**Содержание**

Введение

1. Анализ предметной области

2. Проектирование приложения

3. Разработка программного обеспечения

3.1) Описание технологического стека разработки

3.2) Описание алгоритма работы

3.3) Описание интерфейса пользователя

4. Тестирование приложения.

a) План тестирования

b) Оценка результатов проведения тестирования

5. Заключение

6. Список литературы

Приложения

# 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.

# 1 Анализ предметной 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.

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.

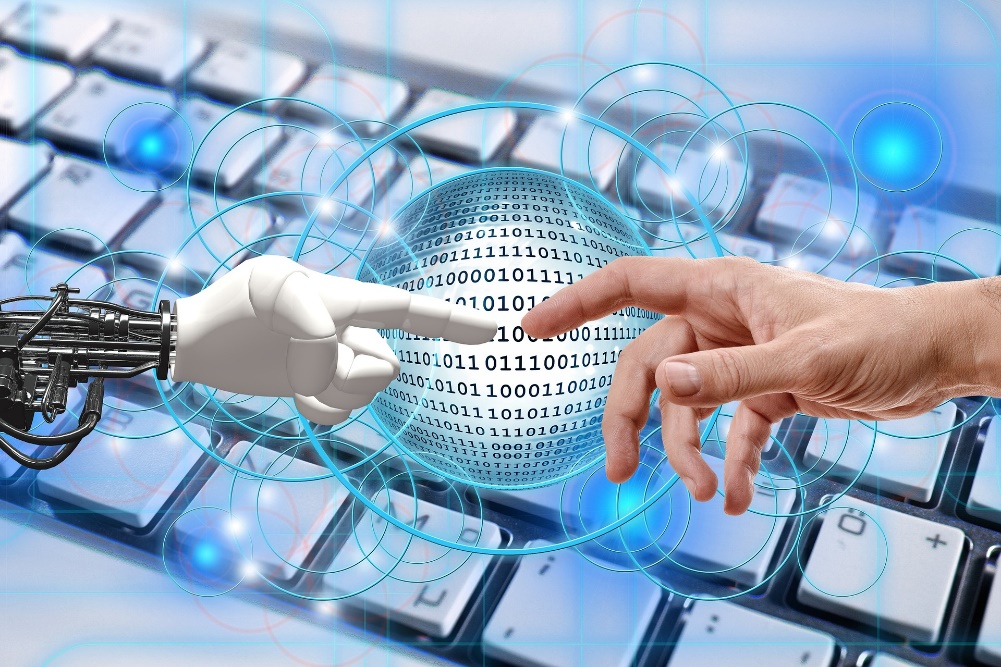


Рисунок 1 - автоматизация

# 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.

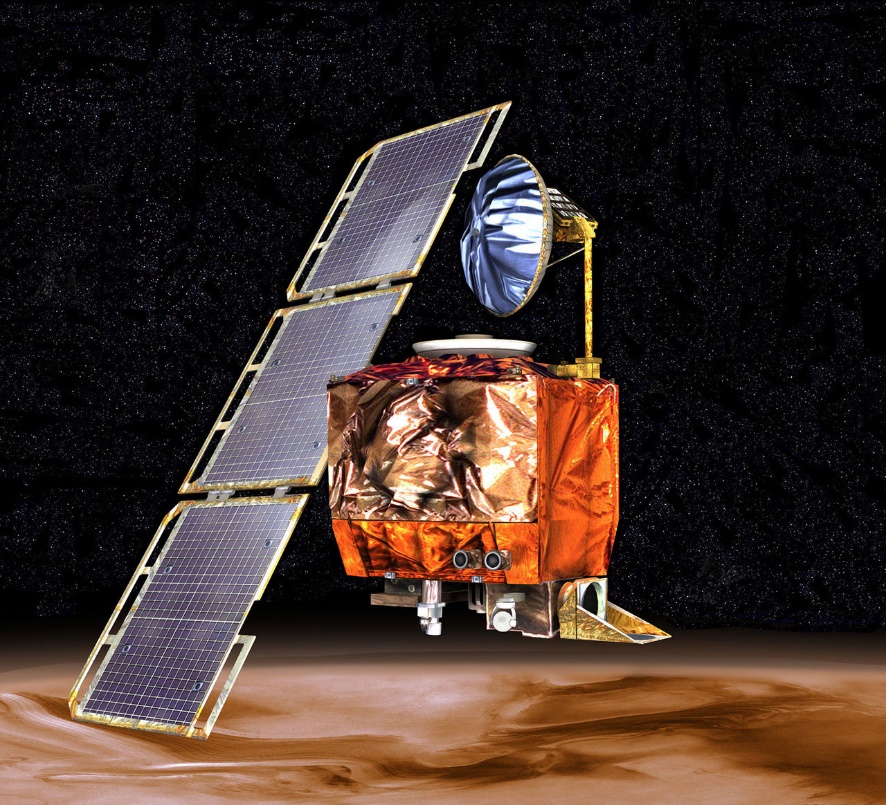


Рисунок 2 – лендер

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;
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 .

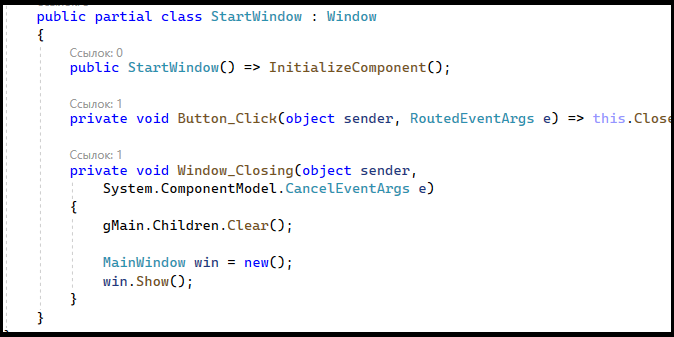


Рисунок 3 – приветствие

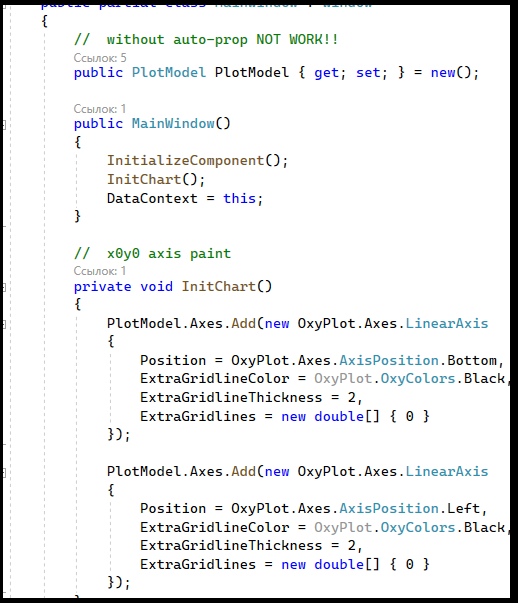


Рисунок 4 – прощание



Рисунок 5 – главный экран

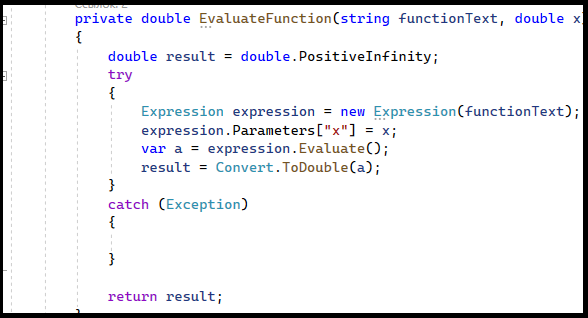


Рисунок 6 - помилование

## 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.

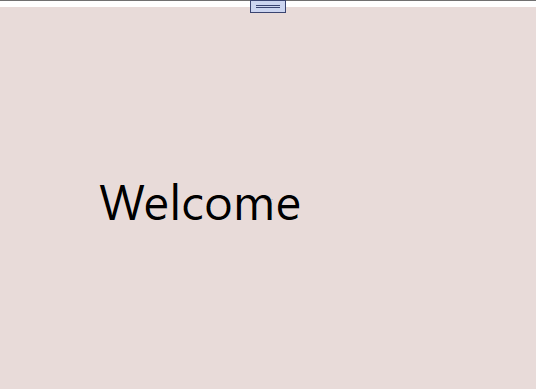


Рисунок 7 – приветственный экран

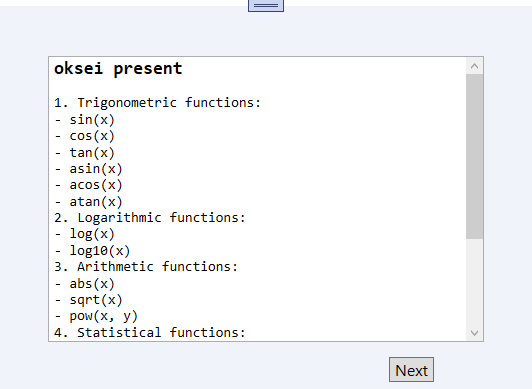


Рисунок 8 – лицензионное соглашение

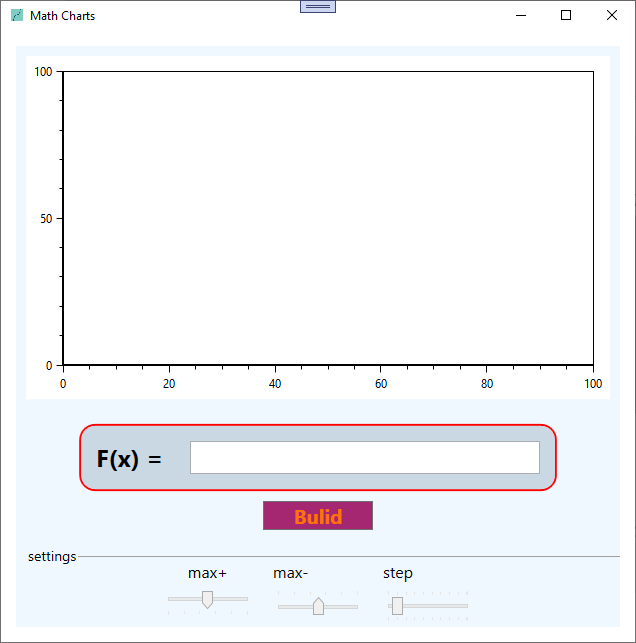


Рисунок 9 – главное окно

**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.

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:

* делать колонку номер1 таблицы;
* делать колонку номер2 таблицы;
* делать остальные колонки таблицы;
* составить таблицу;
* готово.

## Оценка результатов проведения тестирования

Результаты проведенных тестовых сценариев можно наблюдать в таблице 1.

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 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 результата | Функция: x\*x+9.2; границы: (1,-1) | Полноценный график. При приближении имеется возможность детального изучения. | - | fix |
| 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/](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%25D/) (дата обращения19.12.2023)