# Содержание

[Аннотация 2](#_Toc154451829)

[Введение 3](#_Toc154451830)

[1 Анализ предметной области 4](#_Toc154451831)

[2 Проектирование приложения 5](#_Toc154451832)

[3 Разработка программного обеспечения 7](#_Toc154451833)

[3.1 Описание технологического стека разработки 7](#_Toc154451834)

[3.2 описание алгоритма работы 7](#_Toc154451835)

[3.3 Описание интерфейса пользователя 11](#_Toc154451836)

[4 Тестирование 14](#_Toc154451837)

[4.1 План тестирования 14](#_Toc154451838)

[4.2 Оценка результатов проведения тестирования 15](#_Toc154451839)

[Список использованных источников 17](#_Toc154451840)

[Приложение А 18](#_Toc154451841)

[Приложение Б 19](#_Toc154451842)

# Аннотация

Ключевые слова:

* график;
* функция;
* метод.

Содержание работы представляет собой описание создания и тестирования приложения. Все это состоит из 20 страниц, которые включают в себя не более 10 рисунков, ровно 1 таблица а так же приложение А и приложение Б.

# Введение

В современном мире математика играет все более важную роль в повседневной жизни. Она пронизывает все сферы нашего существования, от финансовых расчетов до научных исследований. Решая сложные задачи и моделируя различные явления, математические функции становятся ключевым инструментом анализа окружающего мира. Однако, для большинства людей математика остается непреодолимым барьером из-за ее абстрактности и сложности.  
 В этом контексте разработка приложения для построения графиков математических функций представляет собой важную задачу, направленную на упрощение доступа к инструментам математического анализа. С помощью подобного приложения пользователи смогут быстро и эффективно визуализировать различные функции, изучать их свойства и применять полученные знания в решении конкретных задач.  
 Цель данного исследования заключается в разработке приложения для построения графиков математических функций на основе .NET WPF и языка C#. Используя эти мощные технологии, а также интегрируя сторонние фреймворки, стоит попытаться реализовать приложение, которое сочетает в себе простоту использования, эстетический дизайн и высокую производительность.  
 При разработке конкретных продуктов, от платформы .NET WPF, следует брать эффективное использование функционала этой платформы для создания гибкого и отзывчивого пользовательского интерфейса. Кроме того, приложение будет разрабатываться на языке C#, что позволит использовать его мощные инструменты для создания надежного и высокопроизводительного программного обеспечения.  
 Интеграция сторонних фреймворков также сыграет ключевую роль в обеспечении разнообразных возможностей для пользователей. При разработке следует учесть момент с использованием современных фреймворков для визуализации данных, улучшения интерактивности и оптимизации процесса разработки.

Изучение математических функций через графики является эффективным способом увидеть их поведение и взаимосвязь между переменными. Приложение, разрабатываемое на основе .NET WPF и языка C#, преследует цель предоставить пользователям инструмент для удобного и интуитивно понятного анализа функций. Визуализация математических функций в виде графиков позволит пользователям быстро увидеть их особенности, такие как экстремумы, перегибы, асимптоты, и использовать эту информацию для более глубокого понимания математических концепций.  
 Это приложение также может быть полезным для студентов и учителей, позволяя им исследовать различные функции и демонстрировать их особенности в учебных целях. Благодаря возможности масштабирования, перемещения и изменения параметров графиков, пользователи смогут в интерактивном режиме изучать различные функции и их взаимодействие.  
 Разработка данного приложения требует не только понимания математических концепций, но и умения эффективно использовать инструменты разработки, такие как .NET WPF и язык C#. Кроме того, необходимо уделить особое внимание дизайну пользовательского интерфейса, чтобы обеспечить удобство использования приложения для широкого круга пользователей.

# 1 Анализ предметной области

Предметная область — множество всех предметов, свойства которых и отношения между которыми рассматриваются в научной теории. В логике — подразумеваемая область возможных значений предметных переменных логического языка. Предметная область — часть реального мира, рассматриваемая в пределах данного контекста. Под контекстом здесь может пониматься, например, область исследования или область, которая является объектом некоторой деятельности.

Предметная область не только определяет множество объектов, рассматриваемых в пределах одного рассуждения или в научной теории, но также их свойства, отношения, функциональные связи. При этом сама по себе предметная область является абстрактным понятием, это некоторый способ «объединения предметов в мысли».

Понятие предметной области важно для гносеологии. Само понятие «знание» иногда определяют как «закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области».

Проблемная сфера человеческой деятельности, нуждающаяся в автоматизации представляет из себя математическое моделирование и анализ функций, которые играют важную роль в различных областях, включая науку, инженерное дело, финансы, исследования и образование. Одной из проблемных сфер в этой области является удобство и доступность инструментов для визуализации и анализа математических функций. Традиционные методы ручного построения графиков и анализа функций требуют значительных временных затрат и способны стать преградой для пользователей, которые не обладают специализированными математическими знаниями.

Автоматизированные инструменты для анализа и визуализации математических функций в современных компьютерных системах для анализа и визуализации математических функций используется широкий спектр автоматизированных инструментов. Эти инструменты позволяют пользователям быстро и легко создавать графики функций, находить их экстремумы, строить асимптоты и выполнять другие операции, необходимые для анализа поведения функций. Одним из наиболее распространенных инструментов для визуализации функций является графический калькулятор. Графические калькуляторы позволяют пользователям быстро и легко строить графики функций, а также находить их экстремумы и нули. Графические калькуляторы могут также использоваться для построения асимптот и для выполнения других операций, необходимых для анализа поведения функций.

Другим распространенным инструментом для анализа и визуализации функций является программа математического моделирования. Программы математического моделирования позволяют пользователям создавать модели математических функций, а затем исследовать поведение этих моделей путем построения графиков, нахождения экстремумов, построения асимптот и выполнения других операций. Программы математического моделирования также могут использоваться для решения различных математических задач, таких как нахождение корней уравнений и решение систем уравнений.

В последние годы также появились новые инструменты для анализа и визуализации функций, основанные на машинном обучении и искусственном интеллекте. Эти инструменты позволяют пользователям создавать модели математических функций, а затем использовать эти модели для прогнозирования поведения функций в будущем.

Сущности, явления, отношения и процессы:  
 1. Математические функции - различные типы функций, их свойства, графики и визуальное представление;  
 2. Визуализация данных - методы отображения математических функций, включая различные графические подходы и инструменты для интерактивной работы с графиками;  
 3. Алгоритмы анализа - автоматизированные методы для определения особых точек, нахождения экстремумов, построения асимптот и других характеристик функций;  
 4. Интерактивность - возможность взаимодействия с графиками, изменение параметров функций, подписей, масштабирования.

Различные аспекты этой деятельности требуют автоматизации, чтобы упростить и ускорить процесс анализа и визуализации математических функций, делая его более доступным для широкого круга пользователей, как на рисунке 1.

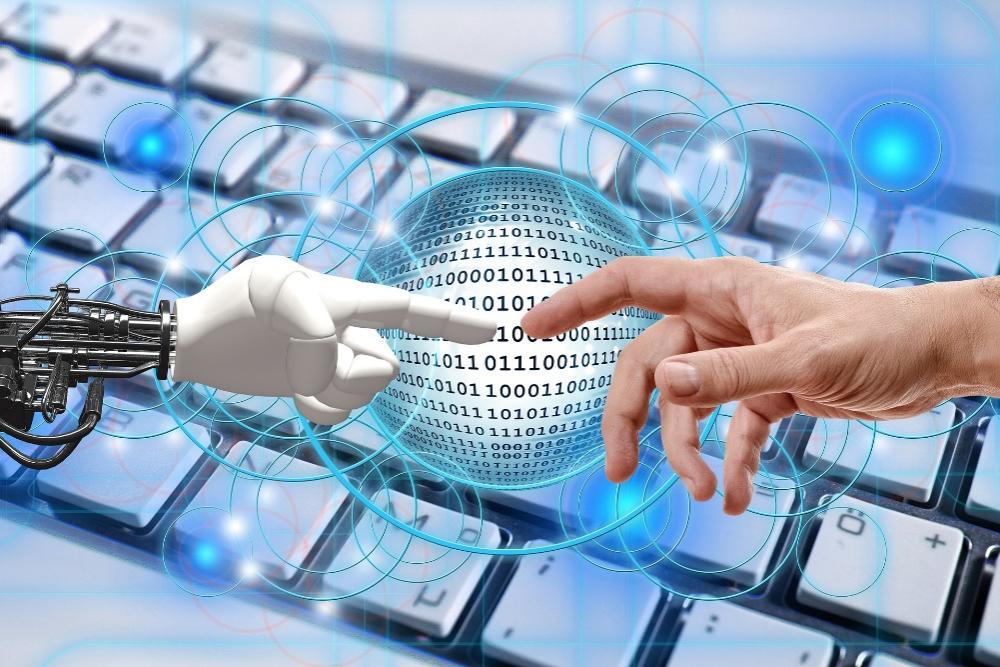


Рисунок 1 - автоматизация

# 2 Проектирование приложения

Проектирование приложения является критически важным этапом в создании программного продукта. От корректного определения требований до архитектурного проектирования, каждый шаг имеет важное значение. Недостаточное внимание к проектированию может привести к серьезным проблемам в будущем, таким как нестабильность приложения, неудовлетворение пользовательских потребностей, и сложности в дальнейшем развитии, например, недостаточное внимание к проектированию привело к печальным последствиям в ситуации, когда в 1999 году компания NASA потеряла марсианский лэндер, изображенный на рисунке 2 и носящий имя "Mars Climate Orbiter" - причиной стала ошибочная интерпретация данных из-за использования различных систем единиц в разработанных различными группами специалистов компонентах программного обеспечения [1]. Этот случай проиллюстрировал важность тщательного проектирования и обеспечения согласованности между компонентами программного продукта.

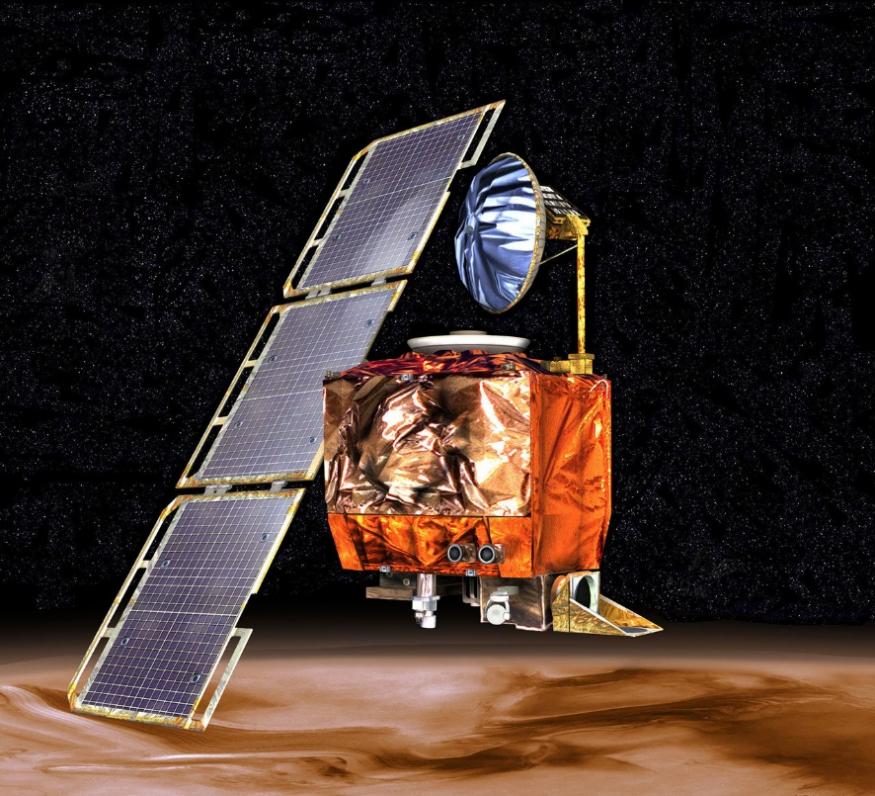


Рисунок 2 – лендер

Функциональные требования:  
 1. построение графиков - приложение должно обеспечивать возможность построения графиков для широкого спектра математических функций, включая линейные, квадратичные, тригонометрические, и другие функции;  
 2. интерактивный интерфейс - пользователи должны иметь возможность взаимодействовать с графиками, масштабировать, перемещать, и менять параметры функций для динамического анализа;  
 3. анализ функций - предоставление инструментов для определения особых точек, нахождения экстремумов, асимптот, и других характеристик функций.

Нефункциональные требования:  
 1. Производительность: Приложение должно демонстрировать отзывчивость при работе с большими объемами данных и сложными функциями;  
 2. Интуитивный интерфейс: Дружелюбный и интуитивно понятный пользовательский интерфейс, предоставляющий возможность эффективного использования инструментов приложения;  
 3. Масштабируемость: Способность обрабатывать разнообразные функциональные и нефункциональные требования при работе с различными моделями и объемами данных.

# 3 Разработка программного обеспечения

При разработке программного обеспечения для описываемого приложения, важно определить технологический стек, который будет использоваться. Для этого необходимо рассмотреть различные инструменты и технологии, позволяющие эффективно реализовать поставленную функциональность. Одним из основных инструментов, используемых в разработке программного обеспечения, является Visual Studio. Определим, что представляет собой этот инструмент.

Visual Studio - это интегрированная среда разработки (IDE) от компании Microsoft, предназначенная для создания различных типов приложений: от веб-сайтов и мобильных приложений до настольных программ. Она включает в себя широкий спектр инструментов и ресурсов для упрощения процесса разработки, включая редактор кода, отладчик, дизайнер форм, инструменты для тестирования и многие другие.

Технологический стек разработки также будет включать в себя другие инструменты, такие как языки программирования, фреймворки для визуализации данных, системы управления версиями и другие компоненты, обеспечивающие эффективное и безопасное создание программного продукта.

## 3.1 Описание технологического стека разработки

Visual Studio поддерживает различные языки программирования, включая C#, C++, JavaScript, Python, и многие другие, что делает его универсальным инструментом для широкого круга разработчиков. Благодаря своей мощной функциональности, интеграции с различными сервисами и постоянным обновлениям, Visual Studio является популярным выбором для профессиональной разработки программного обеспечения.

## 3.2 описание алгоритма работы

При разработке приложения для анализа математических функций, важно учесть не только технологическую часть, но и алгоритм функционирования, который обеспечит корректное взаимодействие с пользователем и обработку данных. Основные моменты функционирования приложения, включая хранение данных, взаимодействие с пользователем и обработку событий:

1. хранение данных – оперативная память устройства, на котором выполнен запуск программного средства;
2. взаимодействие с пользователем – интерфейс приложения будет организован таким образом, чтобы пользователь мог легко взаимодействовать с функционалом. Это включает в себя возможность ввода математических функций, настройку параметров анализа и визуализацию результатов на графиках. Для этого будут использоваться различные элементы управления, такие как текстовые поля, кнопки и интерактивные графики;
3. обработка событий – приложение будет отслеживать события, такие как ввод новой функции, изменение параметров анализа или взаимодействие с элементами интерфейса. Обработка событий позволит приложению реагировать на действия пользователя, обновлять данные и предоставлять актуальную визуализацию анализа математических функций;
4. обновление интерфейс – после обработки событий, приложение будет обновлять интерфейс, отображая изменения на графиках, обновляя текстовую информацию и предоставляя пользователю возможность взаимодействия с полученными результатами. это обеспечит понятное и интуитивно понятное взаимодействие пользователя с приложением.

Таким образом, алгоритм работы приложения для анализа математических функций включает в себя оптимальное хранение данных, удобное взаимодействие с пользователем и обработку событий для обеспечения плавного и информативного функционирования приложения и за все это отвечает ряд функций, кода, которые можно наблюдать на рисунке 3, рисунке 4, рисунке 5 и рисунке 6 .

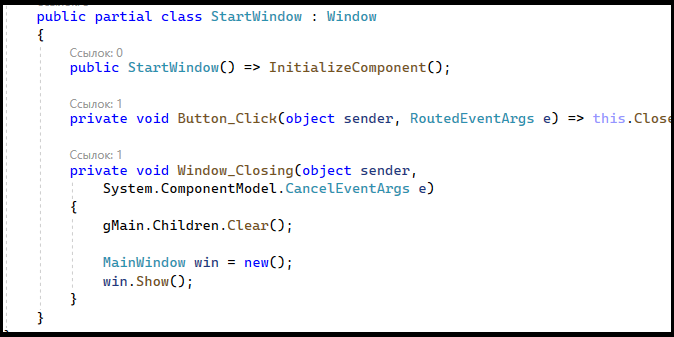


Рисунок 3 – приветствие

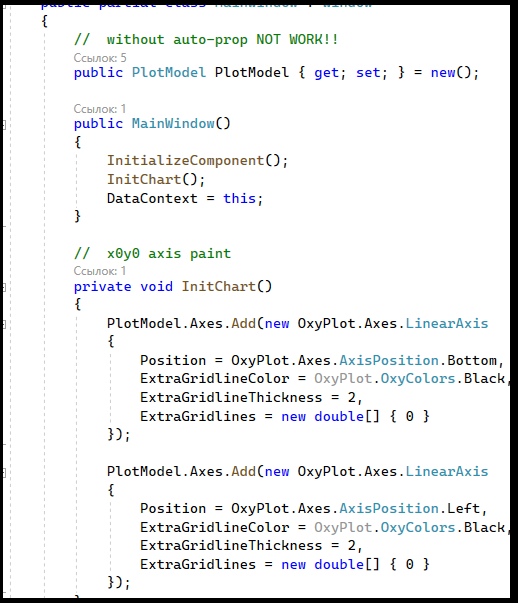


Рисунок 4 – прощание



Рисунок 5 – главный экран

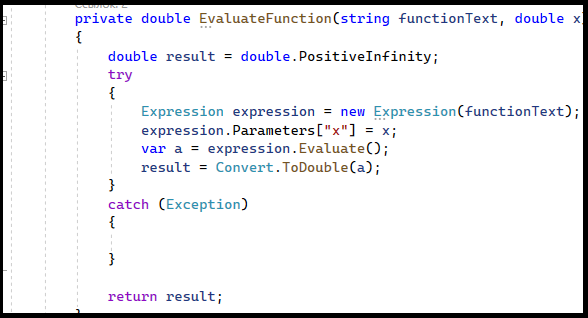


Рисунок 6 - помилование

На всех вышеизложенных иллюстрациях можно наблюдать всю логическую составляющую программы - в будущем она обязана быть подвергнута ряду тестировальных сценариев. Говоря, непосредственно, о логике есть функции отвечающие за приветственное окно, роль которого - перенаправлять пользователя на главный экран.

Так же, в наличии есть функции, отвечающие за, так называемый, парсинг строки функции. Парсинг — автоматизированный сбор и систематизация данных. В текущем случае, нужно, грубо говоря, просто интерпретировать строку в исполняемый код.

В том числе, имеется некотрая чать кода, которая отвечает за первый запуск, а в дальнейшем поддержку вида области построения графика - необходимо всегда отображать оси нулевых координат, как на рисунке 7.

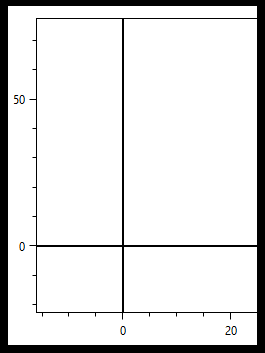


Рисунок 7 - координаты

## 3.3 Описание интерфейса пользователя

При создании интерфейса приложения для анализа математических функций, следует основываться на принципах удобства использования, эргономики и интуитивной навигации. Основная цель заключалась в создании такого интерфейса, который бы обеспечивал пользователю простое и приятное взаимодействие с функционалом, а также интуитивно понятный доступ ко всем возможностям приложения.

При разработке интерфейса следует уделить особое внимание удобству ввода математических функций, визуализации результатов и навигации по приложению. Эффективное и интуитивное взаимодействие пользователя с приложением играло ключевую роль в выборе элементов управления, их расположении и внешнем виде.

Картинки приветственного окна и основного представлены на рисунке 7, рисунке 8 и рисунке 9.

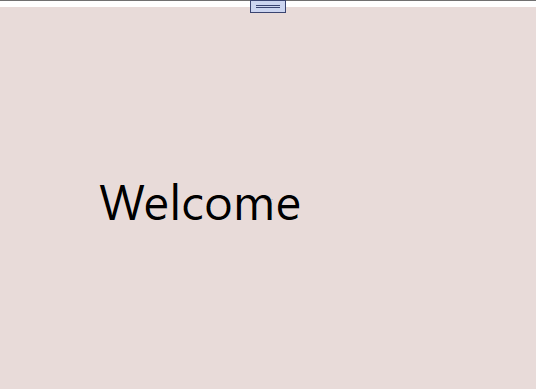


Рисунок 8 – приветственный экран

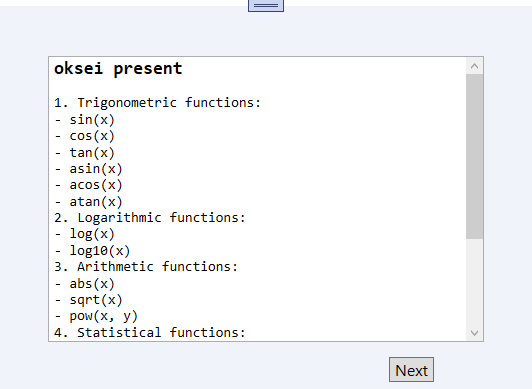


Рисунок 9 – лицензионное соглашение

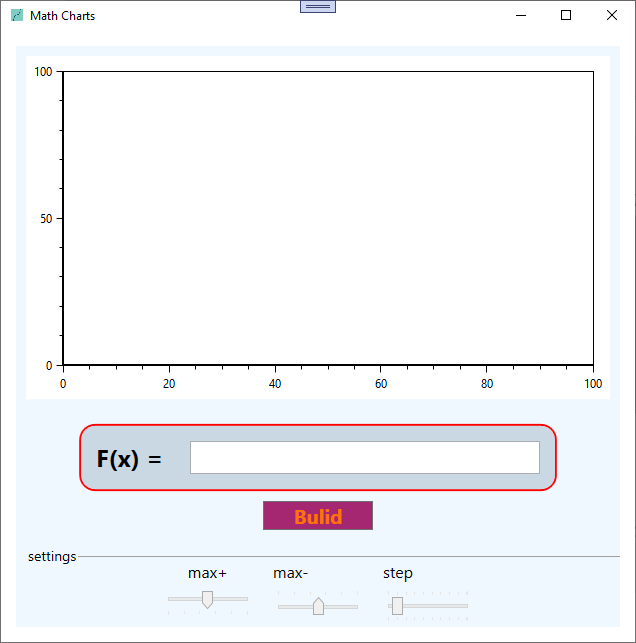


Рисунок 10 – главное окно

# 4 Тестирование

Тест-кейс представляет из себя четкое описание входных данных, условий и процедуры тестирования, ожидаемых результатов. Они определяют один сценарий — конкретную цель тестирования программного обеспечения. Целью может быть проверка ПО: соответствует ли оно требованиям.

Четко определенные тест-кейсы позволяют многократно запускать одни и те же тесты, применять для последовательно изменяющихся версий программного обеспечения. А еще отслеживать регрессивные ошибки ПО — то есть те, которые повторяются и ухудшают качество продукта.

В чек-листе перечисляют аспекты ПО, которые нужно проверить. Когда составляют тест-кейс, описывают состояние программного обеспечения и то, как его изменяют. То есть чек-листом определяют, что тестировать, а тест-кейсом — как тестировать. Чек-лист подойдет в качестве исходного документа, чтобы составить тест-кейсы.

Баг-репорт - это отчет об ошибке. Его составляют, когда находят ошибки в работе программного обеспечения. Тест-кейс же нужен, чтобы определить, есть ли ошибка. Он помогает составить качественный баг-репорт, похожий на то, что изображено на рисунке 11.

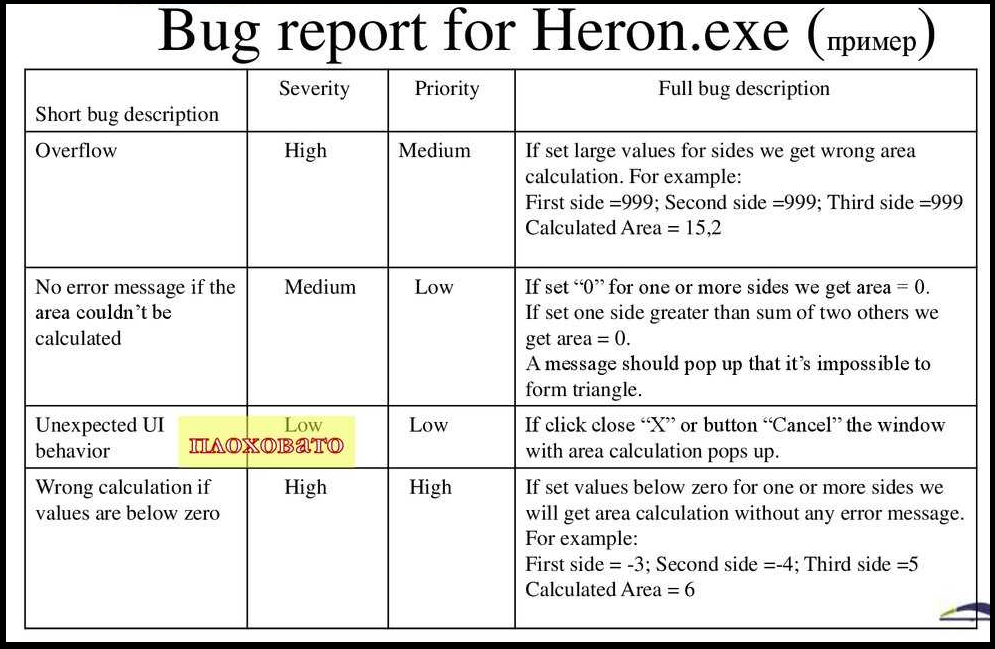


Рисунок 11 - репорт

Содержание тест-кейса, а именно его атрибуты зависят от внутренней культуры компании или возможностей систем управления тест-кейсами. И даже от типа тестируемого ПО. Обычно выделяют следующие атрибуты:

* уникальный идентификатор — некое уникальное значение. по нему на тест-кейс ссылаются из других документов или тест-кейсов. бывает буквенным, числовым, буквенно-числовым. чаще всего применяют простую сквозную нумерацию.
* краткое описание — лаконичное описание сути тест-кейса. может содержать ссылку на требование к по.
* входные данные — сведения о первоначальном состоянии системы, которое важно для тест-кейса. а еще значения для ввода или передачи по.
* шаги — полная последовательность действий. ее выполняют, чтобы провести описываемую тест-кейсом проверку.
* ожидаемый результат — описание планируемого поведения или результата по. может базироваться на требовании к программному обеспечению, общей логике работы.
* фактический результат — описание итогового поведения или результата по. если они совпадают, это указывают. когда не совпадают, подробно описывают расхождения. пометка «не совпадает», «отличается» — это грубая ошибка.
* статус — текущее состояние тест-кейса.

## 4.1 План тестирования

Исходя из вышеописанного материала, а также принимая во внимание функциональны, технические и миграционные особенности разработки, получается следующий план непосредственного составления тестирования:

* определеить баг-репортер;
* делать колонку номер1 таблицы;
* делать колонку номер2 таблицы;
* делать остальные колонки таблицы;
* составить таблицу;
* готово.

Конечно же, при тестировании приложения для построения графиков, может что-то пойти не поплану, ведь все зависит от множества факторов, например интегримрованная среда разработки или же сам язык, который используется при написании программы, в данном случае в его роли выступает обьекто-ориентированный язык программипрования c#, который разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Но в целом, тестирование приложения для построения графиков на C# является важным этапом разработки, который позволяет выявить и устранить ошибки, которые могут привести к некорректной работе программы. Проводя тщательное тестирование, можно повысить качество приложения и обеспечить его надежную работу в различных условиях.

## Оценка результатов проведения тестирования

Результаты проведенных тестовых сценариев можно наблюдать в таблице 1. Результаты тестов показывают, что основы теории поведения человека в процессе принятия решения проверены вполне убедительно. Так, во время тестирования скорость принятия решений была достаточно высокой (с учетом того, конечно, факта, насколько быстро данный вопрос может быть решен в полевых условиях).

Таблица 1 - тестирование

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Идентификатор теста | Описание | Шаги | Вх. данные | Результат ожидаемый | Результат фактический | Статус |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| gp0 | Проверка корректной отрисовки данных при минимальном диапазоне измерений | Ввод функции – выставление минимальных значений границ графика – построение графика – оценка результата | Функция: x\*x+9.2; границы: (1,-1) | Слегка заметный участок графика при приближении или же практически единственная точка при удалении (в зависимости от масштабирования плоскости) | + | ok |
| gp1 | Проверка корректной отрисовки данных при максимальном диапазоне измерений | Ввод функции – выставление максимальных значений границ графика – построение графика – оценка полученного результата | Функция: x\*x+9.2; границы: (1,-1) | Полноценный график. При приближении имеется возможность детального изучения. | - | fix |
| Продолжение таблицы 1 | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| lp0 | Проверка на некорректно веденную формулу | Ввод заведомо неверной строки в поле , где предпологалось корректная формула | Ацуаук?123 | Сообщение или отсутствие реакции | + | Ok |
| lp1 | Малые числовые значение | Подача в приложение функции, коэфицент которой достаточно велик/мал | -999999999999 | График, не подходящий под вид функции | + | ok |
| lp2 | Большие числовые значения | Подача в приложение функции, коэфицент которой достаточно велик/мал | 999999999999 | График, не подходящий под вид функции | + | ok |

# Список использованных источников

1 Mars Climate Orbiter https://mars.nasa.gov/mars-exploration/missions/mars-climate-orbiter/ (дата обращения 14.12.2023).

2 Документация по Windows Presentation Foundation [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/?view=netdesktop-8.0/

3 Руководство по WPF [Электронный ресурс] URL: https://metanit.com/sharp/wpf/ (дата обращения 15.12.2023).

4 Построение графиков на WPF форме под .NET Framework 4 [Электронный ресурс] URL: https://habr.com/ru/articles/145343/ (дата обращения 16.12.2023)

5 WPF — OxyPlot 2015.1 documentation [Электронный ресурс] URL: https://oxyplot.readthedocs.io/en/latest/getting-started/hello-wpf.html (дата обращения 17.12.2023)

6 Git - Работа с удалёнными репозиториями [Электронный ресурс] URL: https://book.git-scm.com/book/ru/v2/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-Git-%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D1%83%D0%B4%D0%B0%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%BC%D0%B8-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F%D0%BC%D/ (дата обращения19.12.2023)

# Приложение А

***(обязательное)***

**Диаграмма вариантов использования**

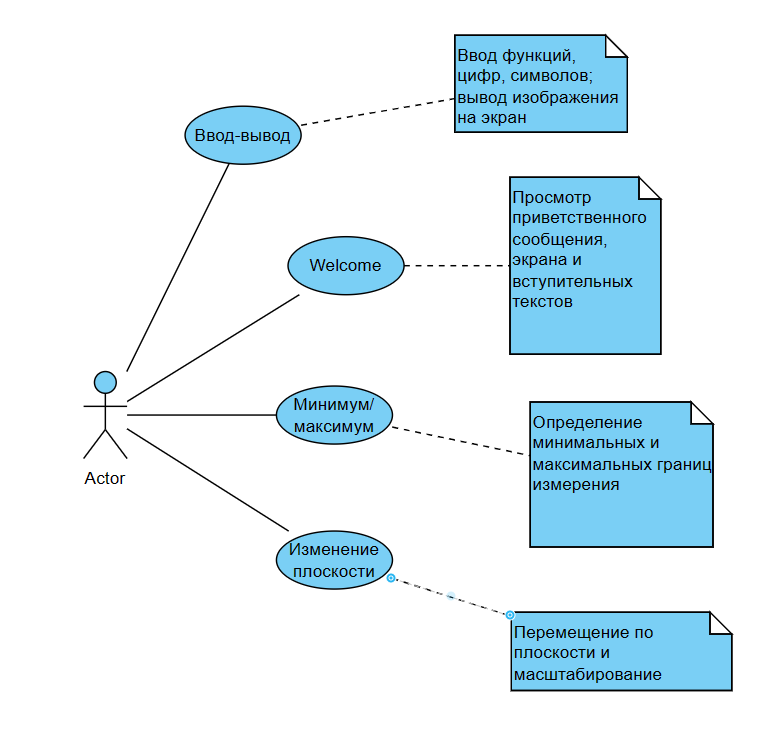


Рисунок А – варианты использования

# Приложение Б

*(обязательное)*

**Диаграмма деятельности**

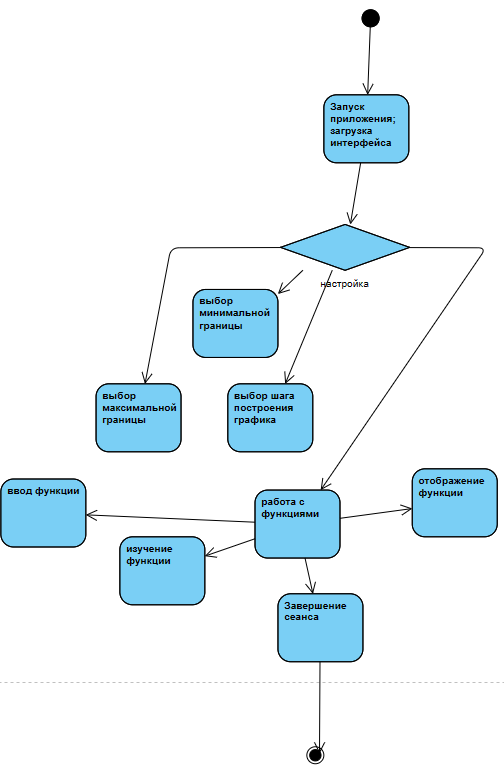


Рисунок Б – деятельность

# Приложение В

## *(обязательное)*

**Диаграмма классов**

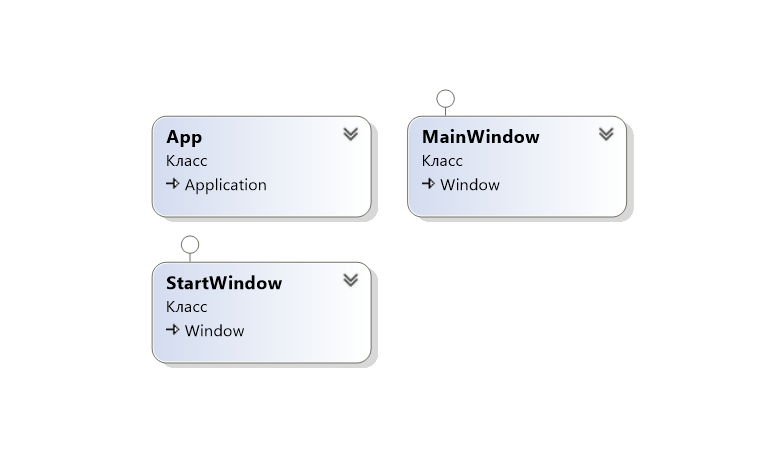
****

Рисунок В – классы