**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Доцент департамента больших данных и информационного поиска факультета компьютерных наук, к.ф.-м.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. Л. Чернышев «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г. | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Шилов «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл | RU.17701729.04.13-01 ПЗ 01-1-ЛУ |

**ПРОГРАММА ДЛЯ МОДЕЛИРОВАНИЯ ДВИЖЕНИЯ ТОЧЕК НА**

**ОРИЕНТИРОВАННОМ МЕТРИЧЕСКОМ ГРАФЕ, С УСЛОВИЕМ**

**СИНХРОНИЗАЦИИ В ВЕРШИНАХ**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.13-01 ПЗ 01-1-ЛУ**

Исполнитель

студент группы БПИ196

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А. А. Баранова /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

**Москва 2019**

УТВЕРЖДЕНRU.17701729.04.13-01 ПЗ 01-1-ЛУ

**ПРОГРАММА ДЛЯ МОДЕЛИРОВАНИЯ ДВИЖЕНИЯ ТОЧЕК НА**

**ОРИЕНТИРОВАННОМ МЕТРИЧЕСКОМ ГРАФЕ, С УСЛОВИЕМ**

**СИНХРОНИЗАЦИИ В ВЕРШИНАХ**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл | RU.17701729.04.13-01 ПЗ 01-1-ЛУ |

**Пояснительная записка**

**RU.17701729.04.13-01 ПЗ 01-1**

**Москва 2019**

**Содержание**

[1. Введение 4](#_Toc40398297)

[1.1. Наименование программы 4](#_Toc40398298)

[1.2. Краткая характеристика области применения 4](#_Toc40398299)

[2. Основания для разработки 5](#_Toc40398300)

[2.1 Документы, на основании которых ведется разработка 5](#_Toc40398301)

[2.2 Наименование темы разработки 5](#_Toc40398302)

[3. Назначение разработки 6](#_Toc40398303)

[3.1. Функциональное назначение 6](#_Toc40398304)

[3.2. Эксплуатационное назначение 6](#_Toc40398305)

[4. Требования к программе 7](#_Toc40398306)

[4.1. Требования к функциональным характеристикам 7](#_Toc40398307)

[4.1.1. Требования к составу выполняемых функций 7](#_Toc40398308)

[4.1.2. Требования к организации входных данных 8](#_Toc40398309)

[4.1.3. Требования к организации выходных данных 8](#_Toc40398310)

[4.2. Требования к интерфейсу 8](#_Toc40398311)

[4.3. Требования к надежности 9](#_Toc40398312)

[4.3.1. Требования к обеспечению надежного (устойчивого) функционирования программы 9](#_Toc40398313)

[4.3.3. Отказы из-за некорректных действий оператора 9](#_Toc40398314)

[4.4. Условия эксплуатации 9](#_Toc40398315)

[4.4.1. Климатические условия 9](#_Toc40398316)

[4.4.2. Требовaния к квалификации оператора 9](#_Toc40398317)

[4.5. Требования к составу и параметру технических средств 9](#_Toc40398318)

[4.6. Требования к информационной и программной совместимости 10](#_Toc40398319)

[4.7. Требования к маркировке и упаковке 10](#_Toc40398320)

[4.8. Требования к транспортированию и хранению 10](#_Toc40398321)

[5. Требования к программной документации 11](#_Toc40398322)

[5.1. Состав программной документации должен включать в себя следующие компоненты: 11](#_Toc40398323)

[5.2. Специальные требования к программной документации 11](#_Toc40398324)

[6. Технико-экономические показатели 12](#_Toc40398325)

[6.1. Ориентировочная экономическая эффективность 12](#_Toc40398326)

[6.2. Предполагаемая потребность 12](#_Toc40398327)

[6.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами 12](#_Toc40398328)

[7. Стадии и этапы разработки 13](#_Toc40398329)

[7.1. Необходимые стадии разработки, этапы и содержание работ 13](#_Toc40398330)

[7.2. Сроки и исполнители 14](#_Toc40398331)

[8. Порядок контроля и приемки 15](#_Toc40398332)

[8.1. Виды испытаний 15](#_Toc40398333)

[8.2. Общие требования к приемке работы 15](#_Toc40398334)

[9. Источники 16](#_Toc40398335)

[ПРИЛОЖЕНИЕ 1 17](#_Toc40398336)

[ПРИЛОЖЕНИЕ 2 18](#_Toc40398337)

# **Введение**

* 1. **Наименование программы**

Наименование программы – «Программа для моделирования движения точек на ориентированном метрическом графе, с условием синхронизации в вершинах»

«The Program for Modeling the Movement of Points on Directed Metric Graph, with the Condition of Synchronization at the Vertices».

* 1. **Документы, на основании которых ведется разработка**

Основанием для разработки является приказ декана факультета компьютерных наук И.В. Аржанцева "Об утверждении тем, руководителей курсовых работ студентов образовательной программы «Программная инженерия» факультета компьютерных наук" № 2.3-02/1112-04 от 11.12.2019.

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**
   1. **Назначение программы**
      1. **Функциональное назначение**

Функциональным назначением разрабатываемого приложения является работа с выбранным пользователем из архива ранее созданных, либо новым, построенным им в специальном редакторе, сильно связным ориентированным метрическим графом с целью моделирования движения на нем точек, с условием синхронизации в вершинах, а также получения минимального анализа поведения этих точек и визуализации происходящих на графе процессов.

* + 1. **Эксплуатационное назначение**

Программа может быть использована преподавателями, студентами или исследователями, работающими в различных областях математики и информатики при рассмотрении разнообразных задач на графах, или нейробиологии - при моделировании различных реальных процессов в нейронных сетях с использованием данной математической модели. Таким образом, продукт разработки позволит решать задачи, возникающие при составлении теоретических моделей в ходе научных исследований.

* 1. **Краткая характеристика области применения**

Программа предназначена для моделирования движения точек на сильно связных ориентированных метрических графах, с условием синхронизации в вершинах. Программный инструмент позволяет строить графы и моделировать на них движение точек двух разных типов: стандартного (синхронизация в вершинах по установленным порогам) и модели песка (синхронизация по степени вершины); получать визуализацию процесса распространения точек на графе, изображение зависимости числа точек от времени в виде графика, изображение графика распределения размеров лавин при моделировании движения песка; сохранять анимацию процесса в формате GIF-изображения.

Данная программа несет научно-образовательный характер и может использоваться исследователями, работающими в различных областях математики и информатики при рассмотрении разнообразных задач на графах или изучении абелевой модели песка, или нейробиологии – при моделировании различных процессов в нейронных сетях с использованием данной математической модели.

1. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**
   1. **Постановка задачи на разработку программы**

Разрабатываемая программа должна:

1. иметь отдельное поле для построения ориентированного метрического графа без петель, осуществлять на нем отрисовку графа и его редактирование;
2. выполнять сохранение построенного графа в файл формата .dgmm [см. Приложение 2];
3. выполнять открытие ранее созданного в этой программе графа из файла формата .dgmm [см. Приложение 2];
4. производить моделирование движения точек с условием синхронизации в вершинах по выбранному правилу, демонстрировать анимацию этого движения, предоставлять пользователю дополнительную запрошенную информацию.
   1. **Описание алгоритма и функционирования программы**
      1. **Алгоритм моделирования движения точек на ориентированном метрическом графе с условием синхронизации в вершинах**

Возможность моделирования движения точек предоставляется членами класса MovementModeling. В класс передаются сильно связный граф, тип моделирования, список выбранных дополнительных действий, а также ссылка на поверхность рисования и экземпляр GraphDrawing для отрисовки графа.

После вызова метода, запускающего движение, производится обработка переданных данных и непосредственно запуск моделирования. Далее управление передается обработчикам событий класса.

По тику основного таймера происходит обновление состояния системы и данных:

1. Обновление состояния и изображения:
2. Вызов события, оповещающего о начале обновления.
3. Исследование движущихся в данный момент точек на предмет окончания движения.
4. Исследование вершин на готовность выпустить точки.
5. Запуск новых точек.
6. Проверка состояния системы на предмет окончания движения и запуск события, оповещающего об этом, если движение окончено.
7. Обновление графиков (в случае их построения), если после действий п.1 общее количество точек на графе было изменено.
8. Сохранение текущего изображения в объект GifBitmapEncoder для возможности последующего сохранения процесса движения точек (в случае, если данная опция была выбрана).

Процесс можно приостановить нажатием кнопки Stop верхнего меню и возобновить нажатием кнопки Continue, а также полностью сбросить, кликнув по кнопке Reset.

* + 1. **Алгоритм проверки ориентированного метрического графа на сильную связность**

Проверка графа на сильную связность осуществляется с использованием упрощенного алгоритма Косарайю для поиска областей сильной связности в ориентированном графе. В процессе работы алгоритма выполняются следующие шаги:

1. С первой вершины графа запускается поиск в глубину.
2. По мере выполнения поиска в глубину заполняется массив bool[] visited. После посещения вершины элементу с ее индексом в массиве присваивается значение true.
3. По окончанию первого обхода, если visited содержит хотя бы одно значение false, метод проверки возвращает false.
4. Если при первом обходе все вершины были посещены, граф инвертируется.
5. На инвертированном графе с первой вершины вызывается поиск в глубину.
6. См. п.2.
7. По окончанию обхода, если visited содержит хотя бы одно значение false, метод проверки возвращает false.
8. Если все вершины при втором обходе были посещены, метод проверки возвращает true.

Данный алгоритм определяет, является ли ориентированный граф 𝐺 = (𝑉, 𝐸) сильно связным, за определенное время 𝑂(|𝑉| + |𝐸|).

Его сложность связана со сложностью алгоритма поиска в глубину, который, в

свою очередь, должен быть использован дважды, а также со сложностью

нахождения обратного графа. Оба эти алгоритма имеют линейную сложность и

именно поэтому Алгоритм Косарайю работает за линейное время.

* + 1. **Алгоритм парсинга введенных значений длин ребер.**

Парсинг введенных математических вражений производится путем перевода введенной строки в польскую запись (польскую нотацию) [см. Список литературы №], особенность которой состоит в том, что такая запись может быть интерпретирована без неоднозначности. В ходе парсинга выражения выполняются следующие действия:

1. На вход подается строка, содержащая введенное выражение.
2. Производится анализ расстановки скобок и перевод букв в нижний регистр.
3. Инициализируется стек, в котором будут храниться операторы, и объект StringBuilder, в который будет собираться выражение в польской нотации.
4. В цикле по позиции в строке выполняется лексический и синтаксический анализ текущей части выражения:
   1. Лексический анализ (получает на вход строку с выражением и ссылку на текущую позицию в строке):
      1. Проверяет, является ли текущий символ оператором:

Если это оператор, определяет его арность и переходит на следующую позицию, возвращает строковое представления оператора в особом формате.

* + 1. Проверяет, является ли текущий символ буквой или одной из поддерживаемых функций или констант.
       1. Передвигается далее по строке, пока она не закончилась и текущий символ – буква. В цикле добавляет в строку-результат метода текущий символ.
       2. Если полученная строка является одной из поддерживаемых функций, возвращает ее строковое представление в особом формате.
       3. Если полученная строка является одной из поддерживаемых констант, возвращает ее строковое представление в особом формате.
       4. Выбрасывает ArgumentException с сообщением «Unknown token».
    2. Проверяет, является ли текущий символ цифрой или десятичным разделителем.
       1. Если текущий символ – цифра, передвигается далее по строке, пока она не закончилась и текущий символ – цифра. В цикле добавляет в строку-результат метода текущий символ.
       2. Если текущий символ – десятичный разделитель, очищает строку-результат.
       3. Если указатель на позицию в строке не вышел за ее пределы и текущий символ – десятичный разделитель, добавляет десятичный разделитель в строку и передвигается далее по строке, пока она не закончилась и текущий символ – цифра. В цикле добавляет в строку-результат метода текущий символ.
       4. Читает научную (суффиксальную) нотацию.
       5. Возвращает полученное в строке-результате число в специальном формате.
    3. Выбрасывает ArgumentException с сообщением "Unknown token in expression".
  1. Синтаксический анализ (получает на вход строку-результат лексического анализа, ссылку на StringBuilder, в котором собирается выражение, и ссылку на стек):

По первому символу, указывающему на вид результата лексического анализа, производит следующие действия:

* + 1. Если это число, просто добавляет его в строку выражения.
    2. Если это функция или открывающая скобка – помещает в стек.
    3. Если это закрывающая скобка, вынимает элементы из стека, пока не дойдет до открывающей скобки и добавляет каждый элемент в строку-результат. Добавляет следующий элемент стека в строку-результат, если это указатель на функцию.
    4. Если это не число, функция или скобка, вынимает объекты из стека, пока их приоритетность меньше либо равна приоритетности последнего элемента стека, добавляет вынимаемые элементы в строку-результат. После помещает исходную (полученную после лексического анализа) строку в стек.
    5. Возвращает собранную строку-результат.

1. Вынимает все собранные в стеке операторы и добавляет их в результирующее выражение.
2. Производит подсчет выражения в польской нотации:
   1. Определяет стек операндов
   2. В цикле, пока позиция в строке не выходит за ее пределы производится лексический и синтаксический анализ всего выражения
      1. Лексический анализ (получает на всход строку с выражением в польской нотации и ссылку на указатель позиции в строке)
         1. Добавляет в строку-результат метода текучий символ и переходит к следующему
         2. Проходит по выражения до следующего маркера, добавляя символы в строку-результат.
         3. Возвращает полученную строку
      2. Синтаксический анализ (получает на вход ссылку на стек оперендов и результат лексического анализа)
         1. Если лексический анализ вернул операнд, добавляем его в стек.
         2. Если это операция, в зависимости от арности применяем ее к одному или двум последним операндам в стеке.
         3. Добавляет в стек результат вычислений.
   3. В результате в стеке остается единственный элемент – результат вычисления, введенного пользователем выражения.
   4. **Обоснование выбора алгоритма решения задачи.**

Алгоритмы 1-4 являются алгоритмами построения по формулам, описывающими зубчатые колеса, для них нет точных алгоритмов построения поэтому они были написаны самостоятельно с учетом механики фреймворка.

Алгоритмов для определения пересечения цилиндров не так много (мною было найдено 3) и все они отличаются сложным математическим аппаратом. В связи с этим был выбран тот, который использует более простой математический аппарат. Это позволило написать код, который легко отлаживать и поддерживать.

* 1. **Возможные взаимодействия программы с другими программами**

В целом программа работает самостоятельно. Однако для сохранения, редактирования записи и печати решения задачи необходим текстовый редактор, например Microsoft Word. Можно скопировать решение задачи из программы в графический редактор и выполнять необходимые действия по сохранению, редактированию и печати.

**ПРИЛОЖЕНИЕ 1**

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ**

|  |  |
| --- | --- |
| Класс/Структура | Назначение |
| MainWindow | Представляет главное окно приложения. Содержит обработчики пользовательских событий, в нем так же содержатся и взаимодействуют объекты других классов и поля и методы необходимые для взаимодействия этих объектов. |
| Arc | Структура, представляющая дуги нрафа. Содержит индексы начальной и конечной вершины в орграфе, |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |