**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по учебной практике**

**Тема: Визуализация алгоритмов на языке Java.**

|  |  |  |
| --- | --- | --- |
| Студент  гр. 5304 |  | Якушкин А.Э. |
| Студентка гр. 5304 |  | Гайдук М.А. |
| Студентка гр. 5304 |  | Федорова М.Д. |
| Руководитель |  | Томша А.Э. |

Санкт-Петербург

2017

**ЗАДАНИЕ**

**на учебную практику**

|  |  |  |
| --- | --- | --- |
| Студент группы 5304 Якушкин А.Э. | | |
| Студентка группы 5304 Гайдук М.А. | | |
| Студентка группы 5304 Федорова М.Д. | | |
| Тема практики: визуализация алгоритмов на языке Java | | |
| Задание на практику:  Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.  Алгоритм: поиск компонент сильной связности в ориентированном графе. | | |
| Сроки прохождения практики: 21.06.2017 – 04.07.2017 | | |
| Дата сдачи отчета: 28.06.2017 | | |
| Дата защиты отчета: 28.06.2017 | | |
|  | | |
| Студент гр.5304 |  | Якушкин А.Э. |
| Студентка гр.5304 |  | Гайдук М.А. |
| Студентка гр.5304 |  | Федорова М.Д. |
| Руководитель |  | Томша А.Э. |

**АННОТАЦИЯ**

В данной работе изучается и разрабатывается визуализатор алгоритма поиска сильно связных компонент в ориентированном графе. Программа разработана в среде IntelliJ IDEA. Язык, используемый для написания программы – Java. В проекте используется Qt Jambi — библиотека Java, представляющая собой обёртку Qt, каркаса графических приложений, которая позволяет Java разработчикам использовать Qt в своих проектах.

Программа подробно показывает процесс нахождения сильно связных компонент ориентированного графа. Для визуализации в программе предусмотрен графический интерфейс, описание которого приведено в данном отчёте. Работоспособность программы проверена рядом примеров.

**SUMMARY**

In this work, an algorithm for searching for strongly connected components in an oriented graph is studied and developed. The program was created in IntelliJ IDEA. The language, which used for writing the program, is Java. In this project it is used Qt Jambi  - a Java binding of the cross-platform application framework [Qt](https://en.wikipedia.org/wiki/Qt_(software)). It enables Java developers to use Qt within Java programming language.

The program shows process of searching for strongly connected components in an oriented graph in details. For visualization, the program provides a graphical interface, the description of which is given in this report. The working capacity of the program is checked by a number of tests.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc486340018)

[1. ОПИСАНИЕ АЛГОРИТМА 6](#_Toc486340019)

[2. СПЕЦИФИКАЦИЯ ПРОГРАММЫ 7](#_Toc486340020)

[2.1. Структура классов 7](#_Toc486340021)

[2.2. Описание интерфейса 8](#_Toc486340022)

[3. ТРЕБОВАНИЯ К ПРОГРАММЕ 9](#_Toc486340023)

[4. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ 10](#_Toc486340024)

[5. оСОБЕННОСТИ РЕАЛИЗАЦИИ 11](#_Toc486340025)

[6. ТЕСТИРОВАНИЕ 11](#_Toc486340026)

[6.1 Тестирование графического интерфейса 11](#_Toc486340027)

[6.2 Тестирование кода алгоритма 11](#_Toc486340028)

[ЗАКЛЮЧЕНИЕ 11](#_Toc486340029)

[СПИСОК ИСПОЛЬЗОВАННых ИСТОЧНИКОВ 11](#_Toc486340030)

[пРИЛОЖЕНИЕ а 11](#_Toc486340031)

# ВВЕДЕНИЕ

**Формулировка задания:** требуется разработать программу, визуализирующую алгоритм поиска компонент сильной связности(алгоритм Косорайю) в ориентированном графе. При этом должен присутствовать графический интерфейс.

Дан ориентированный граф G, множество вершин которого V и множество рёбер — E. Петли и кратные рёбра допускаются. Обозначим через n количество вершин графа, через m — количество рёбер.

Компонентой сильной связности (strongly connected component) называется такое (максимальное по включению) подмножество вершин C, что любые две вершины этого подмножества достижимы друг из друга, т.е. для любых u,v ϵ C:

u → v, v → u

где символом → здесь и далее мы будем обозначать достижимость, т.е. существование пути из первой вершины во вторую.

1. **ОПИСАНИЕ АЛГОРИТМА**

Для поиска компонент сильной связности используется алгоритм Косорайю:

1 шаг. Запустить серию обходов в глубину графа G, которая возвращает вершины в порядке увеличения времени выхода {\rm tout}, т.е. некоторый список \rm order.

2 шаг. Построить транспонированный граф G^T. Запустить серию обходов в глубину/ширину этого графа в порядке, определяемом списком \rm order (а именно, в обратном порядке, т.е. в порядке уменьшения времени выхода). Каждое множество вершин, достигнутое в результате очередного запуска обхода, и будет очередной компонентой сильной связности.

1. **СПЕЦИФИКАЦИЯ ПРОГРАММЫ**

2.1. Структура классов

В рамках данной работы будет использоваться структура классов, схожая с MVC-парадигмой. Моделью в данной работе выступает граф, реализованный с помощью отдельного класса-контейнера. Этот класс содержит в себе информацию о вершинах в виде списка инцидентности и должен поддерживать операцию транспонирования. Контроллером в данной схеме является класс, реализующий исследуемый алгоритм. Этот класс должен реагировать на события графического интерфейса пользователя и реализовывать алгоритм Косорайю как в пошаговом, так и в автоматическом режиме. Видом в данной структуре будет являться класс окна, содержащий в себе все элементы графического интерфейса пользователя, включая поле графического вывода графа. На рисунке 1 представлена UML-диаграмма проекта.

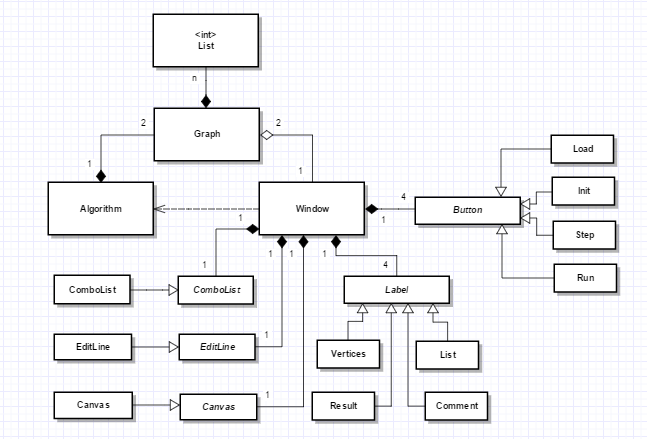


Рисунок 1. UML-диаграмма проекта

2.2. Описание интерфейса

Графический интерфейс пользователя состоит из 8 основных компонентов (см. рисунок 2).

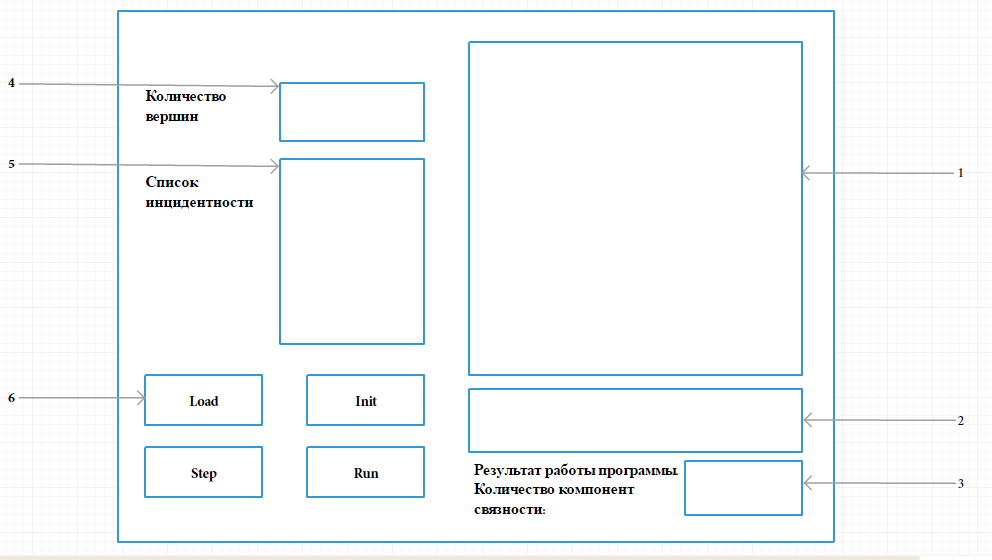


Рисунок 2. Схема графического интерфейса пользователя

Программа может поддерживать как ручной ввод пользователем, так и автоматический ввод из файла. Для того, чтобы ввести граф из файла, необходимо нажать кнопку Load в блоке 6 на рисунке 2. Если пользователь хочет ввести граф самостоятельно, то он должен ввести количество вершин графа в поле 4, после чего ввести список инцидентности в список 5. Для того, чтобы начать выполнение программы, необходимо нажать на кнопку Init в блоке 6, загрузив введенный граф в память. После этого кнопки Step и Run позволяют выполнить один шаг алгоритма и весь алгоритм полностью соответственно. Во время выполнения программы в окне 1 будет выводиться графическая интерпретация работы алгоритма с кратким пояснением в надписи 2; также в надписи 3 выводится итог работы алгоритма в виде количества найденных компонент сильной связности.

1. **ТРЕБОВАНИЯ К ПРОГРАММЕ**

Входные данные:

Граф, записанный в следующем формате: в первой строке количество вершин (целое число), каждая последующая *i-*я строка содержит последовательность вершин, инцидентных к (*i-2*)-й вершине. Отсчет вершин начинается с нуля.

Выходные данные:

Визуализация работы алгоритма Косорайю с краткими пояснениями.

1. **ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ**

Роли в бригаде были распределены следующим образом:

* архитектор-аналитик – Якушкин А.Э.;
* разработчик – Гайдук М.А.;
* тестировщик – Федорова М.Д.

Календарный план разработки приведен в таблице 1.

Таблица 1. Календарный план разработки

|  |  |  |
| --- | --- | --- |
| Дата | Этап работы | Ответственное лицо |
| 21 июня | Согласование темы проекта;  подготовка к работе (создание репозитория, настройка окружения и т.д.);  изучение теоретического материала. | Совместно |
| 22 июня | Составление спецификации проекта и планирование цикла разработки | Якушкин А.Э. |
| 23 июня | Согласование спецификации с руководителем практики и корректировка плана разработки | Якушкин А.Э. |
| 24-25 июня | Разработка графического интерфейса и основных классов | Гайдук М.А. |
| 25 июня | Тестирование и сборка прототипа | Федорова М.Д. |
| 26 июня | Согласование прототипа с руководителем практики | Совместно |
| 27 июня | Завершение реализации класса алгоритма и графа | Гайдук М.А. |
| 27 июня | Тестирование вывода графа на экран | Федорова М.Д. |
| 28 июня | Согласование рабочей версии проекта с руководителем практики | Якушкин А.Э. |
| 29 июня | Отладка проекта | Федорова М.Д. |
| 30 июня | Демонстрация и защита финальной версии | Совместно |

1. **оСОБЕННОСТИ РЕАЛИЗАЦИИ**
   1. **Описание классов**
   2. **Алгоритм работы**
2. **ТЕСТИРОВАНИЕ**

# Тестирование графического интерфейса

* 1. **Тестирование кода алгоритма**

**ЗАКЛЮЧЕНИЕ**

В ходе данной учебной практики была написана и протестирована программа, реализующая поиск компонент сильной связности ориентированного графа. Для реализации был использован алгоритм Косорайю. Программа позволяет, как демонстрировать пошаговую работу алгоритма, так и автоматически, с выводом кратким описанием.

В ходе написания данной работы были получены следующие знания и навыки:

* Основы программирования на языке Java
* Реализация графического интерфейса
* Работа с распределённой системой управления версиями Git и репозиторием GitHub

**СПИСОК ИСПОЛЬЗОВАННых ИСТОЧНИКОВ**

1. Java. Базовый курс. // Stepik. URL: [https://stepik.org/course/Java-Базовый-курс-187/syllabus](https://stepik.org/course/Java-%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BA%D1%83%D1%80%D1%81-187/syllabus) (дата обращения: 22.06.2017).
2. Алгоритм Косарайю.// Википедия. URL: <https://ru.wikipedia.org/wiki/Алгоритм_Косорайю> (дата обращения 24.06.2017)
3. Брюс Эккель. Философия Java. СПб.: Питер, 2009.

**пРИЛОЖЕНИЕ а**