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

ОТЧЕТ

по учебной практике

Тема: Визуализация алгоритма Флойда-Уоршелла

Студент гр. 8304 _____ Рыжиков А.В.

Студент гр. 8304 _____ Мешков М.А.

Студент гр. 8382 _____ Колногоров Д.Г.

Руководитель _____ Ефремов М.А.

Санкт-Петербург

2020

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Рыжиков А.В. группы 8304

Студент Мешков М.А. группы 8304

Студент Колногоров Д.Г. группы 8382

Тема практики: Визуализация алгоритма Флойда-Уоршелла

Задание на практику:

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

Алгоритм: Флойда-Уоршелла.

Сроки прохождения практики: 29.06.2020 – 12.07.2020

Дата сдачи отчета: 00.07.2020

Дата защиты отчета: 00.07.2020

Студент	_____	Рыжиков А.В.
Студент	_____	Мешков М.А.
Студент	_____	Колногоров Д.Г.
Руководитель	_____	Ефремов М.А.

АННОТАЦИЯ

Целью практической работы является итеративная разработка программы для визуализации работы алгоритма Флойда-Уоршелла. Пользователю программы должна быть представлена возможность самостоятельно задать входные данные для алгоритма с помощью графического интерфейса, а также возможность пошагового исполнения алгоритма с просмотром промежуточных состояний алгоритма (матрицы смежности). Разработка осуществляется с использованием языка программирования Java командой из трех человек.

SUMMARY

The purpose of practical work is iterative development of the program for visualization of Floyd-Warshell algorithm. The user of the program should be provided with the possibility to set the input data for the algorithm using the GUI, as well as the possibility of step-by-step execution of the algorithm with viewing the intermediate states of the algorithm (adjacency matrix). The development is performed using Java programming language by a team of three people.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью практической работы является итеративная разработка программы для визуализации работы алгоритма Флойда-Уоршелла для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа. Пользователю программы должна быть представлена возможность самостоятельно задать входные данные для алгоритма с помощью графического интерфейса, а также возможность пошагового исполнения алгоритма с просмотром промежуточных состояний алгоритма (матрицы смежности). В конечной версии программы должна быть предусмотрена возможность сохранения и загрузки исходных данных для алгоритма из файла.

Разработка осуществляется с использованием языка программирования Java командой из трех человек. Каждому участнику команды назанчается роль и выполняемые им задачи (разработка интерфейса, логики алгоритма и тестирование программы). В результате выполнения работы должна быть представлена программа, без ошибок собирающаяся из исходных файлов и протестированная на корректность.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Требования к вводу исходных данных.

Должны быть реализованы два способа ввода исходных данных для алгоритма (граф):

- Чтение графа из файла с проверкой формата данных на корректность (в случае некорректного формата данных пользователю должна выводиться соответствующая информация).
- Создание графа в пользовательском интерфейсе (возможно ограничение на максимальное количество вершин в графе).

1.2. Требования к визуализации.

Пользовательский интерфейс должен состоять из двух частей:

- Окно для создания и модификации графа. Пользователю предоставляется возможность создавать и удалять вершины и рёбра, а также задавать вес каждого ребра.
- Окно для просмотра процесса выполнения алгоритма. Пользователю предоставляется возможность выполнения одного или нескольких шагов алгоритма и просмотра внутреннего состояния алгоритма.

Также должен быть реализован функционал выбора файла, из которого загружается граф или в который сохраняется созданный пользователем граф.

На рисунке 1 представлен макет интерфейса программы.

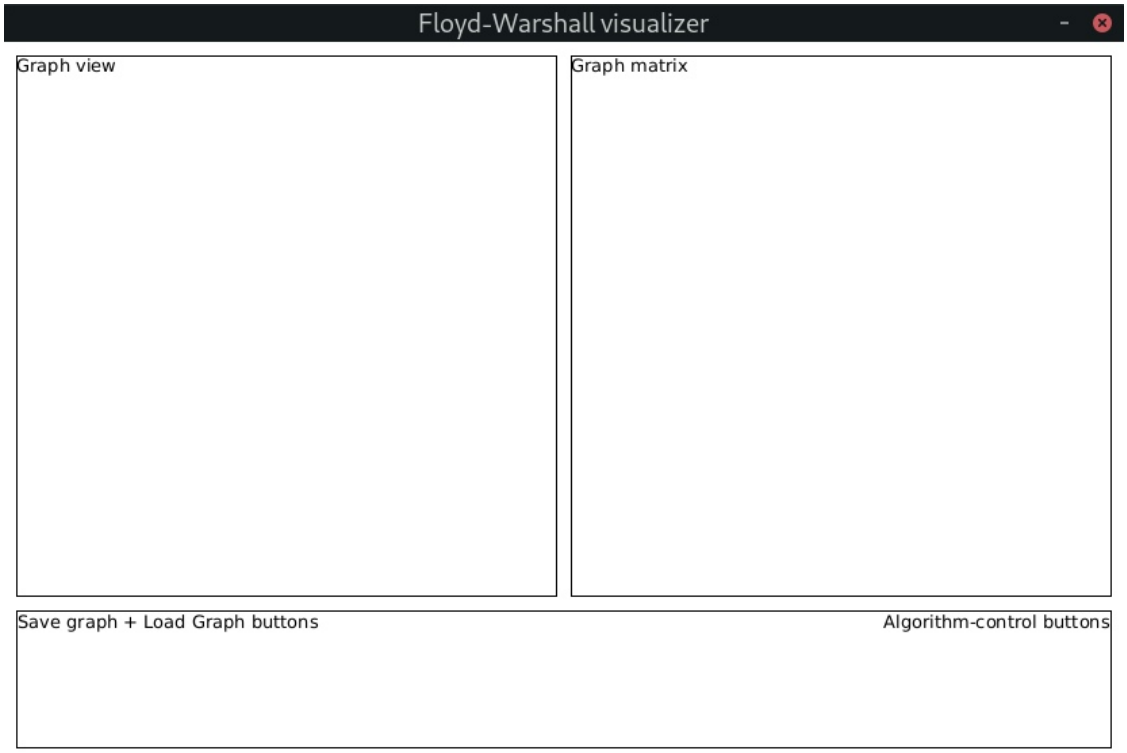

Рисунок 1 - Макет интерфейса программы

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

- Составление спецификации
- Распределение ролей
- Разработка прототипа пользовательского интерфейса
- Возможность построения графа в пользовательском интерфейсе
- Просмотр начального и конечного состояний алгоритма
- Чтение/сохранение графа в файл
- Возможность пошагового исполнения алгоритма
- Возможность возврата алгоритма к предыдущим состояниям
- Тестирование корректности работы программы
- Реализация дополнительного функционала

На рисунке 2 представлена диаграмма последовательности.

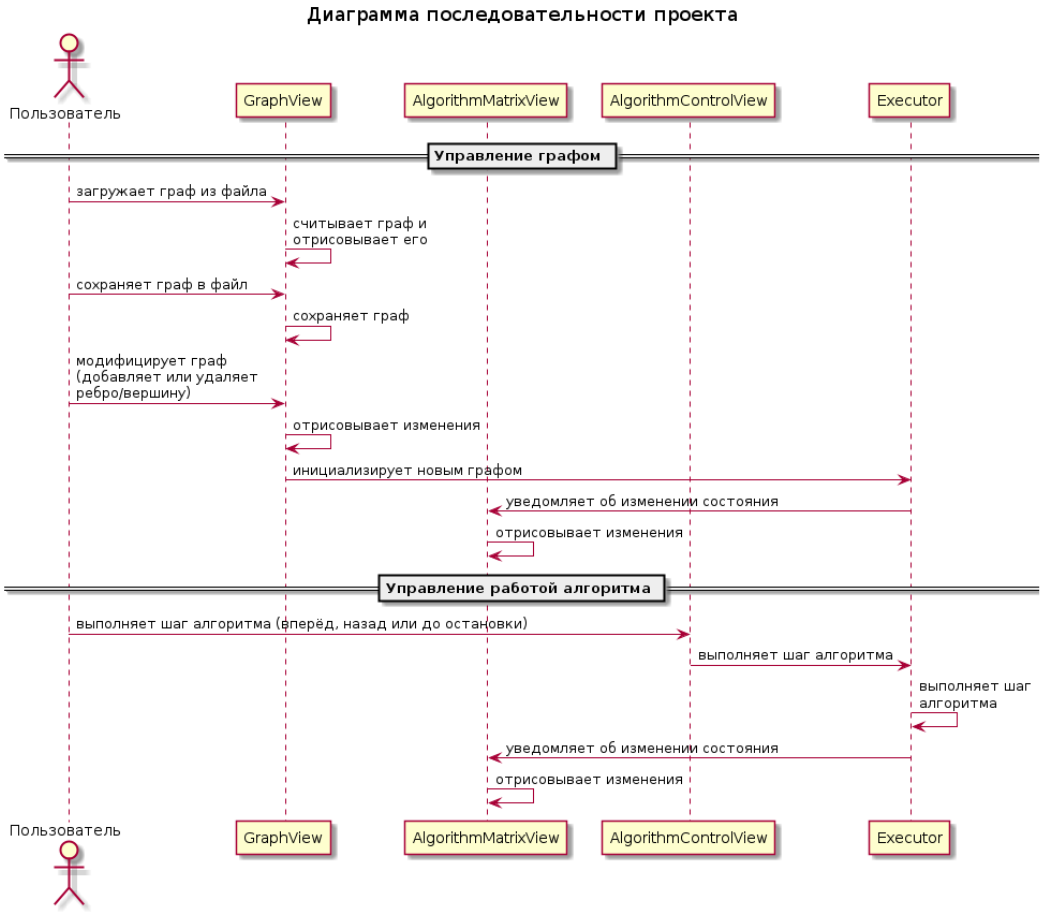

Рисунок 2 - Диаграмма последовательности

На рисунке 3 представлена диаграмма прецедентов.

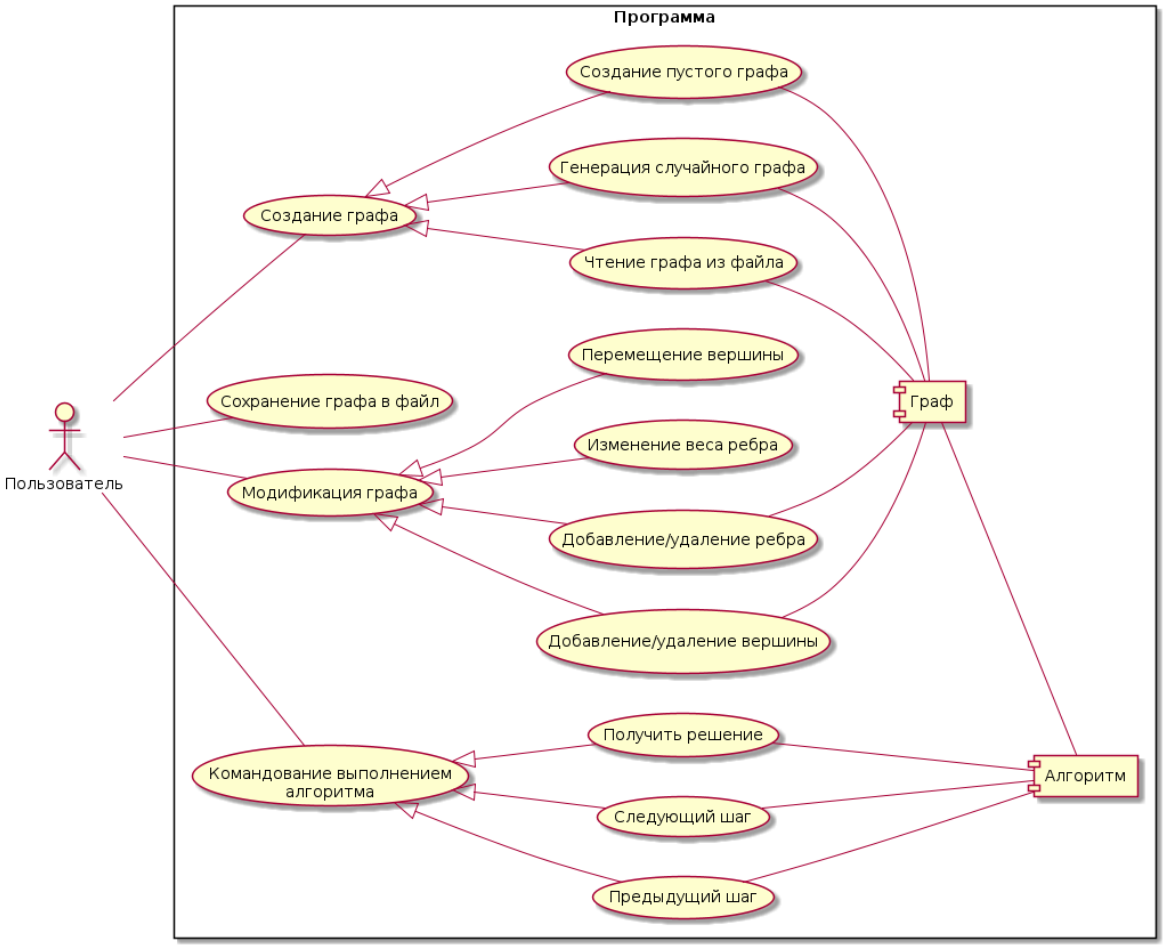

Рисунок 3 - Диаграмма прецедентов

2.2. Распределение ролей в бригаде

Рыжиков Александр - пользовательский интерфейс (ввод и отрисовка графа).

Мешков Максим - пользовательский интерфейс (визуализация работы алгоритма).

Колногоров Даниил - реализация алгоритма, тестирование.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Реализуемые классы

GraphView реализует компоненту интерфейса, отвечающую за взаимодействи пользователя с графом: создание произвольного графа, загрузка графа из файла, генерирование случайного графа. При очередном изменении графа пользователем алгоритм заново инициализируется новым графом.

AlgorithmMatrixView реализует компоненту интерфейса, отвечающую за отображение используемой алгоритмом матрицы смежности. При выполнении шага алгоритма *AlgorithmMatrixView* получает уведомления об изменении матрицы алгоритма и обновляет свое содержимое.

AlgorithmControlView реализует компоненту интерфейса, отвечающую за управление работой алгоритма. Состоит из нескольких кнопок, при нажатии на которые происходит выполнение/отмена одного или нескольких шагов алгоритма.

ExecutorInterface определяет интерфейс взаимодействия с алгоритмом. Определяет несколько групп методов: методы получения текущего состояния матрицы алгоритма, методы выполнения/отмены произвольного количества шагов, метод установки графа, с которым работает алгоритм, и метод установки наблюдателя, который будет уведомлён об изменении состояния алгоритма.

Executor реализует *ExecutorInterface.*

Gui создает экземляры классов, представляющих компоненты пользовательского интрефейса, и располагает их в окне программы.

Main является точкой входа в программу. Создаёт экземпляр *Gui* и устанавливает в него экземпляр *Executor*, а затем передаёт управление *Gui*.

GravitySimulation реализует симуляцию гравитации для вершин графа.

На рисунке 4 представлена диаграмма классов.

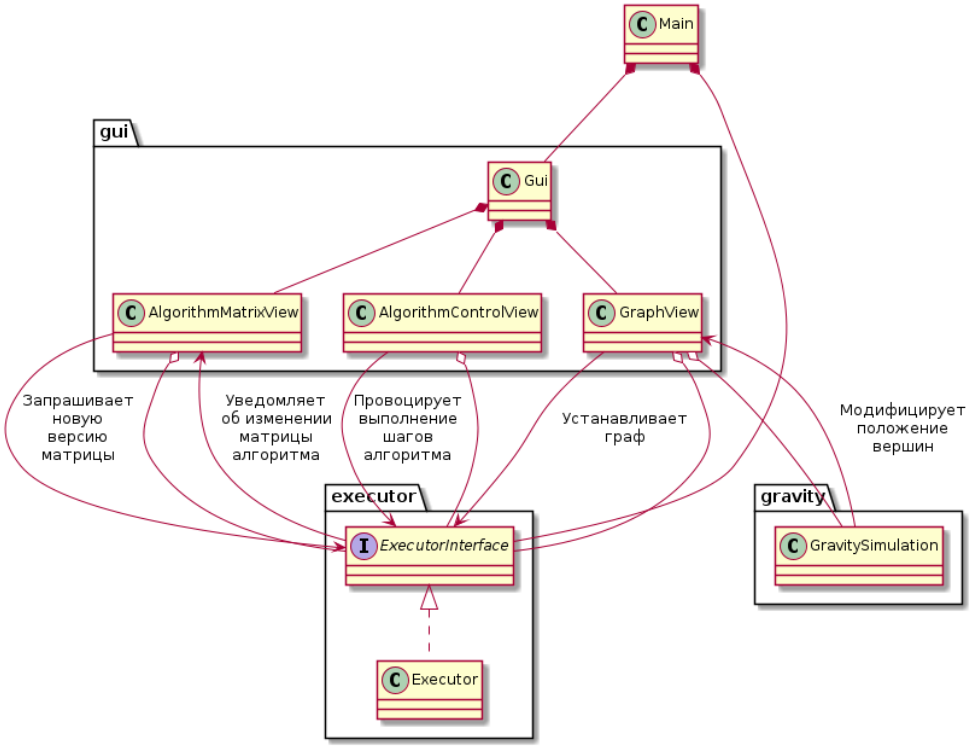

Рисунок 4 - Диаграмма классов

3.2. Основные методы

На рисунке 5 представлена диаграмма состояний программы.

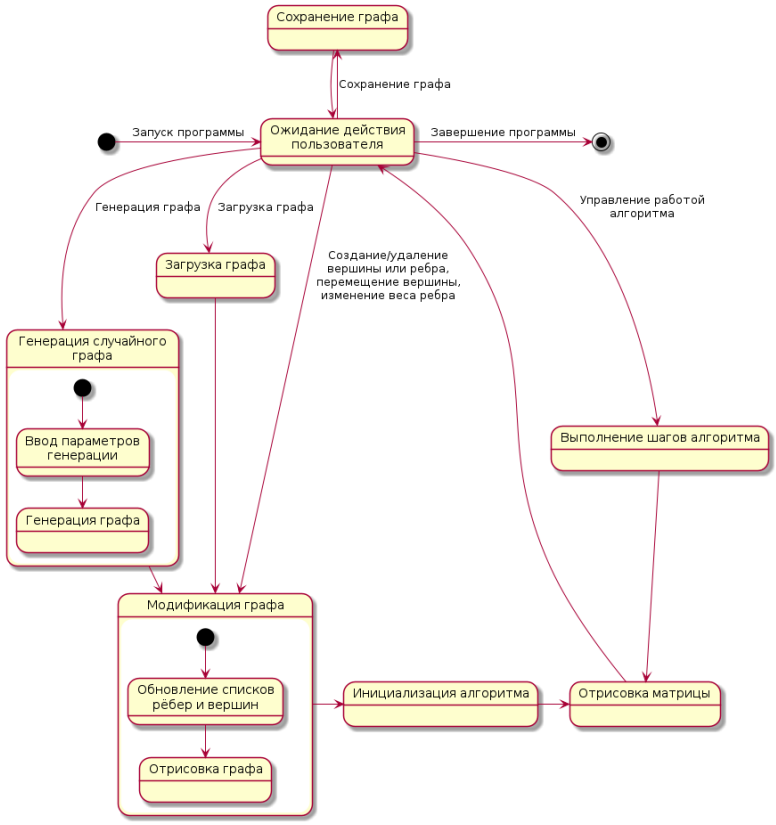

Рисунок 5 - Диаграмма состояний

4. ТЕСТИРОВАНИЕ

4.1. План тестирования программы

В процессе разработки программы планируется проведения тестирования корректности работы программы с использованием библиотеки для модульного тестирования JUnit. Основной объект тестирования - реализуемый алгоритм Флойда-Уоршелла. В таблице 1 представлен план тестирования класса Executor, реализующего алгоритм.

Таблица 1 - План тестирования

Тестируемый функционал	Исходные данные	Ожидаемый результат
Конструктор класса	Пустой граф	После инициализации экземпляра класса метод *isFinished* должен возвращать значение *True*, вызов метода *step* не должен влиять на состояние алгоритма.
Методы *setGraph* и *getPathLength*	Непустой граф (список рёбер и количество вершин)	Матрица смежности, совпадающая с матрицей смежности передаваемого графа
Методы *step*, *toEnd* и *getPathLength*	Непустой граф (список рёбер и количество вершин)	Матрица, в ячейке *(i, j)* которой находится значение минимального расстояния от вершины *i* до вершины *j*, а при отсутствии пути — значение *null*.
Методы *step*, *nextCell* и *prevCell*	Непустой граф (список рёбер и количество вершин)	Алгоритм перемещается по таблице смежности в правильном порядке: слева-направо и сверху-вниз. После посещения нижней правой ячейки матрицы алгоритм должен возвращаться к верхней левой ячейка.
Возможность возврата алгоритма к более ранним состояниям	Непустой граф (список рёбер и количество вершин)	После вызова метода *step* с положительным аргументом *n*, не превышающим количество оставшихся шагов до завершения алгоритма, и последующим вызовом метода *step* с аргументом *-n* состояние матрицы смежности не должно измениться.
Реализуемый классом паттерн проектирования «Наблюдатель»	Несколько экземпляров класса, реализующего *ExecutorInterface*	После вызов метода *step* все подписчики должны быть уведомлены об изменении состояния алгоритма.

Дополнительно проверяется корректность сборки проекта из исходных файлов в jar-архив. Программа считается корректно работающей при успешном прохождении ею тестов и корректной сборке проекта.

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