**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по учебной практике**

**Тема: Сильно связные компоненты орграфа.**

| Студентка гр. 1381 |  | Тулегенова А.О. |
| --- | --- | --- |
| Студент гр. 1381 |  | Сагидуллин Э.Р. |
| Студентка гр. 1381 |  | Васильева О.М. |
| Руководитель |  | Фирсов М.А. |

Санкт-Петербург

2023

**ЗАДАНИЕ**

**НА УЧЕБНУЮ ПРАКТИКУ**

| Студентка Тулегенова А.О. группы 1381 | | |
| --- | --- | --- |
| Студент Сагидуллин Э.Р. группы 1381 | | |
| Студентка Васильева О.М. группы 1381  Тема практики: Сильно связные компоненты орграфа | | |
| Задание на практику:  Командная итеративная разработка визуализатора алгоритма на Kotlin с графическим интерфейсом.  Алгоритм: алгоритм Косарайю. | | |
| Сроки прохождения практики: 30.06.2023 – 13.07.2023 | | |
| Дата сдачи отчета: 00.07.2023 | | |
| Дата защиты отчета: 00.07.2023 | | |
|  | | |
| Студентка |  | Тулегенова А.О. |
| Студент |  | Сагидуллин Э.Р. |
| Студентка |  | Васильева О.М. |
| Руководитель |  | Фирсов М.А. |

**АННОТАЦИЯ**

Данная работа посвящена изучению и реализации алгоритма Косарайю для нахождения сильно связных компонент в орграфе. В работе представлен графический интерфейс, разработанный для визуализации работы алгоритма Косарайю. Интерфейс позволяет пользователю вводить орграфы, состоящие из вершин и направленных ребер, а затем выполнять алгоритм Косарайю для поиска сильно связных компонент. Результаты работы алгоритма визуализируются в виде графической диаграммы, где каждая сильно связная компонента обозначается отдельным цветом.

**SUMMARY**

This work is devoted to the study and implementation of the Kosaraju algorithm for finding strongly connected components in a digraph. The paper presents a graphical interface designed to visualize the operation of the Kosaraju algorithm. The interface allows the user to enter digraphs consisting of vertices and directed edges, and then execute the Kosaraju algorithm to search for strongly connected components. The results of the algorithm are visualized in the form of a graphical diagram, where each strongly connected component is indicated by a separate color.

**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.  1.4 | Уточнение требований после сдачи 1-ой версии  Уточнение требований после сдачи 2-ой версии | 0  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**

Данная работа посвящена изучению и реализации алгоритма Косарайю для определения сильно связных компонент в орграфах. Сильно связные компоненты являются важным понятием в теории графов и находят широкое применение в различных областях, таких как анализ социальных сетей, оптимизация транспортных систем и другие.

Целью данной практической работы является исследование и реализация алгоритма Косарайю, а также разработка графического интерфейса для визуализации его работы. Графический интерфейс предоставляет пользователю возможность вводить орграфы с вершинами и направленными ребрами, а затем запускать алгоритм Косарайю для поиска сильно связных компонент. Результаты работы алгоритма визуализируются в виде графической диаграммы, где каждая сильно связная компонента обозначается уникальным цветом.

Реализация алгоритма Косарайю включает в себя несколько этапов: первоначальная обработка графа, построение транспонированного графа, выполнение обхода в глубину согласно измененному порядку вершин и определение сильно связных компонент. Полученная реализация алгоритма и графический интерфейс предоставляют удобный инструмент для анализа сильно связных компонент в орграфах. Исследователи и практики могут эффективно использовать этот инструмент для изучения структуры орграфов и выявления важных связей и зависимостей.

**1. ТРЕБОВАНИЯ К ПРОГРАММЕ**

**1.1. Исходные Требования к программе**

**1.1.1 Требования к графическому интерфейсу**

Данное приложение должно иметь графический интерфейс, что означает, что пользователи смогут взаимодействовать с программой с помощью графических элементов, таких как кнопки, поля ввода и т. д. Также следует предусмотреть возможности взаимодействия с графическими элементами с помощью мыши.

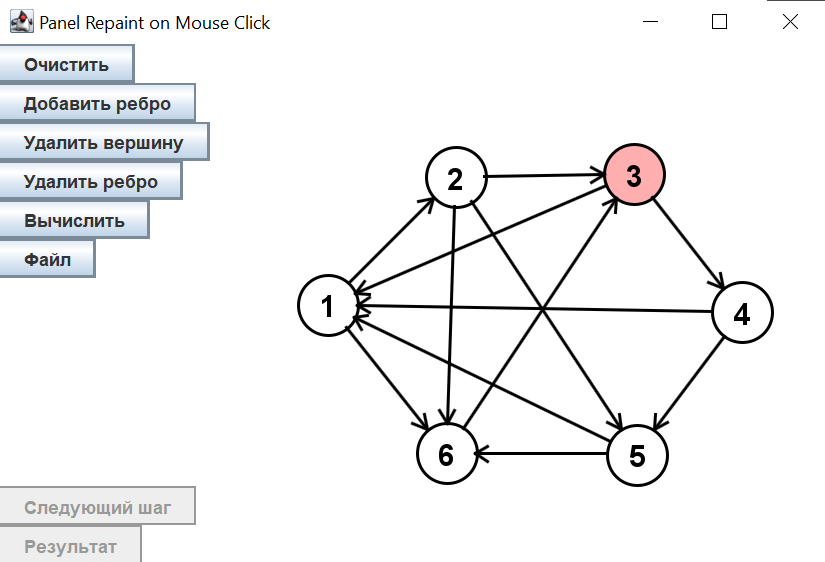
****

Рисунок 1. Скриншот графического интерфейса.

Функциональность программы:

1. Кнопка "Очистить":

При нажатии данной кнопки весь граф на холсте будет удален. Все вершины и ребра, отображаемые на холсте, будут удалены, и холст будет очищен.

2. Кнопка "Добавить ребро":

Для соединения двух вершин необходимо выполнить следующие шаги:

1. Выделите одну вершину, щелкнув на нее.

2. Выделите вторую вершину, щелкнув на нее.

3. После выполнения этих шагов между выделенными вершинами будет установлено ребро.

Ребро будет нарисовано между выбранными вершинами на холсте.

Текстовые пояснения при этом также будут отображаться на графическом интерфейсе. Для удобства, первая вершина выделяется цветом.

3. Кнопка "Удалить вершину":

При нажатии данной кнопки будет удалена выбранная вершина и все ребра, связанные с этой вершиной. Щелчок на вершину, которую нужно удалить, выберет ее и все соединенные с ней ребра будут также удалены. При этом будет отображаться текстовое пояснение с просьбой кликнуть на вершину.

4. Кнопка "Удалить ребро":

Для удаления ребра между двумя вершинами необходимо выполнить следующие шаги:

1. Выделите одну вершину, щелкнув на нее.

2. Выделите вторую вершину, щелкнув на нее.

3. После выполнения этих шагов между выделенными вершинами будет удалено ребро, если оно существует.

Текстовые пояснения при этом также будут отображаться на графическом интерфейсе. Для удобства, первая вершина выделяется цветом.

5. Кнопка "Вычислить":

При нажатии данной кнопки будет запущен алгоритм Косарайю для поиска сильно связных компонент графа. После нажатия на холсте будет представлен первый шаг алгоритма. Выводиться текстовое пояснение, что агоритм был запущен и объяснение того, что произошло на первом шаге. Также при нажатии этой кнопки, кнопки, ответственные за удаление/добавление ребер и вершин, становятся недоступными, пока не будут выполнены все шаги алгоритма.

6. Кнопка "Файл":

При нажатии данной кнопки открывается проводник для выбора файла. Пользователь может выбрать файл, содержащий описание графа в виде списка ребер. Выбранный файл будет загружен, и граф из файла будет отображен на холсте.

7. Кнопка "Следующий шаг":

При нажатии данной кнопки происходит переход к следующему шагу алгоритма. При этом выводятся текстовые пояснения того, какие изменения произошли на данном шаге. Если был выполнен последний шаг алгоритма, то выполнение алгоритма завершаются и кнопки, ответственные за удаление/добавление ребер и вершин, вновь становятся доступными.

8. Кнопка "Результат":

При нажатии данной кнопки сразу визуализируется конечный результат выполнения программы. Кнопки, ответственные за удаление/добавление ребер и вершин, вновь становятся доступными.. Появляется текстовое пояснение, что на холсте представлено конечное разбиение вершин на компоненты связности. Выполнение алгоритма завершено.

Примечания:

Вся визуализация графа и выполнение операций происходят на холсте программы. Для выделения вершины и соединения ребер необходимо использовать щелчки мыши. Графический интерфейс программы обеспечивает удобное взаимодействие с графами и операциями над ними.   
 Добавление вершин происходит при нажатии на холст мышкой. Также возможно перемещение вершин по холсту, если зажать вершину левой кнопкой мыши и перетащить ее по холсту.

**1.1.1. Требования к вводу исходных данных**

Должна быть возможность считывания входных данных из файла. Данные будут представлены в следующем виде: на каждой строке будут записаны через пробел индексы двух вершин (индексы заданы в виде чисел), соответствующие вершинам и направленному ребру графа. Также входные данные могут создаваться при работе в приложении, то есть при добавлении вершин и ребер на панели приложения, а именно при нажатии мышки или соответствующей кнопки на панели приложения, задается некоторый граф, на котором будет запускаться алгоритм. После задания графа в приложении или с помощью файла данные будут преобразованы к спискам смежности.

**1.1.2. Требования к визуализации**

Приложение предусматривает возможность взаимодействия с графическим интерфейсом, при нажатии левой кнопки мышки по холсту добавляется вершина, также есть возможность перетаскивать вершины по холсту зажатием левой кнопки мышки.

Приложение ясное и удобное для пользователя, при нажатии кнопок выводятся текстовые пояснение с просьбой кликнуть на вершину, а в случае добавления/удаления ребра будет ожидаться нажатие двух вершин и после нажатия на первую вершину она будет выделяться цветом. Также кнопки “Следующий шаг” и “Результат” заблокированы, пока не запущен алгоритм, без запуска алгоритма в них нет потребности. И наоборот, при запуске алгоритма добавление/удаление вершин и ребер недоступно, так как алгоритм уже запущен на текущем графе.   
 Алгоритм предоставляет возможность пошагового отображения итераций, для этого можно воспользоваться кнопкой “Следующий шаг”, на каждом шаге будут также выведены текстовые пояснения того, какие изменения произошли. Также во время работы алгоритма будет выполняться обход графа в глубину, поэтому на каждом шаге будет подсвечиваться текущая вершина. В конце алгоритма мы должны получить разбиение вершин на компоненты сильной связности, оно будет выполнено с помощью раскрашивания вершин в разные цвета, в зависимости от того, какой компоненте они принадлежат. После завершения алгоритма будет выведена соответствующая надпись и кнопки будут разблокированы. Вершины будут сохранять раскраску до тех пор, пока граф не будет изменен, а именно пока не будет удалено/добавлено хоть одно ребро или вершина.

**1.2. Уточнение требований после...**

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

**2.1. План разработки**

1. Выбрать тему проекта. Разработать спецификацию и план разработки. Распределение ролей в бригаде.
2. После согласования спецификации и плана разработки:  
   Разработка приложения была разделена на три основные части: реализация алгоритма Косарайю, а именно класса Algorithm; реализация различных способов задания графа с помощью файла и графическим способом: добавление/удаление вершин и ребер в графе с помощью графического интерфейса, а также добавление кнопок для запуска и демонстрации шагов алгоритма (прототип приложения); объединение первых двух частей: визуализация работы алгоритма: пошаговое изображение работы алгоритма.

Итого будет происходить параллельная разработка алгоритма Косарайю и прототипа приложения различными членами бригады, ориентируясь на составленную UML-диаграмму, после чего они будут объединены для пошаговой визуализации работы алгоритма.

При создании каждой из основных частей будет происходить тестирование.

| **Задача** | **Сроки выполнения** |
| --- | --- |
| Спецификация | До 3 июля |
| План разработки | До 3 июля |
| Написание прототипа | До 5 июля |
| Сдача прототипа | 5 июля |
| Реализация алгоритма Косарайю | До 6 июля |
| Объединение двух частей проекта | До 7 июля |
| Сдача 1-ой версии | 7-8 июля |
| Исправление замечаний и ошибок, реализация дополнительного функционала. Сдача 2-ой и 3-ей версии. | 10-13 июля |

Примечание:

Сдача 1-ой версии проекта планируется на 7-8 июля. После сдачи первой версии будут исправлены замечания и ошибки, а также реализован дополнительный функционал при необходимости для сдачи второй и третьей версии проекта. Дополнительные сроки и детали разработки могут быть определены в дальнейшем в зависимости от потребностей проекта и обсуждений с командой разработчиков. Также планируется написать план тестирования, который будет сдан совместно с прототипом к 5-му июля.  
 В первой версии будет реализован сам алгоритм с возможностью пошаговой визуализации его итераций или получения сразу конечного представления результата алгоритма, а также весь необходимый функционал для добавления/удаления ребер и вершин.

Во второй версии планируется добавить кнопку “Предыдущий шаг”, для отката к прошлому шага алгоритма, при этом восстанавливается вид графа на предыдущем шаге и текстовые пояснения. Также будет добавлена возможность сохранять текущее представление графа в файл. Также будет добавлена возможность переименовывать вершины с помощью кнопки “Переименовать вершину”, которая будет ожидать нажатие на вершину и ввода нового имени вершины в соответствующее поле. Также планируется добавить возможность сохранения текущего представления графа в виде изображения. Также будут исправлены возможные ошибки и замечания, которые были не замечены в первой версии.

**2.2. Распределение ролей в бригаде**

Таблица 1. Распределение ролей.

|  | **Тулегенова А.О** | **Васильева О.М.** | **Сагидуллин Э.Р** |
| --- | --- | --- | --- |
| 1. | Создание uml диаграммы архитектуры проекта. Написание плана разработки | Реализация классов Algorithm и Graph | Создание репозитория проекта на github. Разработка спецификации. |
| 2. | Реализация классов  Panel и FileHandler (функционал, ответственный за корректное изображение самого графа и правильное хранение) | Реализация классов Panel, MouseHandler и Vertex (функционал, ответственный за удобный и интуитивно-понятный для пользователя графический интерфейс) |
| 3. | Объединение и уточнение функционала разработанных классов в соответствии с составленной диаграммой. Связь классов, ответственных за визуализацию, с классом алгоритма с помощью класса Main и соответствующих кнопок на панели экрана. | | |
| 4. | Тестирование классов  Panel и FileHandler | Тестирование классов Graph и Algorithm | Тестирование классов Panel, MouseHandler и Vertex |
| 5. | Создание дополнительного функционала для пошаговой визуализации работы алгоритма | Визуализация дополнительного функционала графического интерфейса | Разработка и корректировка графического интерфейса |
| 6. | Тестирование графического интерфейса, пошаговой визуализации работы алгоритма. | | |

**3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ**

**3.1 Основные классы**

Архитектура проекта будет состоять из следующих основных классов:

1. Graph

Данный класс необходим для хранения графа. В качестве способа представления графа был выбран подход на основе списка смежности. Этот подход является наиболее подходящим для реализации алгоритма Косарайю, который включает в себя обход графа в глубину, требующий быстрый доступ к инцидентным вершинам к данной. Список смежности будет реализован с помощью стандартной модифицируемой коллекции Mutable Map, где в качестве ключа будет идентификатор вершины графа, а в качестве значения модифицируемый список Mutable List, содержащий идентификаторы инцидентных вершин.

В этом классе должны быть реализованы методы для совершения следующих базовых операций на графе: добавление вершины, добавление ориентированного ребра, удаление вершины, удаление ребра, получение списка всех вершин, получение инцидентных вершин для данной вершины, получение транспонированного графа.

1. Algorithm

Данный класс реализует алгоритм Косарайю, который решает задачу поиска компонент сильной связности в ориентированном графе. Конструктор данного класса должен принимать объект класса Graph. В этом классе планируется реализовать методы выполняющие следующие операции: первоначальная обработка графа, построение транспонированного графа, выполнение обхода в глубину согласно измененному порядку вершин и определение сильно связных компонент. Планируется, что в качестве результата работы будет создан и возвращен объект стандартной коллекции содержащий информацию о расположении сильно связных компонент графа.

1. FileHandler

Данный класс отвечает за считывание графа из файла. Он содержит методы, позволяющие считывать информацию о вершинах и ребрах графа из текстового файла. Объект этого класса создается классом Panel для того, чтобы визуализировать входные данные.

1. Main

Данный класс является точкой входа в программу и агрегирует работу других классов. Он содержит метод main(), который запускает выполнение программы. В классе Main должны создаваться экземпляры классов Panel, Graph и Algorithm, а затем использоваться для считывания графа, выполнения алгоритма Косарайю и вывода визуализации работы алгоритма.

1. MouseHandler

Данный класс отвечает за считывание графа с помощью пользовательского ввода через графический интерфейс. Он содержит методы, которые позволяют обрабатывать действия пользователя, связанные с созданием вершин, добавлением ребер и взаимодействием с графическим представлением графа, обрабатывая соответствующие события мыши и создают вершины или ребра на основе координат, полученных от пользователя. Может хранить данные о вершине или ребре с которым взаимодействует пользователь в настоящее время.

1. Panel

Класс Panel отвечает за отрисовку графического интерфейса, включая отображение вершин и ребер графа, а также взаимодействие с пользователем. Он содержит методы для отрисовки элементов интерфейса, а именно вершин и ребер графа и текстовой информации, обработки событий и обновления графического представления при изменении структуры графа в соответствии с взаимодействием пользователя. Также хранит информацию о графе в виде списков смежности вершин для изображения вершин и ребер.

1. Vertex

Данный класс представляет вершину графа и хранит информацию о ее координатах, идентификатора, а также другую необходимую информацию. Он содержит методы для получения и установки координат вершины, а также для работы с дополнительными атрибутами, такими как цвет или метки.

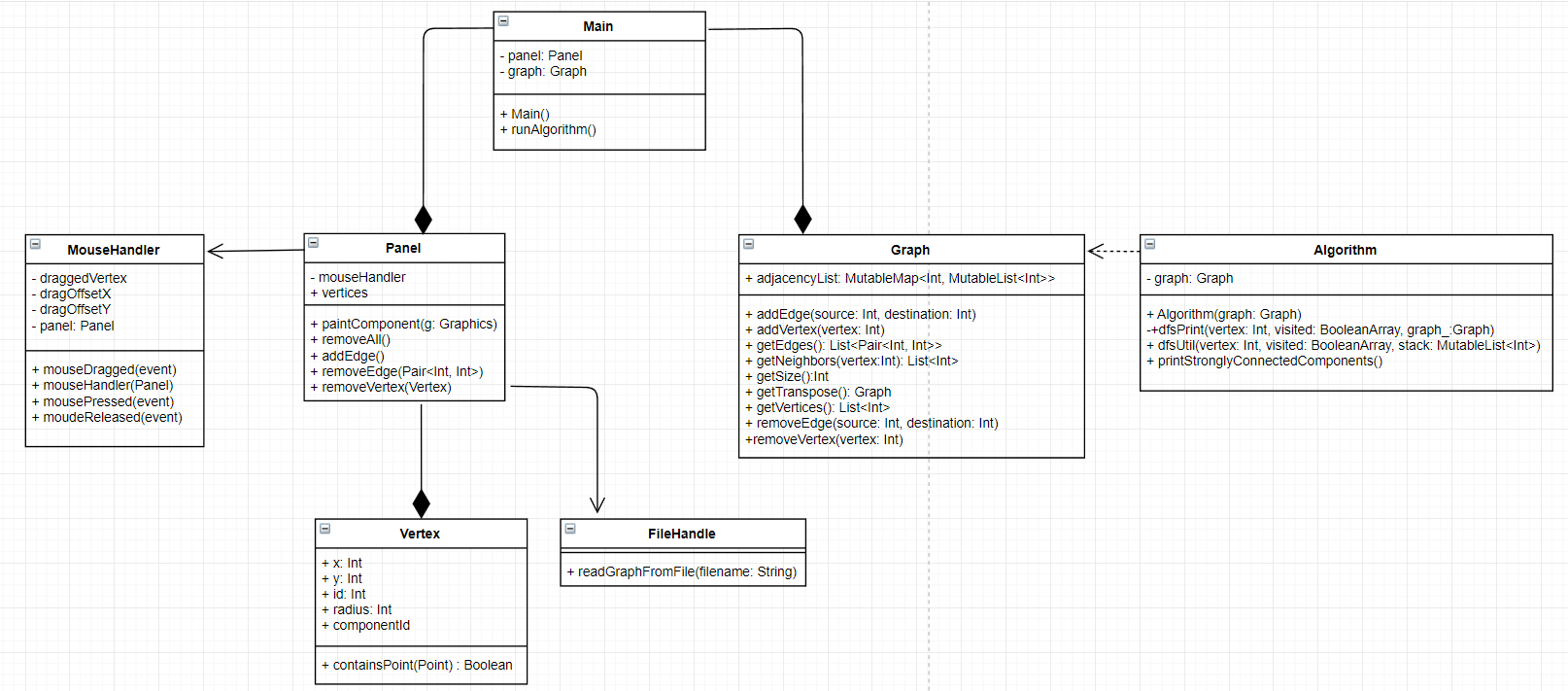


Рисунок 2. UML-диаграмма.

**3.1. Структуры данных**

**3.2. Основные методы**

**4. ТЕСТИРОВАНИЕ**

**4.1. Первый подраздел третьего раздела**

**4.2. Второй подраздел третьего раздела**

**ЗАКЛЮЧЕНИЕ**

Кратко подвести итоги, проанализировать соответствие поставленной цели и полученного результата.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

***Ниже представлены примеры библиографического описания, В КАЧЕСТВЕ НАЗВАНИЯ ИСТОЧНИКА в примерах приводится вариант, в котором применяется то или иное библиографическое описание.***

1. Иванов И. И. Книга одного-трех авторов. М.: Издательство, 2010. 000 с.

2. Книга четырех авторов / И. И. Иванов, П. П. Петров, С. С. Сидоров, В. В. Васильев. СПб.: Издательство, 2010. 000 с.

3. Книга пяти и более авторов / И. И. Иванов, П. П. Петров, С. С. Сидоров и др.. СПб.: Издательство, 2010. 000 с.

4. Описание книги под редакцией / под ред. И.И. Иванова СПб., Издательство, 2010. 000 с.

5. Иванов И.И. Описание учебного пособия и текста лекций: учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2010. 000 с.

6. Описание методических указаний / сост.: И.И. Иванов, П.П. Петров. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2010. 000 с.

7. Иванов И.И. Описание статьи с одним-тремя авторами из журнала // Название журнала. 2010, вып. (№) 00. С. 000–000.

8. Описание статьи с четырьмя и более авторами из журнала / И. И. Иванов, П. П. Петров, С. С. Сидоров и др. // Название журнала. 2010, вып. (№) 00. С. 000–000.

9. Иванов И.И. Описание тезисов доклада с одним-тремя авторами / Название конференции: тез. докл. III международной науч.-техн. конф., СПб, 00–00 янв. 2000 г. / СПбГЭТУ «ЛЭТИ», СПБ, 2010, С. 000–000.

10. Описание тезисов доклада с четырьмя и более авторами / И. И. Иванов, П. П. Петров, С. С. Сидоров и др. // Название конференции: тез. докл. III международной науч.-техн. конф., СПб, 00–00 янв. 2000 г. / СПбГЭТУ «ЛЭТИ», СПБ, 2010, С. 000–000.

11. Описание электронного ресурса // Наименование сайта. URL: http://east-front.narod.ru/memo/latchford.htm (дата обращения: 00.00.2010).

12. ГОСТ 0.0–00. Описание стандартов. М.: Изд-во стандартов, 2010.

13. Пат. RU 00000000. Описание патентных документов / И. И. Иванов, П. П. Петров, С. С. Сидоров. Опубл. 00.00.2010. Бюл. № 00.

14. Иванов И.И. Описание авторефератов диссертаций: автореф. дисс. канд. техн. наук / СПбГЭТУ «ЛЭТИ», СПБ, 2010.

15. Описание федерального закона: Федер. закон [принят Гос. Думой 00.00.2010] // Собрание законодательств РФ. 2010. № 00. Ст. 00. С. 000–000.

16. Описание федерального постановления: постановление Правительства Рос. Федерации от 00.00.2010 № 00000 // Опубликовавшее издание. 2010. № 0. С. 000–000.

17. Описание указа: указ Президента РФ от 00.00.2010 № 00 // Опубликовавшее издание. 2010. № 0. С. 000–000.

**ПРИЛОЖЕНИЕ А**

**НАЗВАНИЕ ПРИЛОЖЕНИЯ**

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