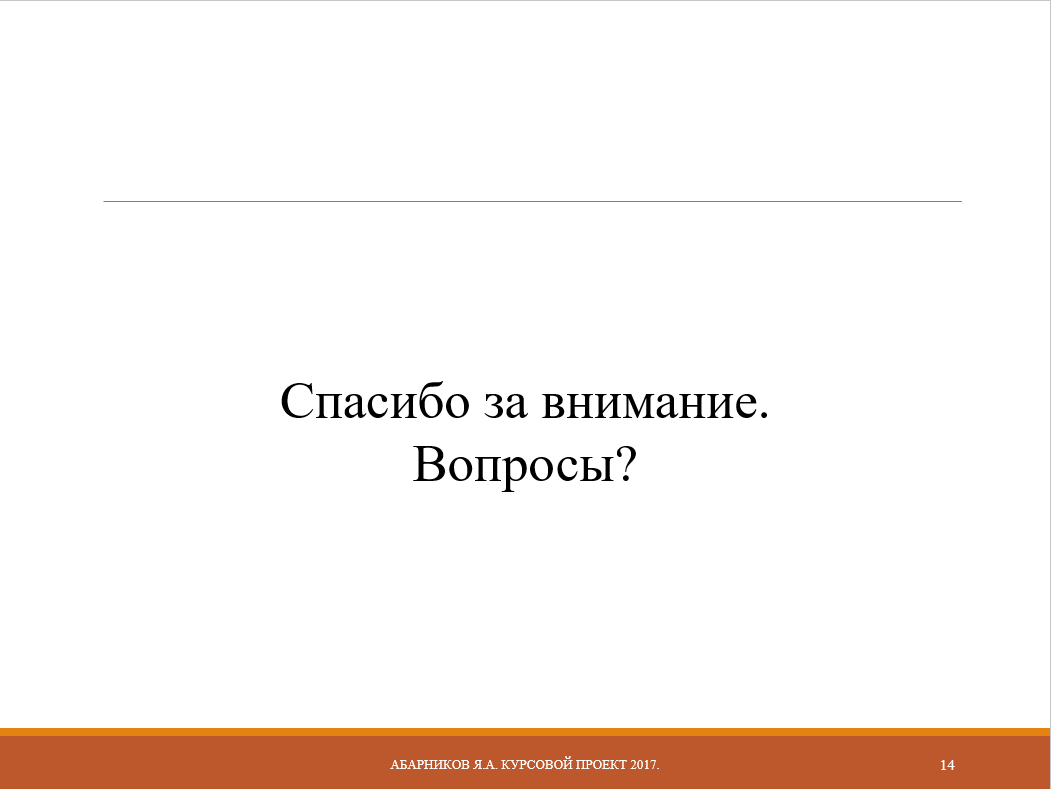


Рисунок П.14 – Слайд 14