Министерство просвещения Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»
Институт математики, физики, информатики и технологий
Кафедра информатики, информационных технологий
и методики обучения информатике

ПРОГРАММА ИЛЛЮСТРАЦИИ МАТЕМАТИЧЕСКИХ АЛГОРИТМОВ В КОМПЬЮТЕРНОЙ ГРАФИКЕ

Выпускная квалификационная работа
бакалавра по направлению подготовки
09.03.02 – Информационные системы и технологии

Исполнитель: студент группы ИСиТ-1601
Института математики, физики, информатики
и технологий
Камышников Б.В.

Руководитель: к.п.н., доцент кафедры ИКТО
Сардак Л.В.

Работа допущена к защите
«__» __ 2020 г.
Зав. кафедрой _____

Екатеринбург – 2020

Оглавление

ВВЕДЕНИЕ

В настоящее время всё чаще приходится сталкиваться с такой наукой как математика. Она встречается практически повсюду: в детском саду, школе, колледже, вузах, работе и т.д. И иногда приходится решать сложные задачи, которые требуют особого внимания. Задачи с числами обычно считаются наиболее сложными, поскольку они требуют определенной логики для их решения, особенно это касается тех задач, которые состоят из нескольких компонентов, и их решение может занимать от нескольких минут до пары часов.

Чтобы люди могли себе сэкономить время и нервы, были придуманы самые разные формулы, которые позволяли находить самые разнообразные величины. С появлением персональных компьютеров вести расчеты стало еще проще, чем было. Появились программы-калькуляторы, которые позволяли находить нужный результат за пару кликов мыши, позже появлялись самый различный инструментарий, состоявший из формул, где можно вести расчеты еще быстрее. Появлялись и другие программы, но уже выполняющие немного другие функции.

Для того чтобы можно было создать такую программу, необходимо обладать отличной логикой, чтобы не нарушалась работа программы. Такая логика всегда строилась на каких-то формулах или операциях, связанных с математикой.

Таким образом, можно прийти к выводу, что у программ всегда есть определенный набор алгоритмов[26]. В графических программах они также присутствуют и выполняют свою роль.

Представляется актуальным провести разработку такого приложения. Предмет разработки – приложение, написанное для персональных компьютеров с участием математических алгоритмов, присутствующих в компьютерной графике.

Цель работы – разработать программу, отображающую явные признаки алгоритмов в компьютерной графике.

Задачи:

1. Рассмотреть математические алгоритмы, которые используются в созданиях программ.
2. Выбрать оптимальный язык программирования для написания программы.
3. Сформулировать техническое задание на разработку приложения.
4. Разработать программу.

ГЛАВА 1. АНАЛИЗ МАТЕМАТИЧЕСКИХ АЛГОРИТМОВ В КОМПЬЮТЕРНЫХ ПРОГРАММАХ.

1.1 Реализуемые алгоритмы в программах

Алгоритм – совокупность заданных правил решения задач или набор инструкций, описывающих порядок действий исполнителя для решения каких-либо задач[1].

Можно выделить алгоритмы управляющие и вычислительные. Семантика управляющих алгоритмов сводится к выдаче необходимых управляющих воздействий, либо в заданные моменты времени, либо в качестве реакции на внешние события (в таком случае, управляющий может оставаться корректным при бесконечном выполнении, в отличие от вычислительного алгоритма). Вычислительные алгоритмы, по своей сути, преобразуют некоторые стартовые (начальные) данные в выходные, реализуя вычисление некоторой функции.

Теперь рассмотрим свойства алгоритма. Различные определения алгоритма в явной или неявной форме содержат ряд следующих общих требований:

- Дискретность – алгоритм должен представлять процесс решения задач как поэтапное выполнение некоторых шагов. При этом для реализации каждого шага алгоритма требуется конечный отрезок времени, другими словами преобразование начальных данных в результат осуществляется во времени дискретно;
- Понятность – алгоритм должен включать только те команды, которые доступны исполнителю и входят в состав его команд;
- Универсальность (массовость). Алгоритм должен быть применен к разным наборам стартовых (начальных) данных;
- Определенность (детерминированность). Следующий шаг работы в каждый момент времени однозначно определяется состоянием

системы. Другими словами, алгоритм выдает один и тот же ответ (результат) для одних и тех же исходных данных. В современной трактовке у различных реализаций одного и того же алгоритма должен быть изоморфный граф. В другом случае, существуют вероятностные алгоритмы, в которых следующий шаг работы зависит от текущего состояния системы и генерируемого случайного числа. Однако при включении метода генерации случайных чисел в список исходных данных вероятностный алгоритм становится подвидом обычного;

- Конечность (завершаемость) – в более узком понимании алгоритма как математической функции, при правильном задании начальных данных алгоритм должен завершать работу и выдавать результат за определенное число шагов. Дональд Кнут процедуру, которая удовлетворяет всем свойствам алгоритма не включает завершаемость за конечное время. В этом случае метод вычисления (алгоритм) определяет частичную функцию. Для вероятностных алгоритмов завершаемость как правило означает, что алгоритм выдает результат с вероятностью 1 для любых правильно заданных начальных данных (другими словами может в некоторых случаях не завершиться, но вероятность этого должна быть равна 0);
- Результативность – завершение алгоритма, дающий определенный результат.

В настоящий момент существует множество программ, которые содержат в себе математические алгоритмы. Проведем анализ некоторых из них.

Приложение калькулятор для Windows[14].

Калькулятор – программа от Microsoft Windows, предназначенная для выполнения несложных вычислительных операций: сложение, вычитание,

умножение и деление. В нем также предусмотрены и другие, более сложные операции для инженерных и статистических вычислений.

Режимы работы:

1. Обычный режим
   - В обычном режиме представлены функции процентов, квадратного корня, обратной величины;
   - Есть манипуляции с памятью, предоставляется для этого ячейка;

Рис. 1. Обычный режим

2. Инженерный режим

   Вместе с обычным доступны:

   - Сдвиг влево и вправо;

- Вычисление остатка от деления;
- Побитовые операции;
- Перевод долей градуса в минуты и секунды, вычисление факториала, а также пи-функции (является обобщением факториала и вычисляется через гамма-функцию);
- Гиперболические и тригонометрические функции, обратные им функции, которые доступны через «Inv», десятичный и натуральный логарифмы, экспонента, возведение в степень и извлечение корня;
- Группировка операций и режимы отображения с плавающей и фиксированной точкой;
- Расчеты в градусах, радианах и градах;
- Недесятичные системы счисления (переключатель разрядности обрабатываемых данных 8, 4, 2, 1 байт).

Рис. 2. Инженерный режим

3. Режим «Программист»
    - Логические операции: AND, NOT, XOR, OR;
    - Логические сдвиги вправо и влево;
    - Возможность обработки данных в шестнадцатеричной, восьмеричной и двоичной системы счислениях.

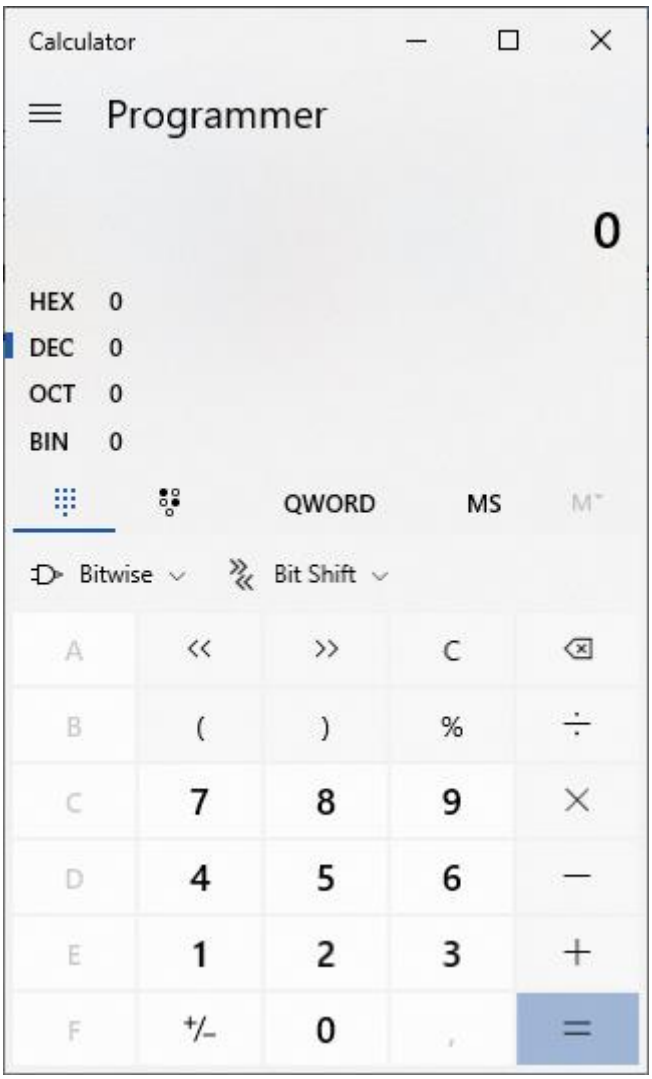

Рис. 3. Режим «Программист»

В приложении «Калькулятор» описаны множество от самых простых алгоритмов, до технически сложных. Все они применяются в 3 разных режимах, которые каждый по-своему уникальны.

Далее, стоит рассмотреть программу Adobe Photoshop CS6[17][21]. В ней применяются немного другие операции, но они связаны напрямую с работой с графикой. Основными алгоритмами можно считать методы интерполяции[16], которые сейчас рассмотрим.

Интерполяция функции нескольких переменных. К ней относятся:

- Билинейная интерполяция;
- Бикубическая интерполяция.

Кроме всего этого, бикубическую интерполяцию можно разбить еще на несколько разделов, которые также выполняют различные функции.

Бикубическая интерполяция

- Бикубическая автоматическая;
- Бикубическая – четче;
- Бикубическая – глаже;
- Бикубическая для плавных градиентов.

Сюда же можно отнести такой способ интерполяции как «***По соседним пикселям***».

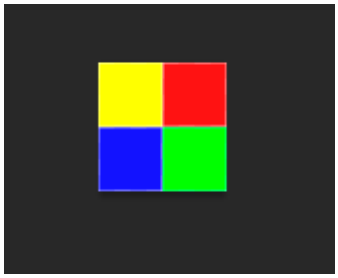

Рис. 4. Исходное изображение

Билинейная интерполяция – такой метод добавляет новые пиксели, находя средние значения окружающих пикселей.

Но в этом способе существует и один главный недостаток. С использованием билинейной интерполяции при масштабировании изображений – при увеличении в x раз исходного изображения размерами w на h пикселей в результате будет получено изображение размером не x умноженное на w (ширина) и x умноженное на h (высота), а (x(w - 1) + 1) «ширина» и (x(h - 1) + 1) «высота» пикселей. Это происходит из-за того, что в исходном изображении по горизонтали имеется w точек, а именно (w - 1) смежных пар. При увеличении изображения в x раз между каждой парой основных точек вставляется по (x - 1) дополнительных точек, а именно при увеличении вдвое между основными точками вставляется еще по одной, при увеличении втрое – по две и так далее. В результате ширина

результирующего изображения будет равна сумме количества основных и дополнительных точек: w + (w – 1)(x - 1) = x(w - 1) + 1. То есть, для пикселей по границам изображения в каждой строке и столбце исходного изображения не находится пары, с которой можно было бы провести интерполирование.

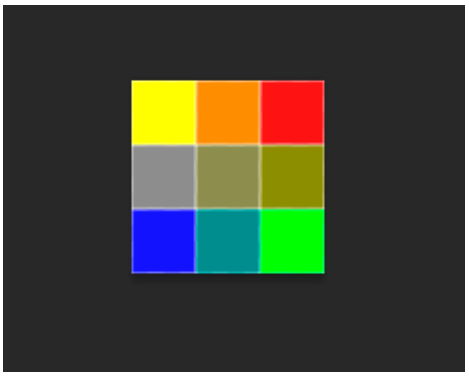

Рис. 5. Изображение, подвергшееся билинейной интерполяции

Бикубическая интерполяция – в этом случае бикубической интерполяции значения функции в искомой точке вычисляется через её значения в 16 соседних точках, расположенных в вершине квадратов плоскости x, y.

При использовании приведенных ниже формул для реализации бикубической интерполяции следует помнить, что значения x и y являются относительными, а не абсолютными. Например, для точки с координатами (300.7, 300.9) x = 0.7, а y = 0.8. Для получения относительных значений координат необходимо округлить вещественные координаты вниз, и вычесть полученные числа из вещественных координат.

$$f(y,x) = b_1 f(0,0) + b_2 f(0,1) + b_3 f(1,0) + b_4 f(1,1) + b_5 f(0,-1) + b_6 f(-1,0) + b_7 f(1,-1) + b_8 f(-1,1) + b_9 f(0,2) + b_{10} f(2,0) + b_{11} f(-1,-1) + b_{12} f(1,2) + b_{13} f(2,1) + b_{14} f(-1,2) + b_{15} f(2,-1) + b_{16} f(2,2),$$ где

$$b_1 = \frac{1}{4}(x-1)(x-2)(x+1)(y-1)(y-2)(y+1),$$

$$b_2 = -\frac{1}{4}x(x+1)(x-2)(y-1)(y-2)(y+1),$$

$$b_3 = -\frac{1}{4}y(x-1)(x-2)(x+1)(y+1)(y-2),$$

$$b_4 = \frac{1}{4}xy(x+1)(x-2)(y+1)(y-2),$$

$$b_5 = -\frac{1}{12}x(x-1)(x-2)(y-1)(y-2)(y+1),$$

$$b_6 = -\frac{1}{12}y(x-1)(x-2)(x+1)(y-1)(y-2),$$

$$b_7 = \frac{1}{12}xy(x-1)(x-2)(y+1)(y-2),$$

$$b_8 = \frac{1}{12}xy(x+1)(x-2)(y-1)(y-2),$$

$$b_9 = \frac{1}{12}x(x-1)(x+1)(y-1)(y-2)(y+1),$$

$$b_{10} = \frac{1}{12}y(x-1)(x-2)(x+1)(y-1)(y+1),$$

$$b_{11} = \frac{1}{36}xy(x-1)(x-2)(y-1)(y-2),$$

$$b_{12} = -\frac{1}{12}xy(x-1)(x+1)(y+1)(y-2),$$

$$b_{13} = -\frac{1}{12}xy(x+1)(x-2)(y-1)(y+1),$$

$$b_{14} = -\frac{1}{36}xy(x-1)(x+1)(y-1)(y-2),$$

$$b_{15} = -\frac{1}{36}xy(x-1)(x-2)(y-1)(y+1),$$

$$b_{16} = \frac{1}{36}xy(x-1)(x+1)(y-1)(y+1).$$

Подобным образом можно использовать и интерполяции более высокого порядка, вычисляя значения функции по соседним $4k^2$ точкам.

Бикубическая автоматическая – данный способ автоматически определяет метод интерполяции из ниже перечисленного списка способов наиболее подходящий для увеличения или уменьшения изображения.

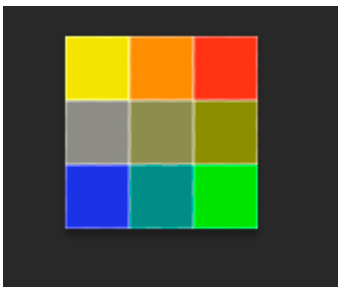

Рис. 6. Изображение, подвергшееся бикубической (автоматической) интерполяции

Бикубическая (четче) – один из методов для уменьшения изображения размера изображения на основе бикубической интерполяции с повышенной резкостью. Такой метод сохраняет детали изображения подвергнутого интерполяцией.

Рис. 7. Изображение, подвергшееся бикубической (четче) интерполяции

Бикубическая (глаже) – еще один метод, основанный на бикубической интерполяции. Позволяет получить более гладкие детали изображения во время его увеличения.

Рис. 8. Изображение, подвергшееся бикубической (глаже) интерполяции

Бикубическая (для плавных градиентов) – более медленный, но и более точный метод, основанный на анализе значений цвета окружающих

пикселей. За счет использования более сложных вычислений бикубическая интерполяция дает плавные цветовые переходы, чем интерполяция *по соседним пикселям* или *билинейная* интерполяция.

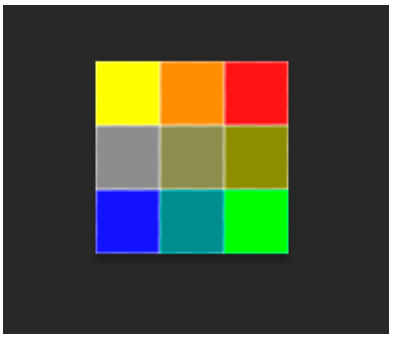

Рис. 9. Изображение, подвергшееся бикубической (для плавных градиентов) интерполяции

По соседним пикселям – быстрый, но менее точный метод, который повторяет пиксели изображения. Этот метод сохраняет четкие края и может создать файл уменьшенного размера в иллюстрациях, содержащих не сглаженные края. Однако такой метод может создать зубчатые края, которые стану заметными при искажении или масштабировании изображения.

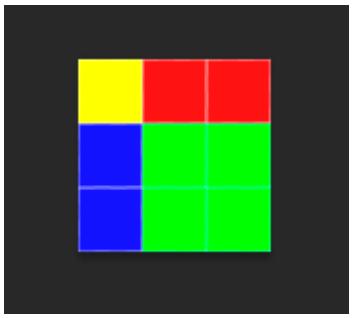

Рис. 10. Изображение, подвергшееся «По соседним пикселям» интерполяции

Исходя из вышесказанного, можно убедиться, что алгоритмы в компьютерной графике точно также используются, как и в обычных программах. Чаще всего, они не только взаимодействуют с инструментами, которые присутствуют в программе Adobe Photoshop CS6, но и также влияют на представленное изображение в рабочей среде.

1.2 Обзор языков программирования

На сегодняшний день в нашем мире известно довольно большое количество языков программирования. Каждый из них по-своему уникален и дает программисту в определенных случаях преимущество там, где его нет в другом. Сейчас одними из самых популярных языков программирования являются:

- Python;
- Java;
- C++.

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен[19]. В то же время стандартная библиотека включает большой объём полезных функций[29].

Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. Есть реализация интерпретатора для JVM с возможностью компиляции, CLR, LLVM, другие независимые реализации. Проект PyPy использует JIT-компиляцию, которая значительно увеличивает скорость выполнения Python-программ.

Преимущества и недостатки языка Python:

1. Python - интерпретируемый язык программирования. С одной стороны, это позволяет значительно упростить отладку программ, с другой - обуславливает сравнительно низкую скорость выполнения.
2. Динамическая типизация. В python не надо заранее объявлять тип переменной, что очень удобно при разработке.
3. Хорошая поддержка модульности. Вы можете легко написать свой модуль и использовать его в других программах.
4. Встроенная поддержка Unicode в строках. В Python необязательно писать всё на английском языке, в программах вполне может использоваться ваш родной язык.
5. Поддержка объектно-ориентированного программирования. При этом его реализация в python является одной из самых понятных.
6. Автоматическая сборка мусора, отсутствие утечек памяти.
7. Интеграция с C/C++, если возможностей python недостаточно.
8. Понятный и лаконичный синтаксис, способствующий ясному отображению кода. Удобная система функций позволяет при грамотном подходе создавать код, в котором будет легко разобраться другому человеку в случае необходимости. Также вы сможете научиться читать программы и модули, написанные другими людьми.
9. Огромное количество модулей, как входящих в стандартную поставку Python 3, так и сторонних. В некоторых случаях для написания программы достаточно лишь найти подходящие модули и правильно их скомбинировать. Таким образом, вы можете думать о составлении программы на более высоком уровне, работая с уже готовыми элементами, выполняющими различные действия.
10. Кроссплатформенность. Программа, написанная на Python, будет функционировать совершенно одинаково вне зависимости от

того, в какой операционной системе она запущена. Отличия возникают лишь в редких случаях, и их легко заранее предусмотреть благодаря наличию подробной документации.

C++ – компилируемый, статически типизированный язык программирования общего назначения[23].

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее, C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

Достоинства:

- Высокая совместимость с языком Си;
- Вычислительная производительность;
- Поддержка различных стилей программирования: структурное, объектно-ориентированное, обобщённое программирование, функциональное программирование, порождающее метапрограммирование;
- Автоматический вызов деструкторов объектов (в порядке обратном вызову конструкторов) упрощает и повышает

надёжность управления памятью и другими ресурсами (открытыми файлами, сетевыми соединениями, т. п.);

- Перегрузка операторов;
- Шаблоны (дают возможность построения обобщённых контейнеров и алгоритмов для разных типов данных);
- Возможность расширения языка для поддержки парадигм, которые не поддерживаются компиляторами напрямую;
- Доступность. Для C++ существует огромное количество учебной литературы, переведённой на всевозможные языки.

Недостатки:

- Плохо продуманный синтаксис сужает спектр применимости языка;
- Язык не содержит многих важных возможностей;
- Язык содержит опасные возможности;
- Производительность труда программистов на языке оказывается неоправданно низка;
- Громоздкость синтаксиса;
- Тяжелое наследие;
- Необходимость следить за памятью.

Java — строго типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle)[13][18][27]. Разработка ведётся сообществом, организованным через Java Community Process, язык и основные реализующие его технологии распространяются по лицензии GPL. Права на торговую марку принадлежат корпорации Oracle[28].

Программы на Java транслируются в байт-код Java, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор[12].

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Часто к недостаткам концепции виртуальной машины относят снижение производительности. Ряд усовершенствований несколько увеличил скорость выполнения программ на Java:

- применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде;
- обширное использование платформенно-ориентированного кода (native-код) в стандартных библиотеках;
- аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология Jazelle, поддерживаемая некоторыми процессорами архитектуры ARM).

Из выше перечисленных языков программирования было решено использовать такой язык как Java из-за его простого синтаксиса, а также кроссплатформенности.

1.3 Формализованное описание технического задания

1. Общие сведения.

1.1. Название организации-заказчика.

ФГБОУ ВО «УрГПУ»

1.2. Название продукта разработки (проектирования).

«Программа иллюстрации математических алгоритмов в компьютерной графике»

1.3. Назначение продукта.

Данная программа может быть полезна как школьникам, так и студентам. В ней пользователь может создать график функции и посмотреть, как он будет выглядеть.

1.4. Плановые сроки начала и окончания работ.

В соответствии с планом выполнения ВКР (01.09.2019 – 19.05.2020).

2. Характеристика области применения продукта.

2.1. Процессы и структуры, в которых предполагается использование продукта разработки.

Предполагается, что данная программа будет использоваться в общеобразовательных организациях и вузах.

2.2. Характеристика персонала (количество, квалификация, степень готовности)

Разработчик – знание Java, опыт работы не менее 1 года. Опыт разработки приложений в интегрированной среде разработки Eclipse[25].

Пользователь – должен обладать базовыми навыками обращения с компьютером, должен знать, как запускается программа, и должен быть знаком с приложениями на языке Java.

3. Требования к продукту разработки.

3.1. Требования к продукту в целом.

Требуется наличие операционной системы с установленной Java Runtime Environment (JRE). Запуск приложения будет осуществляться при выполнении файла *.jre. Также требуется для работы с программой клавиатура и мышь.

3.2. Аппаратные требования.

Персональный компьютер с клавиатурой и мышью.

3.3. Указание системного программного обеспечения (операционные системы, браузеры, программные платформы и т.п.).

Поддерживаются следующие операционные системы: Windows, Mac OS X (10.7.3 и выше), Linux, Solaris с установленной Java.

3.4. Указание программного обеспечения, используемого для реализации.

Eclipse (Neon.3 Release 4.6.3), Java 8, BPwin[22].

3.5. Для сетевых систем – особенности реализации серверной и клиентской частей.

Не предусмотрено.

3.6. Форматы входных и выходных данных

Форматом входных данных являются числа и тригонометрические функции. Выходными данными является график функции в прямоугольной системе координат на плоскости.

3.7. Источники данных и порядок их ввода в систему (программу), порядок вывода, хранения.

В появившемся окне требуется ввести функцию для создания графика, а также пределы X и Y. Источником вывода данных является выводимое окно после нажатия на кнопку “Выполнить”.

3.8. Порядок взаимодействия с другими системами, возможности обмена информацией.

Не предусмотрено.

3.9. Меры защиты информации.

Не предусмотрено.

4. Требования к пользовательскому интерфейсу.

4.1. Общая характеристика пользовательского интерфейса.

Всё выполнено в светлых тонах для удобства вводимых данных и отображения графиков.

4.2. Размещение информации на экране, дизайн экрана.

Дизайн экрана и размещение элементов должно советовать представленным макетам:

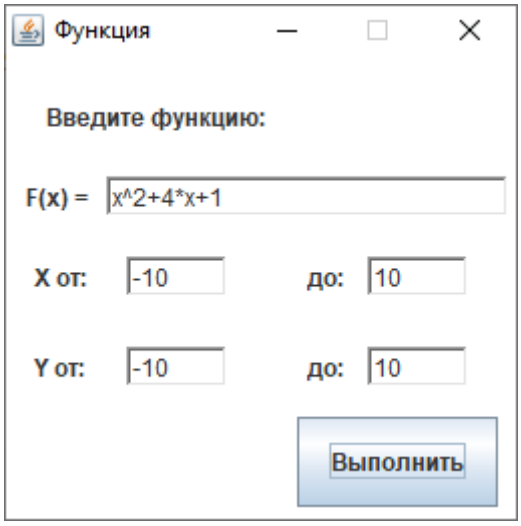

Рис. 11.Вводимые данные

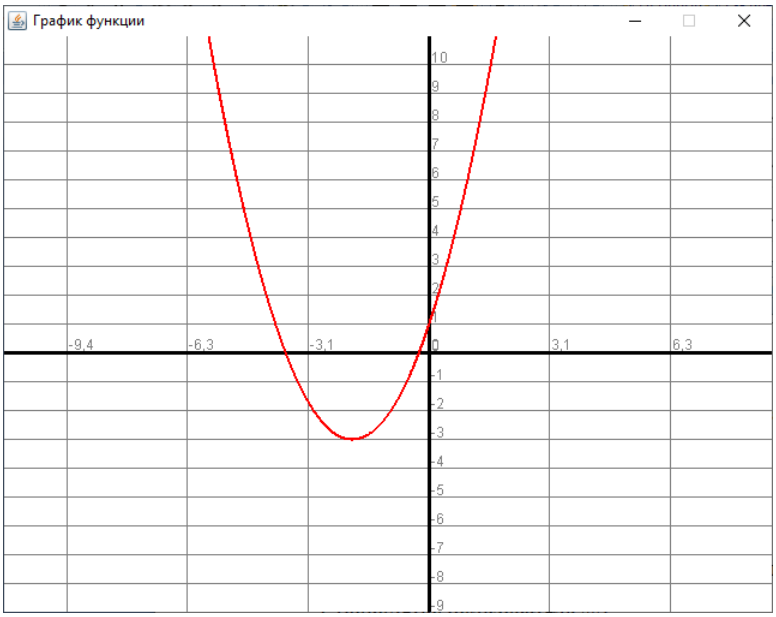

Рис. 12. Получаемый график функции

4.3. Особенности ввода информации пользователем, представление выходных данных.

Пользователь вводит информацию о графике, задавая функцию, а также пределы оси абсцисс и ординат. Выходные данные представляются в виде графика функции.

5. Порядок сдачи-приемки продукта.

В соответствии с графиком выполнения выпускной квалификационной работы.

Глава 2. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ С МАТЕМАТИЧЕСКИМИ АЛГОРИТМАМИ

2.1 Функциональные модели приложения

Для реализации данного приложения была разработана функциональная модель IDEF0. Рассмотрим её подробно.

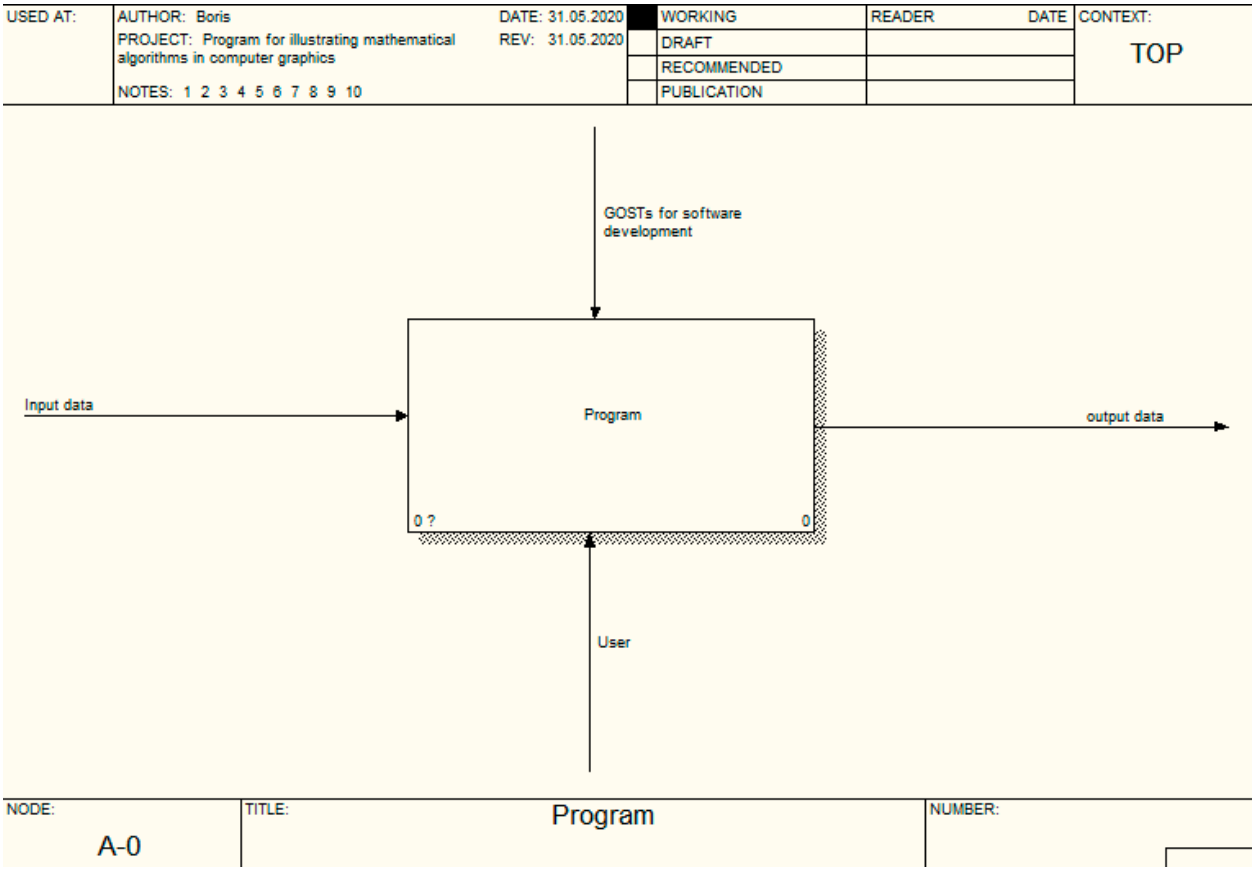

Рис. 13. Функциональная диаграмма IDEF0

На уровне A-0 можно увидеть функциональный блок “Program” (это программа), которая принимает данные на вход “Input data” в виде заданной функции, и установки ограничений графика для их обработки в дальнейшем. В качестве выходных данных “Output data” служит график функции, который был задан по указанной функции с выставленными ограничениями. Механизмом к этой схеме будет служить пользователь “User”, которой является ключевым составляющим для выполнения данной программы. И, наконец, в качестве управляющих механизмов выступает ГОСТы по

разработке программного обеспечения “GOSTs for software development”. Далее стоит рассмотреть декомпозицию блоков.

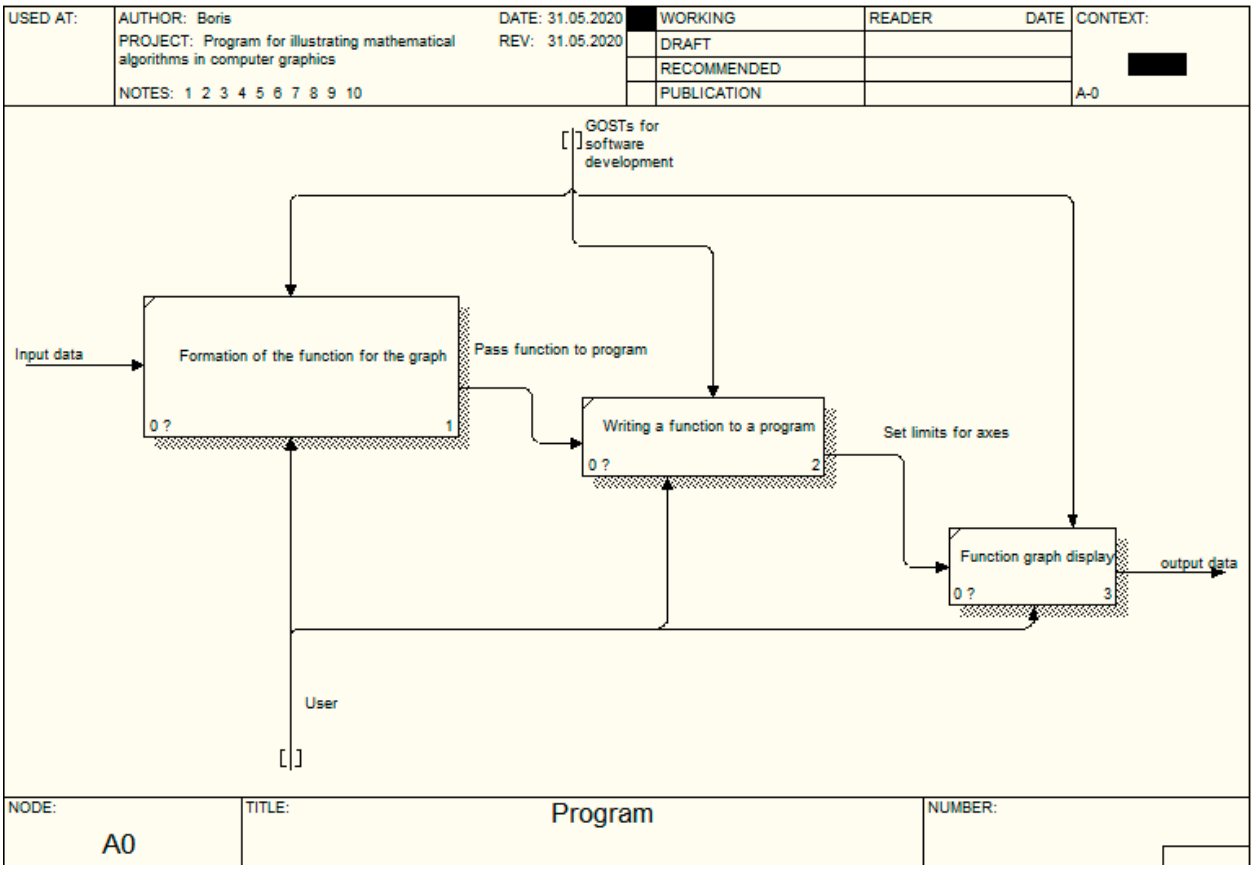

Рис. 14. IDEF0 декомпозиция блоков

На диаграмме декомпозиции блока A0 имеются 3 функциональных блока:

- Formation of the function for the graph (Формирование функции для графа);
- Writing a function to a program (Написание функции в программу);
- Function graph display (Отображение функции на экране).

Блок “Formation of the function for the graph” содержит в себе:

- На входе: Input data «Входные данные»;
- Управление: GOSTs for software development «ГОСТы по разработке программного обеспечения»;
- Механизм: User «Пользователь»;

- На выходе: Pass function to program «Передача функции в программу».

Блок “Writing a function to a program” содержит:

- На входе: Pass function to program «Передача функции в программу»;
- На выходе: Set limits for axes «Установка ограничений для осей»;
- Управление: GOSTs for software development «ГОСТы по разработке программного обеспечения»;
- Механизм: User «Пользователь».

Блок “Function graph display” имеет:

- На входе: Set limits for axes «Установка ограничений для осей»;
- Управление: GOSTs for software development «ГОСТы по разработке программного обеспечения»;
- Механизм: User «Пользователь».
- На выходе: Output data «Выходные данные».

2.2 Создание приложения

Теперь, когда мы определились с языком программирования, нужно определить, в какой среде будет вестись разработка программы. Наиболее предпочтительным вариантом будет Eclipse. Eclipse – свободная интегрированная среда разработки модульных кроссплатформенных приложений. Является, в настоящий момент, одной из самых популярных IDE (Integrated Development Environment), поддерживает и другие язык программирования. Простая и удобная в использовании.

В нашей программе будет использоваться встроенная библиотека AWT (Abstract Window Toolkit)[20]. AWT – платформо-независимая библиотека графического интерфейса. Далее вместе с ней будет еще одна библиотека под названием Swing[30]. Swing – также является платформо-независимой библиотекой, которая представляет более гибкие интерфейсные компоненты.

Её компоненты поддерживают некоторые специфические подключаемые виды и поведения, которые изменяют интерфейс программы, благодаря которой возможна адаптация к графическому интерфейсу платформы.

Перейдем непосредственно в нашу среду.

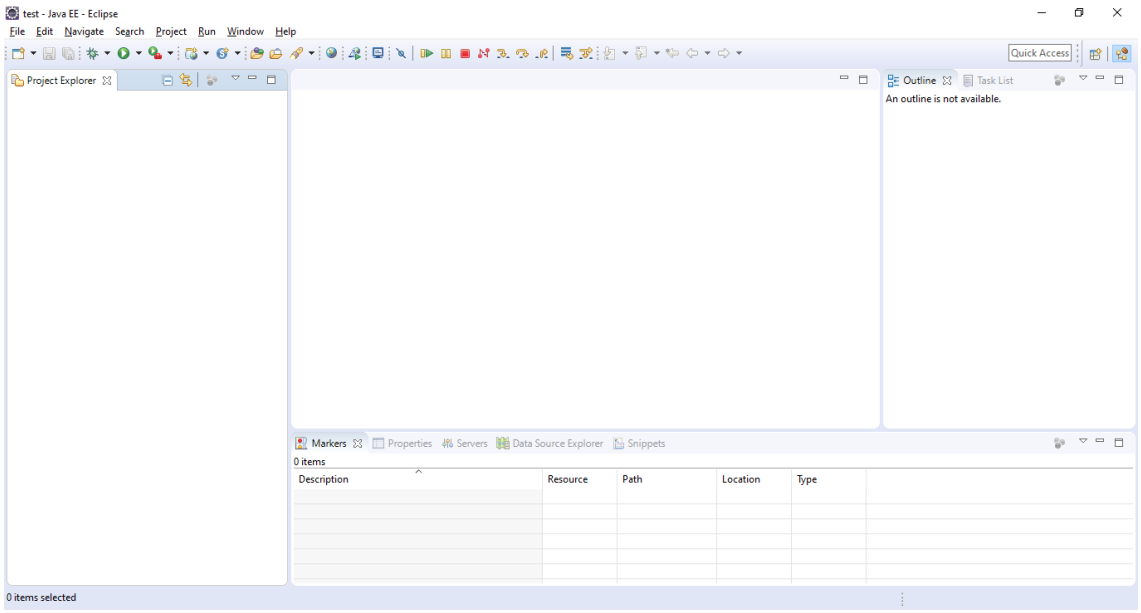

Рис. 15. Eclipse IDE

В окне слева под названием “Project Explorer” нажимаем правую кнопку мыши, выбираем New, затем Project. В открывшемся окне выделяем Java Project и нажимаем next. В строке Project name задаем имя нашего проекта. Здесь можно задать любое название, в дальнейшем оно не будет играть никакой роли. В разделе JRE выбираем последнюю исполняемую среду, она называется JavaSE-*. Во время разработки программы была доступна версия 1.8, поэтому в этом случае была выбрана именно она. В project layout желательно отметить “Use project folder as root for sources and class files” для удобства расположения файлов. Затем нажимаем Finish.

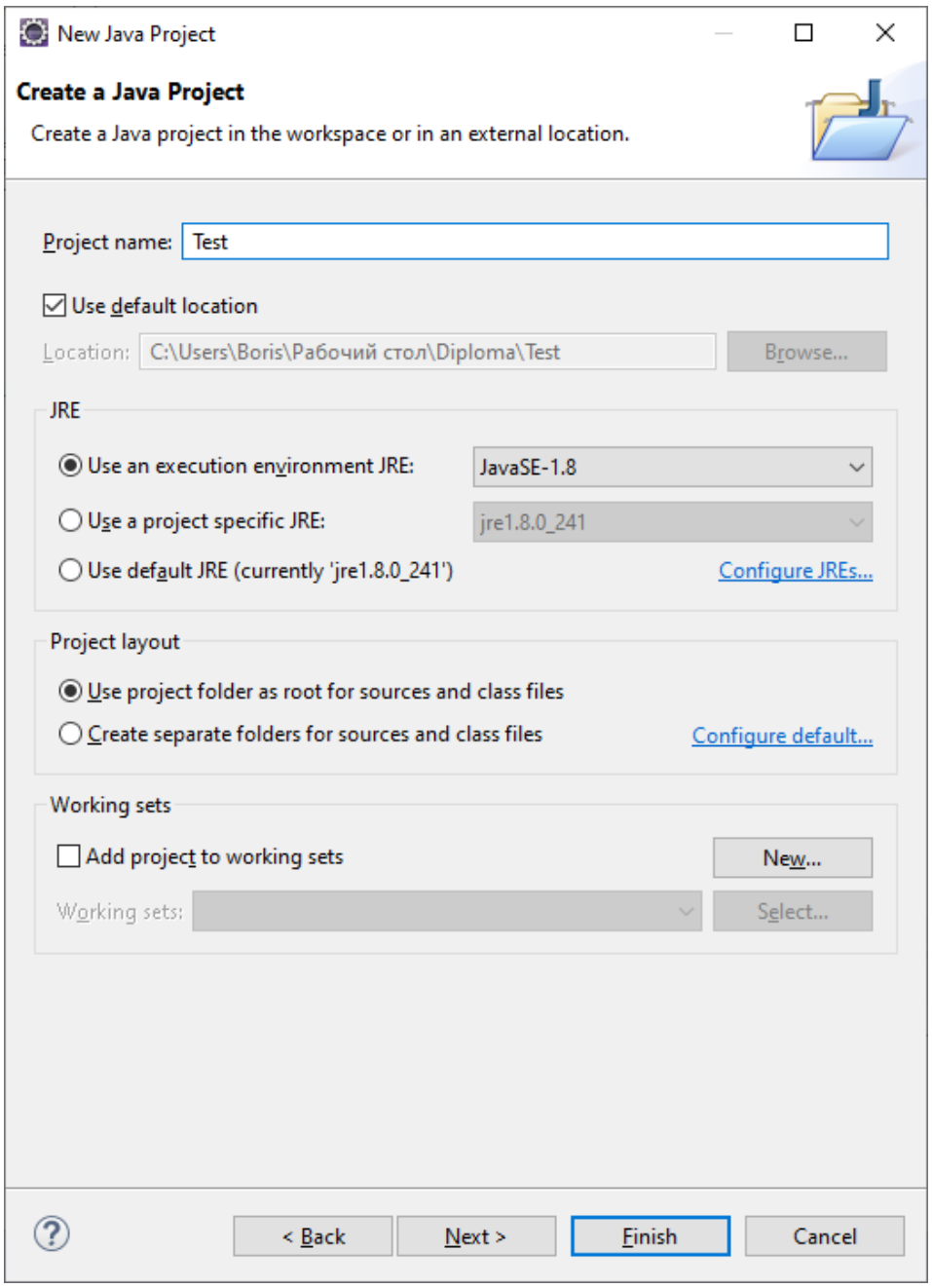

Рис. 16. Создание проекта

Теперь создался проект. Далее нажимаем по нему правой кнопкой мыши и выбираем new, затем class. В нем в строке Name задаем имя класса. В рамках данной работы, он был назван “Main”. Также рекомендуется поставить галочку напротив “public static void main(String[] args)”. Это позволит автоматически создать точку входа и еще не придется писать вручную. Нажимаем finish.

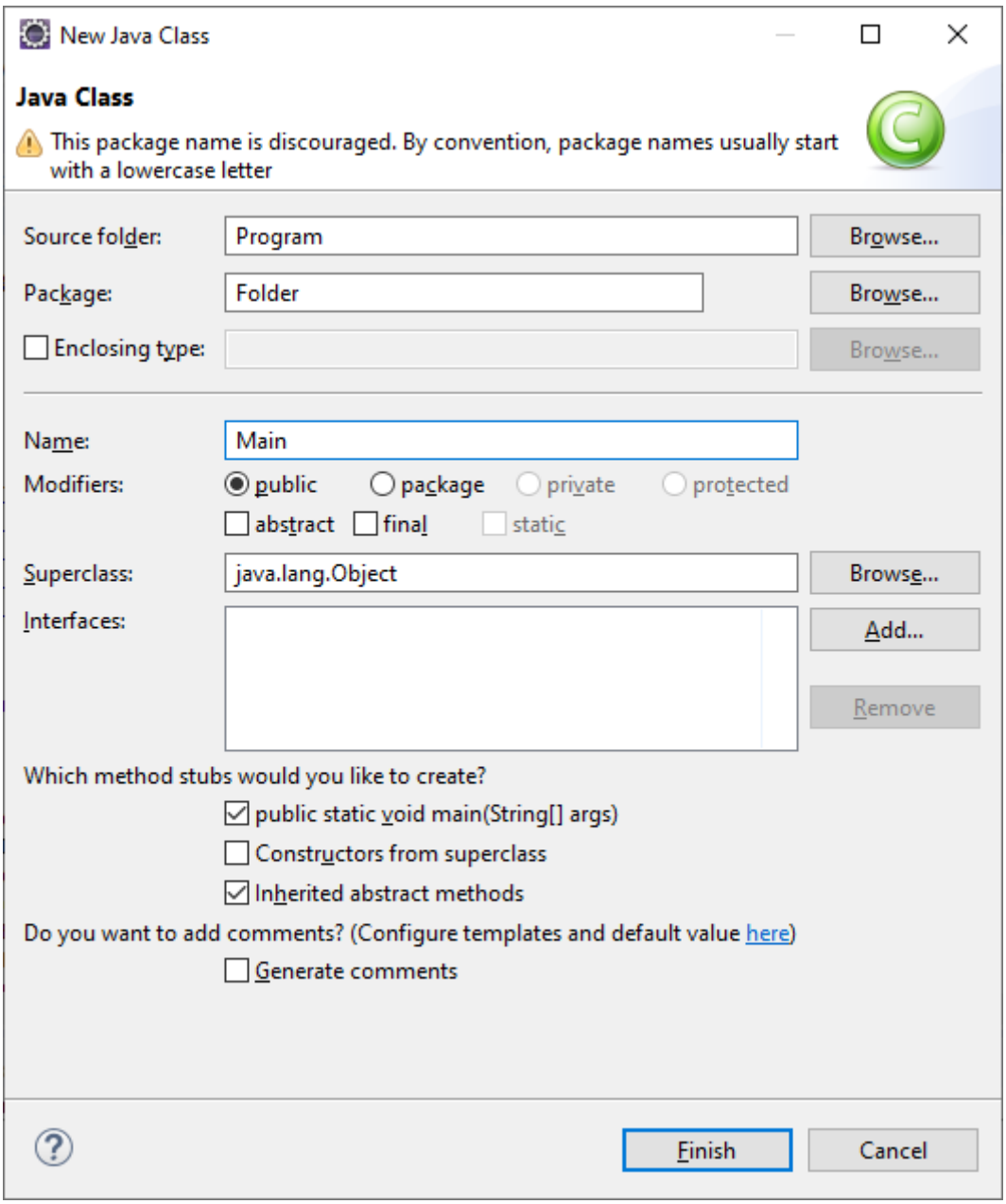

Рис. 17. Создание класса

Для удобства чтения нашей программы стоит разбить ее на 3 файла форматом *.java. Первый, который только создали “Main”, второй будет “Calculation” и, наконец, третий “Draw”. В “Calculation” будут вестись расчеты нашей функции, которая в последствие будет отрисовываться классом “Draw”.

Переходим в созданный класс Main. Для начала лучше все импортировать все необходимые библиотеки для работы нашей программы.

```
package Folder;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

Листинг 1. Библиотеки Main класса

В первой строке указан “package” – это пакет или же папка, где у нас будут храниться все классы. Далее идет импортирование библиотек awt и swing. Следующим шагом будет создание полей в классе Main.

```
public class Main extends
JFrame {
	double x1;
	double x2;
	double y1;
	double y2;
	String s = "";
	TextField TextField;
	TextField TextFx1;
	TextField TextFx2;
	TextField TextFy1;
	TextField TextFy2;
	JFrame Frame;
	JPanel Panel;
	JButton Button;
	JLabel Label;
```

Листинг 2. Поля класса Main

Где double – число с плавающей точкой, String – строка, TextField – поле текста, JFrame – окно, JPanel – панель, JButton – кнопка и JLabel – метка.

Далее создаем конструктор класса Main и описываем его, создавая окно, панель, кнопку и т.д.

```
Main() {
		Frame = new JFrame();

	Frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Panel = new JPanel();
		Frame.add(Panel);
		Panel.setBackground(Color.white);
		Panel.setLayout(null);
		Button = new JButton("Выполнить");
		Frame.setSize(260, 260);
		Frame.setTitle("Функция");
		TextField = new TextField();
		TextFx1 = new TextField("", 40);
		TextFx2 = new TextField("", 40);
		TextFy1 = new TextField("", 40);
		TextFy2 = new TextField("", 40);
```

Листинг 3. Конструктор Main

Для переменных TextFx1, TextFx2, TextFy1 и TextFy2 задаем размер колонок на значение 40, чтобы можно было вводить длинные числа.

```
TextField.setLocation(50, 60);
TextField.setSize(200, 20);
TextFx1.setLocation(60, 100);
TextFx2.setLocation(180, 100);
TextFx1.setSize(50, 20);
TextFx2.setSize(50, 20);
```

Листинг 4. Размещение полей

Для этих же переменных также следует установить положение и их размер.

```
Panel.add(TextFx1);
Panel.add(TextFx2);
TextFy1.setLocation(60, 145);
TextFy2.setLocation(180, 145);
TextFy1.setSize(50, 20);
TextFy2.setSize(50, 20);
Panel.add(TextFy1);
Panel.add(TextFy2);
Panel.add(TextField);
```

Листинг 5. Добавление на панель полей

Добавляем эти поля на панель, таким образом, они будут отображаться на нем.

```
JLabel LabelFunc = new JLabel("Введите функцию:");
JLabel LabelFx = new JLabel("F(x) =");
JLabel Labelx1 = new JLabel("X от:");
JLabel Labelx2 = new JLabel("до:");
JLabel Labely1 = new JLabel("Y от:");
JLabel Labely2 = new JLabel("до:");
```

Листинг 6. Создание надписей

Следующим шагом будет создание надписи. LabelFunc отвечает за создание надписи с названием “Введите функцию: ”. Она будет сообщать пользователю, что рядом с ней нужно ввести функцию. Для метки LabelFx с названием “F(x) =” будет означать, что следующее окно будет полем для ввода функции. Остальные метки будут предполагать, что пользователь

заметит, где нужно будет вводить ограничения по размеру графика функций начала и конца X, Y.

```
LabelFunc.setLocation(20, 20);
LabelFunc.setSize(140, 20);
Panel.add(LabelFunc);
LabelFx.setLocation(10, 60);
LabelFx.setSize(40, 20);
Panel.add(LabelFx);
Labelx1.setLocation(14, 100);
Labelx1.setSize(40, 20);
Panel.add(Labelx1);
Labelx2.setLocation(150, 100);
Labelx2.setSize(40, 20);
Panel.add(Labelx2);
Labely1.setLocation(14, 145);
Labely1.setSize(40, 20);
Panel.add(Labely1);
Labely2.setLocation(150, 145);
Labely2.setSize(40, 20);
```

Листинг 7. Добавление меток на панель

Далее видно, что следует задать размер и размещение для наших созданных меток, а также стоит сразу же добавить их на панель. Для удобства можно для каждой метки сначала задавать местоположение метки, затем задать ей размер, а после этого добавить ее на панель.

```
Button.setLocation(145, 180);
Button.setSize(100, 45);
Panel.add(Button);
Label = new JLabel("");
Label.setLocation(10, 210);
Label.setSize(170, 20);
Panel.add(Label);
```

Листинг 8.Установка кнопки

В следующем действии задаем точно также расположение для кнопки и ее размер, затем добавляем ее на панель. Далее нам понадобиться метка, ее таким же образом добавляем на панель и устанавливаем размеры и местоположение.

```
Frame.setVisible(true);

      Frame.setResizable(false);
```

Листинг 9. Отображение окна

Эти функции задают значения для первого нашего созданного окна. Они подразумевают, что наше окно можно будет увидеть, а также запретить изменять его размер.

```
Button.addActionListener(new ActionListener() {
                         public void actionPerformed(ActionEvent e) {
```

Листинг 10. Описание кнопки

Следующее действие приведет к тому, что мы будем описывать нашу кнопку. К полю Button добавляется слушатель, после чего можно выполнять операции с кнопкой.

```
s = TextField.getText();
if (!s.equals("")) {
       x1 = Double.parseDouble(TextFx1.getText());
       x2 = Double.parseDouble(TextFx2.getText());
       try {
              if (x1 < x2) {
                     y1 = Double.parseDouble(TextFy1.getText());
                     y2 = Double.parseDouble(TextFy2.getText());
                           try {
                                  if (y1 < y2) {
```

Листинг 11. Передача данных в кнопку

Передаются значения ограничений графика в кнопку, и проверяется их правильность заполнения.

```
try {

	Draw z = new Draw(x1, x2, y1, y2, s);
                                                                }
catch (Exception error1) {

	set("Ошибка!");
	System.out.println(error1);
                                                                }
	} else {

	set("Некорректный интервал Y");
                                                            }
	} catch (Exception error2) {

	set("Некорректные координаты");
	}
	} else {
	set("Некорректный интервал X");
                                                        }
} catch (Exception error3) {
	set("Некорректные координаты");
                                                }
                                        } else {
	set("Введите функцию");
                                                }
```

Листинг 12. Описание проверки ограничений графика

Здесь выполняет описание ошибок, которые мог совершить пользователь, во время заполнения полей.

```
public static void main(String args[])
{
                new Main();
                }
```

Листинг 13. Точка входа в программу

Следующая запись гласит, что в этом месте происходит вход в программу. Кроме того, здесь объявляется создание конструктора для класса Main.

```
public void set(String s) {
                Label.setText(s);
                }
```

Листинг 14. Процедура set

В конец нашего класса стоит создать нетипизированную функцию или же процедуру. Назовем ее set, и установим для метки установки текста с передаваемой переменой в процедуру.

Здесь описания класса Main завершается. Переходим к следующему.

Заходим в Calculation и импортируем необходимые библиотеки.

```
package Folder;

import java.util.*;
```

Листинг 15. Импортирование библиотеки util

```
public class Calculation {
      ArrayList<String> arraylist;

            Calculation(String string) {
```

Листинг 16. Класс Calculation

В данном классе мы объявляем глобальную переменную arraylist с типом ArrayList. ArrayList<String> означает, что объявляется строчный массив. Он позволяет изменять свой размер во время исполнения программы, и при этом необязательно указывать его размер при создании.

```
ArrayList<String> list = new ArrayList<String>();
            list.add("(");
            String[] array = new String[1];
            array = (string + " ").split(" ");
            String k = "";
            for (int i = 0; i < array.length; i++) {
                  k = k + array[i];
            }
            array = k.split("");
                  int r = 0;
```

Листинг 17. Описание переменной list

Далее определяем переменную list. Создаем переменную array с типом String[]. Разделяем ее с помощью функции split, и снова создаем переменную k с типом string. После этого заполняем строковую переменную k, заносим в переменную array разделенную строку k. Объявляем и определяем переменную r.

```
for (int i = 0; i < array.length; i++) {
if (array[i].equals("+") | array[i].equals("-") |
array[i].equals("*") | array[i].equals("/")
                                    | array[i].equals("(") |
array[i].equals(")") | array[i].equals("^")) {
String s = "";
for (int j = r; j < i; j++) {
            s = s + array[j];
                        }
list.add(s);
list.add(array[i]);
r = i + 1;
```

Листинг 18. Начало цикла

В созданном цикле идет проверка на знак +, -, *, / и так далее. Создается переменная s с типом string. И внутри первого цикла создается еще один, который заполняет переменную s. После заполнения переменной s, ее добавляют в строковый массив.

```
else {
if ((i + 2) < array.length) {
if ((array[i] + array[i + 1] + array[i + 2]).equals("sin")
                                                | (array[i] + array[i +
1] + array[i + 2]).equals("cos")
                                                | (array[i] + array[i +
1] + array[i + 2]).equals("ctg")
                                                | (array[i] + array[i +
1] + array[i + 2]).equals("sqrt")) {
list.add(array[i] + array[i + 1] + array[i + 2]);
i = i + 2;
r = i + 1;
            }
```

Листинг 19. Выполнение другого условия

```
} else {
      if ((i + 1) < array.length) {
            if ((array[i] + array[i + 1]).equals("tg") | (array[i]
+ array[i + 1]).equals("ln")
                                                      | (array[i] +
array[i + 1]).equals("lg") | (array[i] + array[i +
1]).equals("Pi")) {
list.add(array[i] + array[i + 1] + array[i + 2]);
i = i + 1;
r = i + 1;
                  }
                  }
            }
      }
}
```

Листинг 20. Выполнение последнего условия, в случае неудачи первых двух

```
String s = "";
      for (int j = r; j < array.length; j++) {
                  s = s + array[j];
            }

            list.add(s);

                  int q = 0;
```

Листинг 21. Заполнение переменной s

```
while (q == 0) {
                  q = 1;
                  for (int i = 0; i < list.size(); i++) {
                        if (list.get(i).equals("")) {
                              list.remove(i);
                         q = 0;
                        }
                  }
                        }
                  q = 0;
```

Листинг 22. Проверка переменной list

```
while (q == 0) {
                  q = 1;
      for (int i = 0; i < list.size(); i++) {
                        if (list.get(i).equals("-"))
{
      if (list.get(i - 1).equals("(")) {
      list.remove(i);
      list.add(i, "-1");
      list.add(i + 1, "*");
                                          q = 0;
                                    }
                              }
                  }
                  }
```

Листинг 23. Проверка list на знак “-“

```
arraylist = new ArrayList<String>();
            ArrayList<String> stack = new
ArrayList<String>();
            stack.add("");
            list.add(")");
            list.add("?");
                  int i = 0;
```

Листинг 24. Создание нового списочного массива

```
while (!list.get(i).equals("?")) {
	if (list.get(i).equals("(")) {
	stack.add("(");
		i++;
			} else if (list.get(i).equals("*") |
list.get(i).equals("/")) {
				if (stack.get(stack.size() -
1).equals("*") | stack.get(stack.size() - 1).equals("/")) {

	arraylist.add(stack.get(stack.size() - 1));
					stack.remove(stack.size() - 1);
				} else {
					stack.add(list.get(i));
					i++;
				}
			} else if (list.get(i).equals("+") |
list.get(i).equals("-")) {
				if (stack.get(stack.size() -
1).equals("(")) {
	stack.add(list.get(i));
		i++;
	} else {

	arraylist.add(stack.get(stack.size() - 1));
					stack.remove(stack.size() - 1);
					}
```

Листинг 25. Проверка строкового массива на знак "?"

```
} else if (list.get(i).equals(")")) {
				while (!stack.get(stack.size() -
1).equals("(")) {

	 arraylist.add(stack.get(stack.size() - 1));
			stack.remove(stack.size() - 1);
				}
				i++;
				stack.remove(stack.size() - 1);
			 } else if (list.get(i).equals("^") |
list.get(i).equals("sin") | list.get(i).equals("cos")
					| list.get(i).equals("tg") |
list.get(i).equals("ctg") | list.get(i).equals("lg")
					| list.get(i).equals("ln") |
list.get(i).equals("sqrt")) {
				stack.add(list.get(i));
				i++;
			} else {
				arraylist.add(list.get(i));
				i++;
			}
			}
```

Листинг 26. Продолжение цикла while

```
            q = 0;

            while (q == 0) {
                  q = 1;
                  for (i = 0; i < arraylist.size(); i++) {
                        if (arraylist.get(i).equals("")) {
                              arraylist.remove(i);
                              q = 0;
                        }
                  }
                  }
```

Листинг 27. Проверка списочного массива на пустой знак

```
public Double algorithm(double x) {
            ArrayList<Double> stack = new ArrayList<Double>();
            for (int i = 0; i < arraylist.size(); i++) {
                  if (arraylist.get(i).equals("+")) {
                        double t = stack.get(stack.size() - 1) +
stack.get(stack.size() - 2);
                        stack.remove(stack.size() - 1);
                        stack.remove(stack.size() - 1);
                        stack.add(t);
                  } else if (arraylist.get(i).equals("-")) {
                        double t = stack.get(stack.size() - 2) -
stack.get(stack.size() - 1);
                        stack.remove(stack.size() - 1);
                        stack.remove(stack.size() - 1);
                        stack.add(t);
                  } else if (arraylist.get(i).equals("*")) {
                        double t = stack.get(stack.size() - 2) *
stack.get(stack.size() - 1);
                        stack.remove(stack.size() - 1);
                        stack.remove(stack.size() - 1);
                        stack.add(t);
                  } else if (arraylist.get(i).equals("/")) {
                        double t = stack.get(stack.size() - 2) /
stack.get(stack.size() - 1);
                        stack.remove(stack.size() - 1);
                        stack.remove(stack.size() - 1);
                        stack.add(t);
                  } else if (arraylist.get(i).equals("^")) {
                        double t = Math.pow(stack.get(stack.size() -
2), stack.get(stack.size() - 1));
                        stack.remove(stack.size() - 1);
                        stack.remove(stack.size() - 1);
                              stack.add(t);
```

Листинг 28. Создание функции типа Double

В этой функции типа Double идет вычисление функции. Так как функции практически всегда используются символы операции, а также тригонометрические функции, например, sin(x), то идет проверка на их присутствие.

```
} else if (arraylist.get(i).equals("x") |
arraylist.get(i).equals("X")) {
                        stack.add(x);
                    } else if
(arraylist.get(i).equals("sin")) {
                        double t =
Math.sin(stack.get(stack.size() - 1));
                        stack.remove(stack.size() - 1);
                        stack.add(t);
                    } else if
(arraylist.get(i).equals("cos")) {
                        double t =
Math.cos(stack.get(stack.size() - 1));
                        stack.remove(stack.size() - 1);
                        stack.add(t);
                    } else if
(arraylist.get(i).equals("tg")) {
                        double t =
Math.tan(stack.get(stack.size() - 1));
                        stack.remove(stack.size() - 1);
                        stack.add(t);
                    } else if
(arraylist.get(i).equals("sqrt")) {
                        double t =
Math.sqrt(stack.get(stack.size() - 1));
                        stack.remove(stack.size() - 1);
                              stack.add(t);
```

Листинг 29. Продолжение проверки функции Algorithm

Здесь проверяется, что задействован ли знак x в нижнем или верхнем регистре, проверяется на наличие тригонометрических функций, таких как синус, косинус, тангенс, а также квадратный корень (обозначается в программе как запись sqrt).

```
} else if (arraylist.get(i).equals("ctg")) {
                        double t =
Math.cos(stack.get(stack.size() - 1)) /
Math.sin(stack.get(stack.size() - 1));
                        stack.remove(stack.size() - 1);
                        stack.add(t);
                    } else if (arraylist.get(i).equals("ln")) {
                        double t =
Math.log(stack.get(stack.size() - 1));
                        stack.remove(stack.size() - 1);
                        stack.add(t);
                    } else if (arraylist.get(i).equals("lg")) {
                        double t =
Math.log10(stack.get(stack.size() - 1));
                        stack.remove(stack.size() - 1);
                        stack.add(t);
                    } else if (arraylist.get(i).equals("Pi") |
arraylist.get(i).equals("pi")) {
                        stack.add(Math.PI);
                    } else if (arraylist.get(i).equals("e") |
arraylist.get(i).equals("E")) {
                        stack.add(Math.E);
                    } else {

      stack.add(Double.parseDouble(arraylist.get(i)));
                        }
```

Листинг 30. Остальная проверка на наличие тригонометрических функций

В этой части кода идет проверка на наличие таких функций как котангенса, натурального логарифма, десятичного логарифма, числа Пи и экспоненты.

```
}
            double y = stack.get(0);
            return y;
      }
      }
```

Листинг 31. Конец функции Algorithm

В конце функции возвращается значение y типа double с плавающей точкой, которому присвоено значение stack.get(0), возвращающий первый элемент списка stack.

Переходим к последнему описанию класса. Заходим в раннее созданный Draw.java.

```
package Folder;

import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;
import javax.swing.JFrame;
import javax.swing.JPanel;
```

Листинг 32. Импортирование библиотека для Draw.java

Для начала, как и других *.java файлов, требуется импортировать библиотеки. Нам понадобятся awt и swing.

```
public class Draw extends JPanel {
	double[] X;
	double x1, x2, y1, y2,
step_x, step_y;
	int width;
	int height;
	int last_X;
	int last_Y;
	Calculation a;
```

Листинг 33. Начало описания класса Draw

В этом классе понадобятся несколько переменных типов double, double[], int и Calculation.

```
Draw(double x1, double x2, double y1, double y2, String s)
{
		this.x1 = x1;
		this.x2 = x2;
		this.y1 = y1;
		this.y2 = y2;
		height = 480;
		width = 640;
		step_x = Math.PI;
		step_y = 1;
		last_X = 0;
		last_Y = 0;
		a = new Calculation(s);
		openFrame();
		}
```

Листинг 34. Конструктор класс Draw

Далее объявляется и описывается конструктор класса Draw с параметрами double x1, double x2, double y1, double y2 и String s. В нем глобальные переменные присваиваются параметрам конструктора с именем

x1, x2, y1, y2. Задается значения для переменных width и height (в дальнейшем они послужат для размера окна). Кроме того, создается еще несколько переменных и вызывается функция openFrame().

```
public void openFrame() {
		JFrame Frame = new JFrame("График функции");
		Frame.setBounds(100, 100, width + 6, height +
28);
		Frame.setLayout(null);

	Frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Frame.setVisible(true);
		Frame.setResizable(false);
		this.setSize(width, height);
		Frame.add(this);
			this.setBackground(Color.WHITE);
```

Листинг 35. Описание процедуры openFrame

Следующим шагом будет описание процедуры openFrame. В ней создается окно с названием Frame, задаются размеры функцией setbounds, устанавливается функция для правильного выхода из программы, устанавливается видимость окна, запрещается его изменение, и цвет фона становится белым.

```
MouseAdapter MouseAdapter = new MouseAdapter() {
		@Override
		public void mouseMoved(MouseEvent e) {
			last_X = e.getX();
			last_Y = e.getY();
		}

		@Override
		public void mousePressed(MouseEvent e) {
			last_X = e.getX();
			last_Y = e.getY();
			}
```

Листинг 36. Обработка мыши

Следующий код отображает обработку мыши, в момент ее перемещения и нажатия, в этом случае запоминаются последние координаты.

```
@Override
        public void mouseDragged(MouseEvent e) {
                if (e.getModifiers() == e.BUTTON1_MASK) {
                                        int new_X = e.getX();
                                        int new_Y = e.getY();
double dx = (x2 - x1) / width * (last_X - new_X);
double dy = -(y2 - y1) / height * (last_Y - new_Y);
                                        x1 += dx;
                                        x2 += dx;
                                        y1 += dy;
                                        y2 += dy;
                                        last_X = new_X;
                                        last_Y = new_Y;
                                        repaint();
        } else if (e.getModifiers() == e.BUTTON3_MASK) {
                                        int new_X = e.getX();
                                        int new_Y = e.getY();
double dx = (x2 - x1) / width * (last_X - new_X);
double dy = -(y2 - y1) / height * (last_Y - new_Y);
                                        x2 += dx;
                                        y1 += dy;
                                        last_X = new_X;
                                        last_Y = new_Y;
                                        repaint();
                                }
                                }
```

Листинг 37. Обработка нажатий мыши

Далее идет обработка нажатий мыши. По нажатию левой кнопки мыши будет осуществляться перемещение камеры в окне с графиком функции, а по нажатию правой кнопки мыши – увеличение или уменьшение сетки с графиком. Метод repaint() отвечает за перерисовку графика при перемещении мыши.

```
addMouseListener(MouseAdapter);
                addMouseMotionListener(MouseAdapter);
                this.addMouseWheelListener(new
MouseWheelListener() {
                        @Override
public void mouseWheelMoved(MouseWheelEvent e) {
        int u = e.getWheelRotation();
        double dx = (x2 - x1) / (10 + Math.abs(u + 1) / 2);
        double dy = (y2 - y1) / (10 + Math.abs(u + 1) / 2);
                x1 -= u * dx * last_X / width;
                x2 += u * dx * (width - last_X) / width;
                y1 -= u * dy * (height - last_Y) / height;
                                y2 += u * dy * last_Y / height;
                                repaint();
                        }
                });
                }
```

Листинг 38. Обработка колесика мыши

Данный код отображает описание прокрутки колеса мыши. В случае если колесико мыши крутят вперед, то график функции приближается, и наоборот, если колесико крутят назад, то график функции отдаляется.

```
@Override
	public void paintComponent(Graphics graph) {
		super.paintComponent(graph);
		paint1(graph);
		Graphics2D graph2d = (Graphics2D) graph;
		graph2d.setStroke(new BasicStroke(2));
		graph.setColor(Color.RED);
		paint2(graph);
		}
```

Листинг 39. Отрисовка функции

```
public void paint1(Graphics graph) {
		graph.setColor(Color.GRAY);
		int OX = (int) (-x1 / (x2 - x1) * width);
		int OY = (int) (height + y1 / (y2 - y1) * height);
for (int i = (int) Math.floor(x1 / step_x); i <= Math.floor(x2 / step_x);
i++) {
			int positionX = (int) (-(x1 - step_x * i) / (x2 - x1) *
width);
			graph.drawLine(positionX, 0, positionX, height);
			int positionY;
			if (OY < 12) {
				positionY = 10;
			} else if (OY > height - 2) {
				positionY = height - 2;
			} else {
				positionY = OY - 2;
			}
			int format = (int) (height / 5 / (x2 - x1) / step_x);
			if (step_x * i == (int) (step_x * i)) {
				format = 0;
			} else if (format > 10) {
				format = 10;
			}
			graph.drawString(String.format("%." +
String.valueOf(format) + "f", i * step_x), positionX + 2, positionY);
			}
```

Листинг 40. Отрисовка вспомогательных линий графика

```
for (int i = (int) Math.floor(y1 / step_y); i <= Math.floor(y2 /
step_y); i++) {
                    int positionY = (int) (height + (y1 - step_y * i)
/ (y2 - y1) * height);
                    graph.drawLine(0, positionY, width, positionY);
                    int positionX;
                    int format = (int) (height / 5 / (x2 - x1) /
step_y);
                    if (step_y * i == (int) (step_y * i)) {
                          format = 0;
                    } else if (format > 10) {
                          format = 10;
                    }

                    String formatted_value = String.format("%." +
String.valueOf(format) + "f", i * step_y);
                    int length = (int) new
TextLayout(formatted_value, graph.getFont(), new
FontRenderContext(null, true, true))
                                .getBounds().getWidth();

                    if (OX < 1) {
                          positionX = 2;
                    } else if (OX > width - 7 - length) {
                          positionX = width - 5 - length;
                    } else {
                          positionX = OX + 2;
                    }
                    graph.drawString(formatted_value, positionX,
positionY - 1);
              }
              graph.setColor(Color.BLACK);
              graph.fillRect(OX - 1, 0, 3, height);
              graph.fillRect(0, OY - 1, width, 3);
              }
```

Листинг 41. Отрисовка оси абсцисс и ординат

```
public void paint2(Graphics graph) {
              int h1 = height - (int) Math.floor((height /
(Math.abs(y2 - y1))) * (a.algorithm(x1) - y1));
              for (int i = 1; i < width; i++) {
                    double i2 = a.algorithm(x1 + ((Math.abs(x2 -
x1)) / width) * i);
                    int h2 = height - (int) Math.floor((height /
(Math.abs(y2 - y1))) * (i2 - y1));
                    graph.drawLine(i - 1, h1, i, h2);
                    h1 = h2;
              }
        }
        }
```

Листинг 42. Процедура paint2

Последняя процедура, описывающая отображение функции на графике.

ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы было разработано приложение, отображающие действия математических алгоритмов в компьютерной графике.

Был проведен сравнительный анализ языков программирования. Им стал Java, так как позволяет запустить приложения на различных операционных системах.

Приложение разработано в соответствии с техническим заданием.

ЛИТЕРАТУРА

1. Алгоритм // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Алгоритм (дата обращения: 01.06.2020).
2. ГОСТ 19.502-78. Описание применения. Требования к содержанию и оформлению; Введ. 1980-01-01. М.: Стандартинформ. 2 с. (Единая система программной документации).
3. ГОСТ 19.503-79. Руководство системного программиста. Требования к содержанию и оформлению; Введ. 1980-01-01. М.: Стандартинформ. 3 с. (Единая система программной документации).
4. ГОСТ 19.504-79. Руководство Программиста. Требования к содержанию и оформлению; Введ. 1980-01-01. М.: Стандартинформ. 3 с. (Единая система программной документации).
5. ГОСТ 19.505-79. Руководство Оператора. Требования к содержанию и оформлению; Введ. 1980-01-01. М.: Стандартинформ. 3 с. (Единая система программной документации).
6. ГОСТ 2.105-95. Общие требования к текстовым документам. Взамен ГОСТ 2.105-79, ГОСТ 2.906-71; Введ. 1996-07-01. М.: Всероссийский научно-исследовательский институт стандартизации и сертификации в машиностроении (ВНИИНМАШ). 27 с. (Межгос. стандарт. Единая система конструкторской документации).
7. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Взамен ГОСТ 7.0-84; Введ. 2004-07-01. Межгосударственный совет по стандартизации, метрологии и сертификации. М. : Изд-во стандартов. 111 с. (Межгос. стандарт. Система стандартов по информации, библиотечному и издательскому делу. Общие требования и правила составления).
8. ГОСТ Р 52653-2006. Информационно-коммуникационные технологии в образовании. Термины и определения; Введ. 2008-07-01. М.: Стандартинформ. 18 с. (Национальный Стандарт Российской Федерации).
9. ГОСТ Р 52657-2006. Образовательные интернет-порталы федерального уровня. Рубрикация информационных ресурсов; Введ. 2008-07-01. М.: Стандартинформ. 10 с. (Информационно-коммуникационные технологии в образовании).
10. ГОСТ Р 53620-2009. Электронные образовательные ресурсы. Информационно-коммуникационные технологии в образовании; Введ.

2011-01-01. М.: Стандартинформ. 10 с. (Информационно-коммуникационные технологии в образовании).

11. ГОСТ Р 7.0.83-2013. Электронные издания. Основные виды и выходные сведения. С изм. и допол. в ред. от 12.09.2018.; Введ. 2013-10-15. М. : Стандартинформ. 16 с. (Система стандартов по информации, библиотечному и издательскому делу).
12. Джошуа Блох Java. Эффективное программирование. - М.: Лори, 2014.
13. Загрузить Java для Windows // Java.com URL: https://www.java.com/ru/download/win10.jsp (дата обращения: 15.03.2020).
14. Калькулятор (Windows) // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Калькулятор_(Windows) (дата обращения: 01.06.2020).
15. Оформитель библиографических ссылок // SNOSKA.INFO URL: http://snoskainfo.ru/ (дата обращения: 10.02.2020).
16. Половко А.М., Бутусов П.Н. Интерполяция. Методы и компьютерные технологии их реализации. BHV, 2004 . - 320 с.
17. Попробовать Adobe Photoshop бесплатно // Adobe URL: https://www.adobe.com/ru/products/photoshop/free-trial-download.html (дата обращения: 31.05.2020).
18. Шилдт, Г Java-8-Полное-руководство. - Полное руководство изд. Вильямс, 2017. - 1376 с.
19. Язык программирования Python 3 для начинающих // Python 3 для начинающих URL: https://pythonworld.ru/ (дата обращения: 14.03.2020).
20. Abstract Window Toolkit // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Abstract_Window_Toolkit (дата обращения: 01.06.2020).
21. Adobe Photoshop // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Adobe_Photoshop (дата обращения: 31.05.2020).
22. BPwin // KPMS URL: https://www.kpms.ru/Automatization/BPwin.htm (дата обращения: 31.05.2020).
23. C++ // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/C%2B%2B (дата обращения: 01.04.2020).
24. Download Eclipse Technology that is right for you // Eclipse URL: https://www.eclipse.org/downloads/ (дата обращения: 31.05.2020).

25. Eclipse (среда разработки) // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Eclipse_(среда_разработки) (дата обращения: 31.05.2020).
26. Erickson J. Algorithms. - 978-1792644832 изд. June 13, 2019. - 472 с.
27. Evans B Java: The Legend. - 9781491934661 изд. - Newton, Massachusetts, USA: O'Reilly Media, Inc., 2015. - 61 с.
28. Java // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Java (дата обращения: 15.03.2020).
29. Python // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Python (дата обращения: 14.03.2020).
30. Swing // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Swing (дата обращения: 01.06.2020).