

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по учебной практике**

**Тема:** Визуализация алгоритма Косарайю-Шарира с предоставлением графического интерфейса

Студент гр. 9382

\_\_\_\_\_

Юрьев С.Ю.

Студент гр. 9382

\_\_\_\_\_

Русинов Д.А.

Студент гр. 9382

\_\_\_\_\_

Герасев Г.А.

Руководитель

\_\_\_\_\_

Жангиров Т.Р.

Санкт-Петербург

2021

## ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Студент Юрьев С.Ю. группы 9382

Студент Русинов Д.А. группы 9382

Студент Герасев Г.А. группы 9382

Тема практики: Визуализация алгоритма Косарайю-Шарира с  
предоставлением графического интерфейса

Задание на практику:

Командная итеративная разработка визуализатора алгоритма(ов) на Kotlin с  
графическим интерфейсом.

Алгоритм: Косарайю-Шарира.

Сроки прохождения практики: 01.07.2020 – 14.07.2020

Дата сдачи отчета: 00.07.2020

Дата защиты отчета: 00.07.2020

Студент	_____	Юрьев С.Ю.
Студент	_____	Русинов Д.А.
Студент	_____	Герасев Г.А.
Руководитель	_____	Жангиров Т.Р.

## **АННОТАЦИЯ**

Целью практики является теоретическое и практическое изучение итеративной разработки программ с графическим интерфейсом в команде, с разделением основной задачи на несколько подзадач и разделением зон ответственности между всеми участниками процесса разработки.

Основной задачей практики является создание визуализатора алгоритма Косарайю-Шарира на языке Kotlin с использованием библиотеки Swing.

Способ проведения практики – дистанционный.

## **SUMMARY**

The goal of the practice is the theoretical and practical study of iterative development of programs with a graphical interface in a team, with the division of the main task into several subtasks and the division of areas of responsibility between all participants in the development process.

The main task of the practice is to create a visualizer for the Kosarayu-Sharira algorithm in the Kotlin language using the Swing library.

The practice is carried out remotely.

## СОДЕРЖАНИЕ

	Введение	5
1.	Требования к программе	6
1.1.	Исходные требования к программе*	0
1.2.	Уточнение требований после сдачи прототипа	0
1.3.	Уточнение требований после сдачи 1-ой версии	0
1.4.	Уточнение требований после сдачи 2-ой версии	0
2.	План разработки и распределение ролей в бригаде	0
2.1.	План разработки	0
2.2.	Распределение ролей в бригаде	0
3.	Особенности реализации	0
3.1.	Структуры данных	0
3.2.	Основные методы	0
3.3.		0
4.	Тестирование	0
4.1.	Тестирование графического интерфейса	0
4.2.	Тестирование кода алгоритма	0
4.3.	...	0
	Заключение	0
	Список использованных источников	0
	Приложение А. Исходный код – только в электронном виде	0

## **ВВЕДЕНИЕ**

Программа реализует визуальный интерфейс, для взаимодействия с алгоритмом поиска компонент сильной связности. В ней будет возможно изучение алгоритма Косарайю-Шарира благодаря наглядному изображению состояний алгоритма на введенных пользователем данных, и возможности их пошагового изучения, с переходом как вперед, так и отмены последнего шага.

# 1. ТРЕБОВАНИЯ К ПРОГРАММЕ

## 1.1. Исходные Требования к программе

### *Требование к реализации алгоритма:*

- 1) Алгоритм должен реализован так, чтобы можно было использовать любой тип данных. (Например через generic классы)
- 2) Алгоритм должен поддерживать возможность включения промежуточных выводов и пошагового выполнения

### *Требование к проекту:*

- 1) Возможность запуска через GUI и по желанию CLI (в данном случае достаточно вывода промежуточных выводов)
- 2) Загрузка данных из файла или ввод через интерфейс
- 3) GUI должен содержать интерфейс управления работой алгоритма, визуализацию алгоритма, окно с логами работы
- 4) Должна быть возможность запустить алгоритма заново на новых данных без перезапуска программы
- 5) Должна быть возможность выполнить один шаг алгоритма, либо завершить его до конца. В данном случае должны быть автоматически продемонстрированы все шаги
- 6) Должна быть возможность вернуться на один шаг назад
- 7) Должна быть возможность сбросить алгоритма в исходное состояние

### *Проектирование архитектуры:*

- 1) Архитектура должна быть модульной, то есть разделен на разные слои абстракции
- 2) Описание модулей проекта через UML диаграммы классов
- 3) Описание работы программы и алгоритма через UML диаграммы состояний
- 4) Описание работы программы и алгоритма через UML диаграммы последовательности
- 5) Желательно соблюдать принципы SOLID
- 6) Желательно использовать паттерны проектирования

## **2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ**

### **2.1. План разработки**

#### ***Итерация 1: (06.07.2021)***

- 1) Эскиз/Прототип GUI
- 2) Описание архитектуры

#### ***Итерация 2: (09.07.2021)***

- 1) Прототип GUI с реализацией частичного функционала
- 2) Реализованный алгоритм

#### ***Итерация 3: (12.07.2021)***

- 1) Полностью рабочий GUI и CLI
- 2) Реализация взаимодействия с алгоритмом

#### ***Финал: (14.07.2021)***

- 1) Внесение правок/устранения недочетов

### **2.2. Распределение ролей в бригаде**

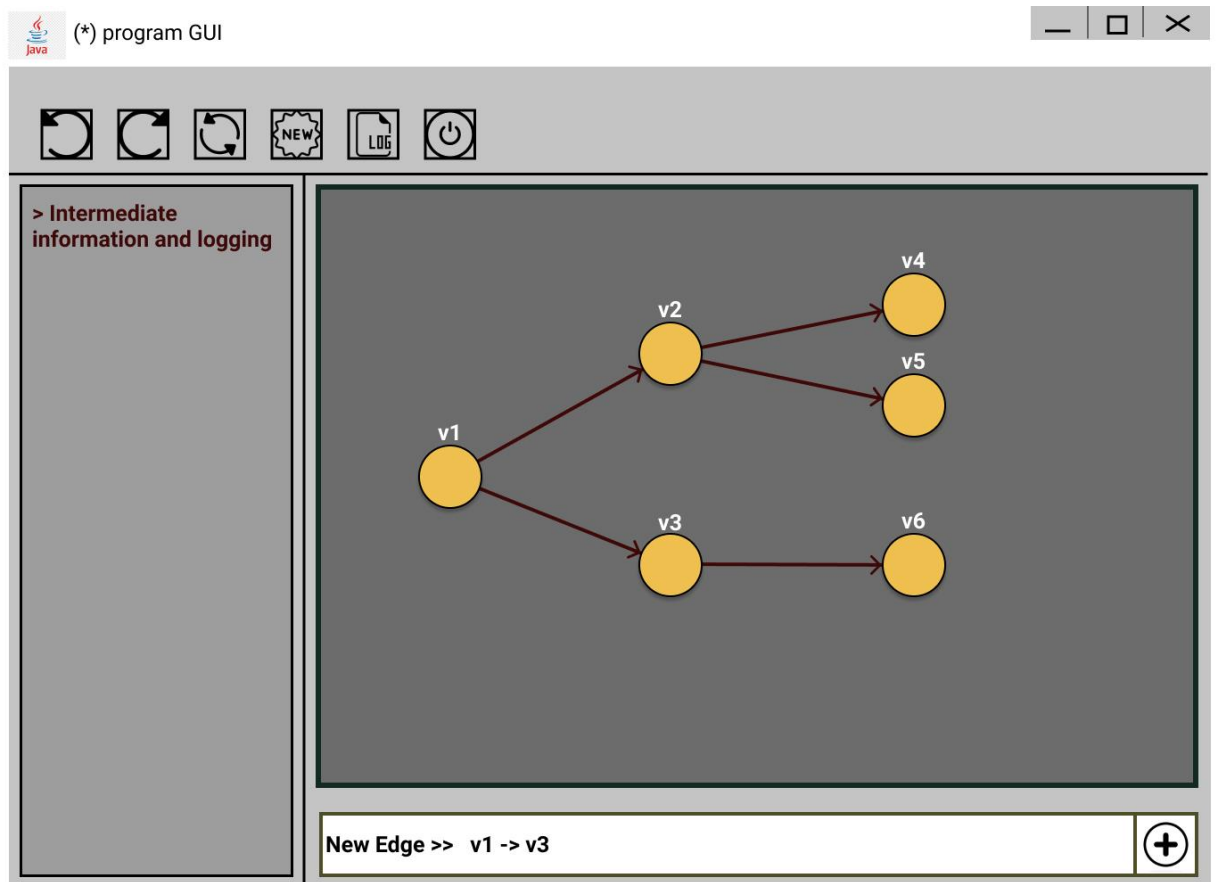
Русинов Д.А. – написание визуализатора алгоритма, написание GUI приложения, обеспечение корректной связи GUI с основным алгоритмом

Герасев Г.А. – написание кода основного алгоритма, разработка общей архитектуры приложения

Юрьев С.Ю. – тестирование корректной работы GUI и основного алгоритма, разработка эскиза GUI, теоретическая и программная помощь команде

### 3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

#### Эскиз GUI:



#### ОПИСАНИЕ АРХИТЕКТУРЫ:

Программа делится на две основнй части – пользовательский интерфейс и основной алгоритм программы.

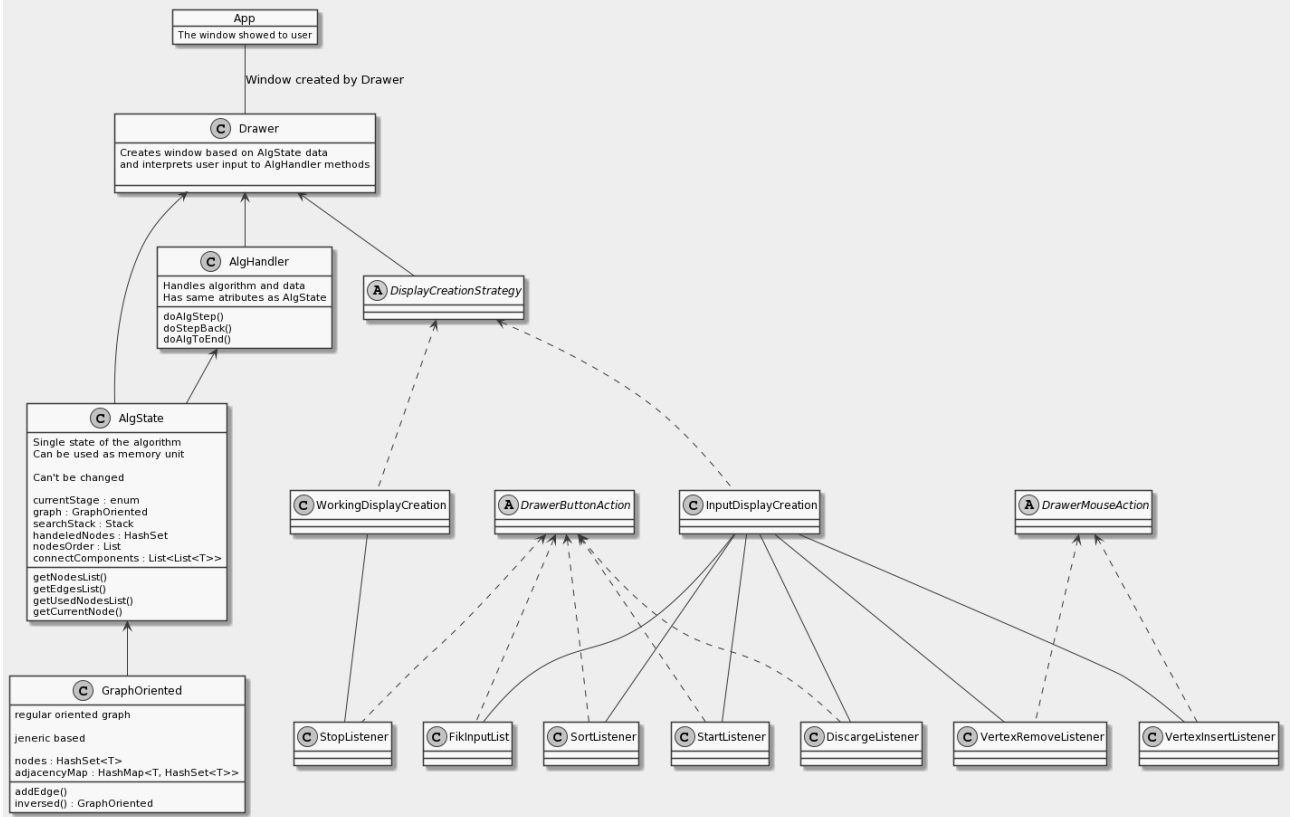
Последний устроен следующим образом. Имеется класс, в котором сохраняется состояние алгоритма Косарайю-Шарира (вместе с текущим этапом алгоритма). Реализуется метод, вычисляющий следующий шаг алгоритма, а также реализуется закрытый внутренний класс, позволяющий сохранять и

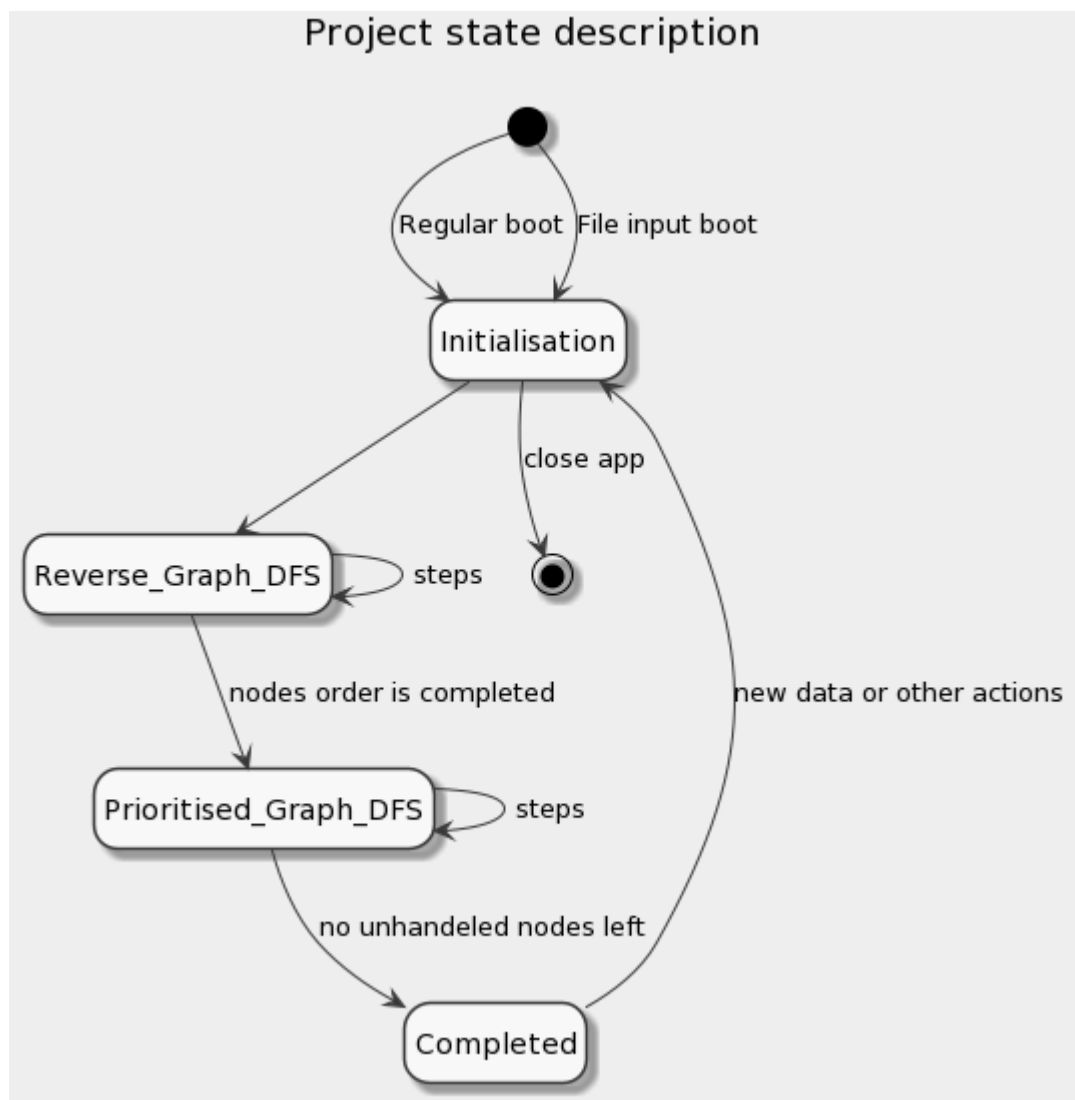
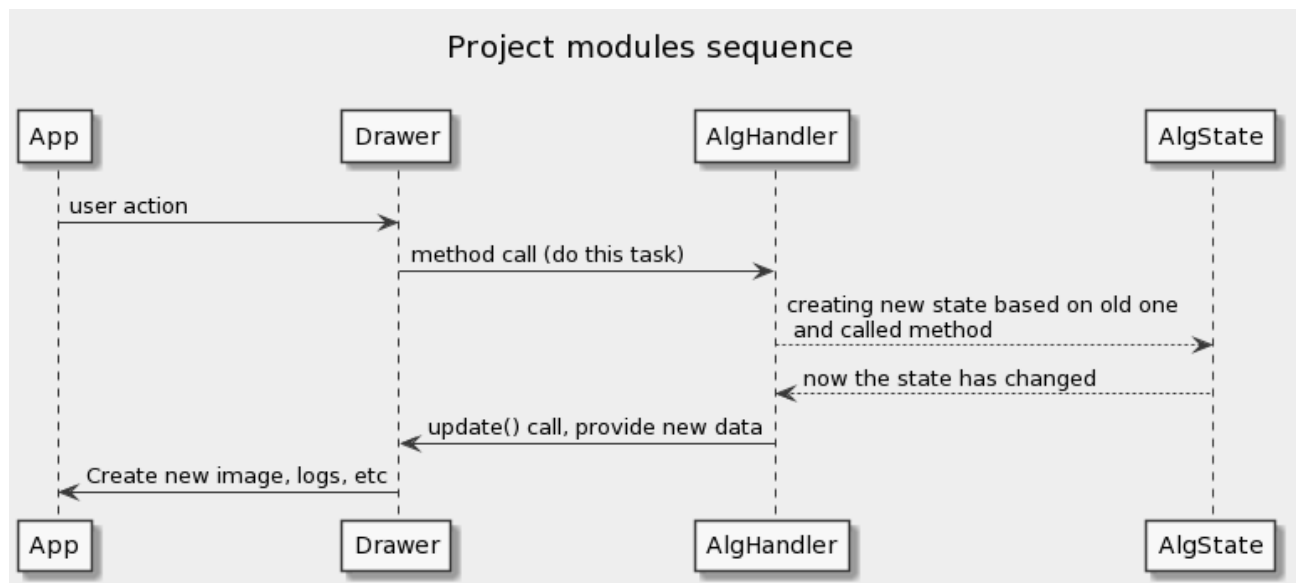


восстанавливать состояния алгоритма с помощью паттерна «Снимок». Также эти состояния будут передаваться классам ответственным за пользовательский интерфейс, и пользуясь методами данного класса будет получена вся необходимая для отрисовки информация. В дальнейшем создание состояний алгоритма будет передано специальному классу-строителю.

Пользовательский интерфейс же работает по следующему принципу. Класс окна содержит стратегию рисования текущих компонентов. Всего определено две стратегии: отображение меню редактирования графа и отображение меню управления ходом алгоритма. Каждый компонент имеет «слушателей» за действиями пользователя. При взаимодействии пользователя с компонентом, «слушатель» выполнит ряд инструкций, которые каким-либо образом обновляют пользовательский интерфейс.

# Project modules description





## 4. ТЕСТИРОВАНИЕ

Найденные ошибки:

1) Ввод:  $A \leftrightarrow B$ ,  $B \leftrightarrow C$ ,  $A \leftrightarrow C$

Пояснение: цикл, где каждая вершина связана с каждой

Вывод программы:

Graph

a -> [b, c]

b -> [a]

c -> [a]

Print alg result

[c, b, a]

Ошибка: неправильное отражение введенного графа, итоговый ответ верный

Ошибка была исправлена!!!

2) Ввод:  $A \leftrightarrow B$ ,  $B \leftrightarrow C$ ,  $A \leftrightarrow C$

Пояснение: цикл, где каждый связан с каждым

Вывод программы:

Graph

a -> [b, c]

b -> [a, c]

c -> [a, b]

Print alg result

[b, c, b, a]

Ошибка: в итоговый ответ вершина b была добавлена дважды, отражение графа верное

Ошибка была исправлена!!!

3) Ввод:

Пояснение: пустой ввод

Вывод программы:

Exception in thread "main" java.lang.IndexOutOfBoundsException: Empty list contain element at index 0.

Ошибка: при пустом вводе в алгоритме была попытка получить

несуществующий элемент списка в результате чего вылетало исключение

Ошибка была исправлена!!!

### Проведенные успешные тесты:

1) Ввод:  $A \leftrightarrow B, B \leftrightarrow C, A \leftrightarrow C$

Пояснение: цикл, где каждый связан с каждым

Вывод программы:

Graph

a -> [b, c]

b -> [a, c]

c -> [a, b]

Print alg result

[b, c, a]

Отражение графа и ответ корректны.

2) Ввод:  $A \rightarrow B, B \rightarrow C, B \rightarrow D, D \rightarrow E, E \rightarrow C, C \rightarrow A$

Пояснение: цикл с меньшим циклом внутри

Вывод программы:

Graph

b -> [c, d]

c -> [a]

d -> [e]

e -> [c]

Print alg result

[c, e, d, b, a]

Отражение графа и ответ корректны.

3) Ввод:  $A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow F$

Пояснение: цепочка не замкнутая на концах

Вывод программы:

Graph

a -> [b]

b -> [c]

c -> [d]

d -> [e]

e -> [f]

[f] [e] [d] [c] [b] [a]

4) Ввод:  $A \leftrightarrow B, C \leftrightarrow D, E \leftrightarrow F$

Вывод программы:

a -> [b]

$$b \rightarrow [a]$$

c -> [d]

d -> [c]

e -> [f]

f -> [e]

[f, e] [d, c] [b, a]

5) Ввод:  $AB \rightarrow B$ ,  $B \rightarrow AB$ ,  $AB \rightarrow A$ ,  $A \rightarrow AB$

Вывод программы:

a -> [ab]

$$ab \rightarrow [a, b]$$

**b -> [ab]**

[b, a, ab]

6) Ввод: ``->``, ``->``, ``->``

Вывод программы:

## Graph

Отражение графа и ответ корректны, т.к. пустые строки не могут служить названием вершины

7) Ввод:  $A \rightarrow A, B \rightarrow B, B \rightarrow A$

Пояснение: использование путей от вершины к ней же самой

Вывод программы:

Graph

a -> [a]

b -> [a, b]

Print alg result

[a] [b]

Отражение графа и ответ корректны.

8) Ввод: A->A, A->A, A->A, B->B, B->B, B->A, B->A

Пояснение: использование одних путей по несколько раз

Вывод программы:

Graph

a -> [a]

b -> [a, b]

Print alg result

[a] [b]

Отражение графа и ответ корректны.

9) Ввод: A->B, B->C, C->A, B->D, D->E, E->F, F->D

Пояснение: соединение нескольких циклов с помощью одного ребра

Вывод программы:

Graph

a -> [b]

b -> [c, d]

c -> [a]

d -> [e]

e -> [f]

f -> [d]

Print alg result

[f, e, d] [c, b, a]

Отражение графа и ответ корректны.

10) Ввод: Hello Summer Practice->Hello world, Hello world->Hello Kotlin, Hello Kotlin->Hello Summer Practice

Пояснение: использование в качестве названий целых предложений

Вывод программы:

Graph

Hello Summer Practice -> [Hello world]

Hello Kotlin -> [Hello Summer Practice]

Hello world -> [Hello Kotlin]

Print alg result

[Hello Kotlin, Hello world, Hello Summer Practice]

Отражение графа и ответ корректны.

11) Ввод: a->b, b->A, A->c, c->C

Пояснение: использование в качестве названий одинаковых значений с использованием разных регистров

Вывод программы:

Graph

A -> [c]

a -> [b]

b -> [A]

c -> [C]

Print alg result

[C] [c] [A] [b] [a]

Отражение графа и ответ корректны, т.к. названия вершин регистрозависимы.

12) Ввод: A->A, B->B, C->D, D->C

Пояснение: цикл + несколько отдельных не связанных ни с чем вершин

Вывод программы:

Graph

a -> [a]

b -> [b]

c -> [d]

d -> [c]

Print alg result

[d, c] [b] [a]

Отражение графа и ответ корректны.

13) Ввод: 1->;, ;->\$, \$->\n, \n->ctrl+D

Пояснение: использование в качестве названий спец символов и цифр

Вывод программы:

Graph

1 -> [;]

\n -> []

\$ -> [\n]

; -> [\$]



Print alg result

[] [\n] [\$] [;] [1]

Отражение графа и ответ корректны.

14) Ввод:

Пояснение: пустой ввод

Вывод программы:

Graph

Print alg result

Отражение графа и ответ корректны.

Вывод:

Были протестированы все стандартные и крайние случаи возможного пользовательского ввода. Обо всех найденных ошибках сообщалось ответственному за алгоритм, после чего они исправлялись. После исправления проводились повторные тесты на тех же данных, а так же на новых данных, имеющих схожую структуру.

Тестирование возможности файлового ввода:

Данные:

\_c

?ab

cd

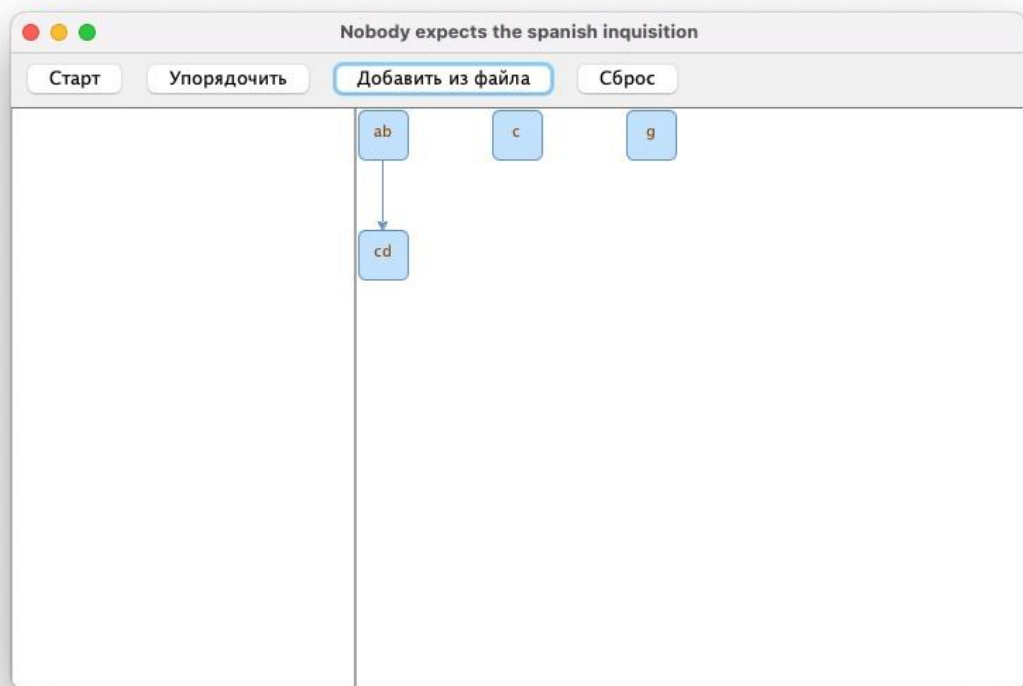
kl

fg

?df

\_g

Результат:



## **ЗАКЛЮЧЕНИЕ**

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

**ПРИЛОЖЕНИЕ А**  
**НАЗВАНИЕ ПРИЛОЖЕНИЯ**

полный код программы должен быть в приложении, печатать его не надо