**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 2](#_Toc177754109)

[1 Постановка задачи 4](#_Toc177754110)

[1.1 Анализ предметной области 4](#_Toc177754111)

[1.2 Техническое задание на разработку ГОСТ 19.201-78 8](#_Toc177754112)

[2 Проектирование 13](#_Toc177754113)

[2.1 Функциональная структура программы 13](#_Toc177754114)

[2.2 Схема модулей программы 14](#_Toc177754115)

[2.3 Паспорта основных модулей 14](#_Toc177754116)

[3 Руководство оператора ГОСТ 19.505-79 19](#_Toc177754117)

[4 Тестирование программы 23](#_Toc177754118)

[4.1 Программа, методика и результаты испытаний ГОСТ 19.301-79 23](#_Toc177754119)

[4.2 Протокол испытаний 26](#_Toc177754120)

[ЗАКЛЮЧЕНИЕ 27](#_Toc177754121)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc177754122)

[ПРИЛОЖЕНИЯ 30](#_Toc177754123)

# ВВЕДЕНИЕ

Цель данного курсового проекта заключается в разработке программы для построения графиков математических функций, которая позволяет пользователю визуализировать зависимости между переменными, управлять параметрами графика и сохранять результаты для дальнейшего анализа.

Построение графиков является неотъемлемой частью анализа данных и помогает лучше понять изменения переменных, зависимости величин и их поведение при изменении различных параметров. В современных условиях развития науки и образования возникает потребность в доступных инструментах для учебного процесса и анализа данных, которые могли бы заменить громоздкие и дорогостоящие специализированные пакеты. Эти решения зачастую избыточны для простых образовательных и аналитических задач, что требует создания легкого в использовании и доступного инструмента.

На сегодняшний день существует немало программных решений для построения графиков, однако они либо ограничены в функционале, либо сложны для использования в учебном процессе, что снижает их применимость. Для удовлетворения данной потребности проект направлен на создание специализированного приложения, отвечающего требованиям доступности и простоты в эксплуатации.

Новизна и практическая ценность проекта заключается в разработке инструмента, способного обеспечить доступную и гибкую визуализацию функций, что делает его полезным для учебных и научных задач. Программа ориентирована на студентов и преподавателей, работающих с математическими и инженерными дисциплинами, и может быть использована в образовательных учреждениях, что позволит эффективно проводить занятия, готовить учебные материалы и облегчить анализ математических зависимостей.

Цель проекта – систематизация и актуализация знаний по программированию и проектированию, а также овладение методами научного исследования для создания программы построения графиков. Проект способствует развитию навыков самостоятельной работы и творческой инициативы, что является важной частью профессиональной подготовки.

**Задачи, решаемые проектом:**

* систематизация и актуализация знаний по программированию и проектированию;
* применение теоретических знаний на практике при разработке программного обеспечения;
* овладение современными методами поиска, обработки и представления информации;
* развитие навыков самостоятельного решения задач, а также ответственности в выполнении профессиональных обязанностей.

Для реализации проекта используются язык программирования C# и технология Windows Presentation Foundation (WPF), которая обеспечивает гибкость и широкие возможности для создания графических интерфейсов и построения визуализаций. WPF позволяет реализовать масштабируемые интерфейсы, которые необходимы для построения и отображения графиков, а также обеспечивает удобный функционал для взаимодействия с пользователем.

# Постановка задачи

# Анализ предметной области

Построение графиков математических функций – важный инструмент для визуализации и анализа функциональных зависимостей. Математические функции используются в различных областях знаний, таких как физика, экономика, инженерия, компьютерные науки и многих других. Визуализация позволяет получить общее представление о поведении функции на определенных промежутках и оценить ключевые характеристики, такие как точки экстремума, пересечения с осями координат, асимптоты и другие особенности.

Графическое представление функций упрощает анализ поведения функций, позволяя пользователям легче понять сложные математические выражения. Это особенно важно при обучении студентов, которым необходимо не только выучить теоретические аспекты математических функций, но и научиться применять их для решения задач. В учебных курсах по математике и физике построение графиков используется для демонстрации принципов изменения зависимых переменных относительно независимых.

Например, при изучении таких функций, как синус, косинус, экспонента, логарифмы, квадратичные и кубические уравнения, их графическое представление облегчает понимание того, как параметры функций влияют на форму графика. При этом не всегда можно получить аналитическое представление о поведении функции, особенно когда уравнения становятся сложными или содержат переменные, которые могут принимать широкий диапазон значений. В таких случаях построение графиков играет ключевую роль, так как позволяет выявить общие тенденции и поведение функций.

На современном этапе развития технологий существует множество программных продуктов для построения графиков функций. Наиболее популярными являются такие инструменты, как:

* **matlab** – мощная среда для численных вычислений, которая позволяет строить как простые графики, так и выполнять сложные симуляции. Тем не менее, matlab является коммерческим продуктом, который требует значительных финансовых затрат, что ограничивает его доступность для учебных заведений и студентов;
* **wolfram mathematica** – программа для математических вычислений и визуализации данных. Она предоставляет широкий набор инструментов для работы с математическими функциями, графиками и моделями. Однако, как и matlab, она относится к коммерческому программному обеспечению с довольно высокой стоимостью лицензий, что может ограничить ее использование в образовательных учреждениях;
* **geogebra** – бесплатное приложение, предназначенное для построения графиков и решения математических задач. Это достаточно популярный инструмент в образовательной среде, однако его возможности ограничены для профессиональных нужд или выполнения более сложных задач. Программа, несмотря на свою простоту, может быть перегружена лишними функциями для конечного пользователя, которому нужно только построение графиков;
* **desmos** – онлайн-калькулятор, который позволяет строить графики и анализировать функции. Desmos широко используется в учебных заведениях и имеет простой и интуитивно понятный интерфейс. Однако его работа зависит от наличия подключения к интернету, что может быть ограничивающим фактором при использовании на занятиях без доступа в сеть.

Таким образом, на рынке существует ряд инструментов для построения графиков, однако они либо имеют высокую стоимость, либо требуют подключения к интернету, либо включают излишние для определенных задач функции, что делает их не всегда удобными для простых пользователей, которым нужно быстрое и простое решение.

Для решения этих проблем необходимо разработать программное обеспечение, которое будет не только эффективным и доступным, но и простым в использовании. Ключевыми требованиями к такому ПО являются:

* **интуитивно понятный интерфейс** – программа должна иметь простой и логически структурированный интерфейс. Это особенно важно для студентов и преподавателей, у которых может не быть глубоких знаний в программировании или сложных математических пакетах;
* **поддержка различных типов функций** – программа должна поддерживать возможность построения графиков для полиномиальных, тригонометрических, логарифмических и других типов функций. Она также должна корректно обрабатывать сложные выражения с использованием нескольких математических операций и функций;
* **интерактивное взаимодействие** – пользователи должны иметь возможность изменять параметры графиков, например, масштабировать или перемещать график, чтобы лучше исследовать его поведение на разных интервалах;
* **минимальные системные требования** – программа должна быть легкой в установке и не требовать значительных ресурсов компьютера, чтобы работать на обычных учебных компьютерах или ноутбуках;
* **автономная работа** – ПО должно быть доступно для использования в автономном режиме, без необходимости подключения к интернету, что обеспечит его работу в любой среде, в том числе в классах или лабораториях без доступа к сети;
* **возможность сохранения результатов** – важно предусмотреть возможность сохранения построенных графиков в виде изображений или других форматов, которые можно использовать для дальнейшего анализа или включения в учебные материалы.

Язык программирования C# в сочетании с технологией WPF (Windows Presentation Foundation) предоставляет идеальные возможности для создания приложения, удовлетворяющего вышеописанным требованиям. C# является популярным языком программирования с широким набором библиотек для работы с математикой и графикой, что делает его отличным выбором для разработки приложений, требующих высокой производительности и надежности.

WPF, в свою очередь, предоставляет возможности для создания современных графических интерфейсов с поддержкой масштабируемой векторной графики, что особенно важно при работе с построением графиков. Одним из ключевых преимуществ WPF является его гибкость, которая позволяет разработчику легко настраивать внешний вид приложения и взаимодействие пользователя с графическим интерфейсом. Кроме того, WPF поддерживает современные технологии, такие как DirectX, что обеспечивает высокую производительность при работе с графикой.

Применение C# и WPF для создания программы для построения графиков функций также позволяет легко интегрировать дополнительные возможности, такие как сохранение графиков, а также их экспорт в различные форматы (например, PNG или SVG), что важно для пользователей, которым требуется дальнейшая обработка или включение этих данных в отчетные документы.

Таким образом, анализ предметной области показывает, что существует необходимость в разработке простого и эффективного приложения для построения графиков математических функций, которое будет доступно для широкого круга пользователей, особенно в образовательной среде. Использование C# и WPF обеспечивает оптимальные условия для создания подобного приложения, которое сможет удовлетворить потребности как студентов, так и преподавателей.

# Техническое задание на разработку ГОСТ 19.201-78

# Введение

Для успешной реализации программы, направленной на построение графиков математических функций, необходимо четкое определение всех требований и характеристик, которые она должна удовлетворять. В рамках курсового проекта разработка программы на языке C# с использованием технологии WPF требует формализации всех технических аспектов, что обеспечит создание корректного и удобного инструмента для конечного пользователя.

Данное техническое задание направлено на детальное описание функций и возможностей программы, а также требований к её интерфейсу и работе. Основной целью является создание программы, визуализировать их в виде графиков, а также управлять графиками с помощью инструментов масштабирования, перемещения и сохранения. Разработка программы будет выполнена в соответствии с государственными стандартами ГОСТ 19.201-78, что обеспечит правильное оформление и структуру программной документации.

# Основания для разработки

Основанием для разработки является задание на выполнение курсового проекта по теме «Программа для построения графиков функций» по МДК 01.01 РПМ под руководством Мачневой Екатерины Александровны.

# Назначение для разработки

Основное назначение программы – упростить процесс построения и изучения графиков математических функций, предоставив удобный и интуитивно понятный интерфейс. Программа должна быть легкой в использовании, требовать минимальных вычислительных ресурсов и работать автономно, без необходимости подключения к интернету. Она также должна поддерживать масштабирование и перемещение графиков, а также сохранять результаты в виде изображений, что позволит использовать их в дальнейшем для анализа, создания отчетов или учебных материалов.

Программа будет ориентирована на образовательные учреждения и студентов, нуждающихся в доступном инструменте для работы с функциями, а также на преподавателей, использующих её для демонстрации учебного материала.

# Требования к программе или программному изделию

* + - 1. Выбор математических функций

Программа должна предоставлять возможность выбора математических выражений, включающих следующие типы функций:

* полиномиальные функции;
* тригонометрические функции;
* экспоненциальные и логарифмические функции;
* композиции функций.
  + - 1. Построение графиков

Программа должна отображать график функции на координатной плоскости с возможностью точного воспроизведения всех выбранных выражений:

* диапазон построения графиков по умолчанию должен быть от -100 до 100 по обеим осям (x и y), с возможностью настройки пользователем.
  + - 1. Интерактивное управление графиком

Пользователь должен иметь возможность:

* **масштабировать график** с использованием колеса прокрутки мыши или специальных кнопок;
* **перемещать график** по координатной плоскости с помощью мыши.

# Требования к программной документации

Программа должна быть снабжена руководством оператора и методикой испытания, которые включают в себя:

* назначение программы;
* условия выполнения программы;
* выполнение программы;
* сообщения оператору;
* объект испытаний;
* цель испытаний;
* требования к программе;
* требования к программной документации;
* состав и порядок испытаний;
* методы испытаний.

# Технико-экономические показатели

# Технические показатели

1. **Совместимость с различными операционными системами:**

Программа разрабатывается для платформы Windows (версии 7 и выше), что обеспечивает её работу на большинстве компьютеров, используемых в образовательных и учебных учреждениях, а также на персональных устройствах студентов и преподавателей.

1. **Минимальные системные требования:**

* операционная система: Windows 7 и выше;
* оперативная память: от 4 ГБ;
* процессор: двухъядерный с тактовой частотой 2.0 ГГц и выше;
* свободное место на диске: не менее 300 МБ для установки программы и сохранения результатов.

Программа будет эффективно работать на широком диапазоне современных компьютеров, что снижает необходимость закупки нового оборудования для её использования.

1. **Производительность:**

Программа разработана с учётом минимальных задержек при построении графиков функций. Время построения графика от момента выбора функции до её отображения не превышает 1 секунды, что позволяет пользователям эффективно и быстро анализировать результаты.

1. **Безопасность и автономность:**

Программа не требует подключения к интернету для работы, что повышает её надёжность и безопасность. Все операции (выбор функции, построение графика, сохранение результатов) выполняются локально на компьютере пользователя.

1. **Обслуживание и поддержка:**

Программа не требует сложного обслуживания и может быть обновлена при необходимости добавления новых функций или исправления ошибок. Она не требует постоянного сопровождения специалистов, что снижает расходы на её эксплуатацию.

# Экономические показатели

1. **Низкие затраты на разработку:**

Разработка программы на языке C# с использованием технологии WPF не требует дополнительных финансовых вложений в лицензионное программное обеспечение, так как инструменты разработки (Visual Studio Community) и библиотеки являются бесплатными. Это значительно снижает общие затраты на разработку.

1. **Доступность для образовательных учреждений:**

Программа не требует приобретения платных лицензий, что делает её доступной для массового использования в образовательных учреждениях без дополнительных затрат. Это особенно важно для учебных заведений с ограниченным бюджетом. **Минимальные эксплуатационные расходы**  
Программа не требует дополнительных аппаратных или программных ресурсов, что позволяет использовать её на существующих компьютерах без необходимости обновления оборудования. Это снижает затраты на внедрение программы в учебный процесс.

1. **Экономия времени и ресурсов преподавателей и студентов:**

Использование программы позволяет значительно сократить время, затрачиваемое на построение графиков функций вручную или с помощью более сложных математических пакетов. Это приводит к повышению эффективности учебного процесса и снижению нагрузки на преподавателей.

1. **Потенциал для дальнейшего развития:**

Разработка программы с открытой архитектурой позволяет в будущем расширять её функциональные возможности без значительных затрат. Например, можно добавить новые типы функций, улучшить графический интерфейс или интегрировать программу с другими образовательными платформами, что обеспечивает долгосрочную экономическую выгоду.

# Стадии и этапы разработки

* предварительный анализ и тестирование;
* проектирование;
* программирование;
* тестирование;
* документирование.

# Порядок контроля и приемки

* предварительный контроль;
* функциональное тестирование;
* тестирование надежности;
* испытания;
* приемка;
* контроль;
* документация.

# Проектирование

# Функциональная структура программы

В данном курсовом проекте используются следующие элементы (все элементы показаны на рисунке 1):

* элемент для выбора функции, то есть ListBox;
* элемент параметров, то есть DataGrid;
* элемент области построения, то есть TextBox;
* элемент для построения графика, то есть Button
* элемент для визуального вывода график, то есть библиотека Oxy: PlotView.

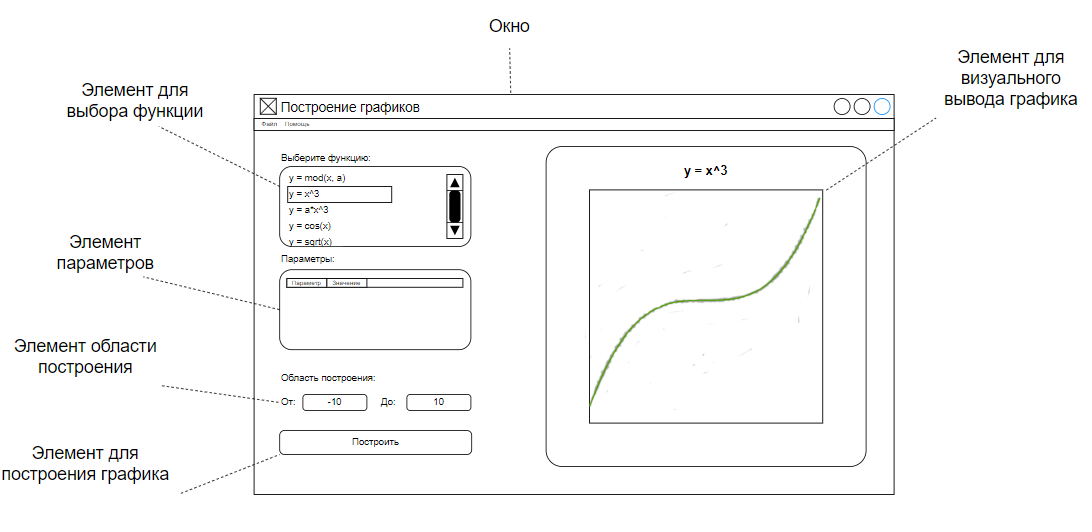


Рисунок 1 – Структура программы

# Схема модулей программы

Для создания схемы Я использовал структурные карты Констайна, чтобы продемонстрировать, каким образом программный продукт выполняет системные требования (показано на рисунке 2).

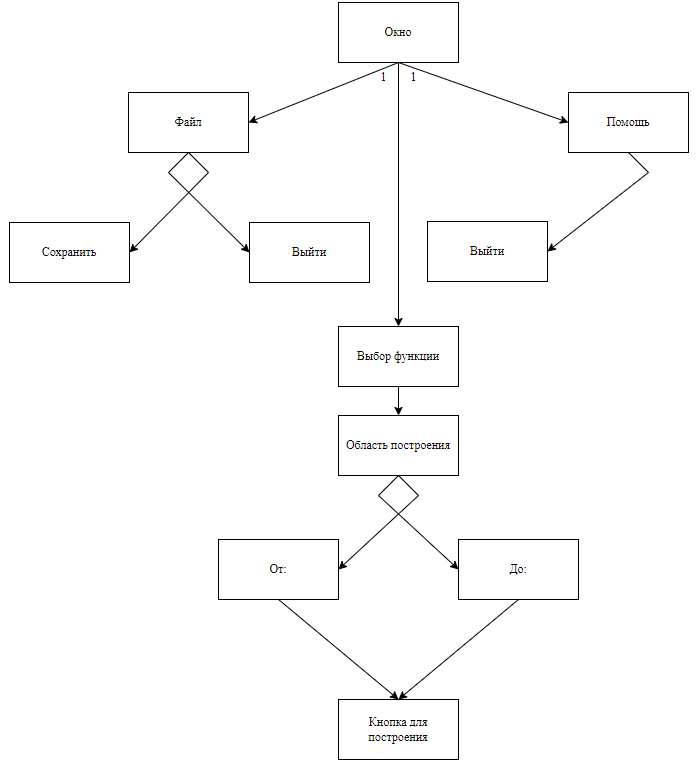


Рисунок 2 – Схема модулей

# Паспорта основных модулей

Модуль: MainWindow

**Описание:**

Модуль MainWindow является основным окном приложения, отвечающим за взаимодействие пользователя с программой. В этом модуле реализованы функции для построения графиков математических функций, управления параметрами функций, сохранения графиков и обработки событий интерфейса.

**Основные функции:**

1. **Конструктор** MainWindow()
   * осуществляет инициализацию окна программы, а также настройку интерфейса.
2. **Функция** SaveFile\_Click()
   * описание: реализует возможность сохранения графиков в формате PNG. Пользователь выбирает путь сохранения файла через диалоговое окно, после чего график экспортируется в выбранный файл;
   * входные данные: нажатие на кнопку сохранения;
   * выходные данные: сохранённый файл PNG с графиком;
   * библиотеки: использует библиотеку OxyPlot для экспорта графика.
3. **Функция** Exit\_Click()
   * описание: закрывает приложение;
   * входные данные: нажатие на кнопку выхода;
   * выходные данные: завершение работы программы
4. **Функция** About\_Click()
   * описание: показывает окно с информацией о программе;
   * входные данные: нажатие на кнопку "О программе";
   * выходные данные: информационное окно.
5. **Функция** InitializeParameterGrid()
   * описание: создаёт и инициализирует таблицу параметров функций для последующего изменения пользователем. Каждая строка таблицы представляет собой параметр (например, a, b, c) и его значение;
   * входные данные: инициализация приложения;
   * выходные данные: созданная таблица параметров.
6. **Функция** ChoiceListBox\_SelectionChanged()
   * описание: обрабатывает выбор функции в списке функций и обновляет таблицу параметров в зависимости от выбранной функции;
   * входные данные: событие выбора функции;
   * выходные данные: обновление параметров в таблице;
7. **Функция** BuildButton\_Click()
   * описание: строит график функции в зависимости от выбранной функции и параметров. Пользователь вводит диапазон X (от и до), а программа строит график;
   * входные данные: пиапазон X, выбранная функция, параметры функции;
   * выходные данные: построенный график.
8. **Функция** PlotGraph()
   * описание: визуализирует график на основе переданных параметров функции. Добавляет точки на график, создаёт модель графика и выводит его в компонент PlotView;
   * входные данные: название функции, параметры (a, b, c), диапазон значений X;
   * выходные данные: отображение графика.
9. **Функция** CalculateFunction()
   * описание: вычисляет значения функции для построения графика. Поддерживает различные математические функции;
   * входные данные: название функции, значения X, параметры a, b, c;
   * выходные данные: значение функции в точке X.
10. **Функция** GetParameter()
    * описание: получает значение параметра из таблицы параметров. Если значение не задано, возвращает значение по умолчанию;
    * входные данные: Название параметра (a, b, c);
    * выходные данные: Значение параметра.
11. **Функции для валидации ввода:**
    * textBox\_PreviewTextInput() – обрабатывает ввод символов в текстовые поля и фильтрует недопустимые символы;
    * textBox\_PreviewKeyDown() – предотвращает вставку текста через комбинацию клавиш Ctrl+V;
    * isTextAllowed() – проверяет, являются ли введённые символы допустимыми (цифры и знак минуса).

**Связи с другими модулями:**

* модуль использует библиотеку OxyPlot для построения и отображения графиков;
* использует стандартные компоненты WPF для построения интерфейса: ListBox, DataGrid, TextBox, Button, PlotView и т.д.

Модуль: OxyPlot.Wpf

**Описание:**  
OxyPlot.Wpf – это внешний модуль, который отвечает за визуализацию графиков в окне WPF. Он предоставляет объект PlotView, в котором можно отображать графики, а также методы для их экспорта в различные форматы.

**Основные функции:**

* plotView – компонент, в котором отображается график;
* pngExporter – объект для экспорта графика в формат PNG. Используется при сохранении графика.

Модуль: SaveFileDialog

**Описание:**  
Модуль SaveFileDialog – стандартное средство Windows для сохранения файлов. Применяется в функции сохранения графика, позволяя пользователю выбрать путь для сохранения файла и задать имя файла.

**Основные функции:**

* выбор файла для сохранения;
* фильтрация файлов по типу (в данном случае PNG).

Модуль: DataGrid

**Описание:**  
DataGrid используется для отображения параметров функции, таких как a, b и c. Это позволяет пользователю легко изменять параметры функции перед её построением.

**Основные функции:**

* отображение таблицы параметров;
* взаимодействие с параметрами функции через интерфейс.

# Руководство оператора ГОСТ 19.505-79

# Назначение программы

Программа предназначена для выполнения следующих задач:

1. **Визуализация математических функций:**

Программа предоставляет пользователю возможность выбирать различные математические функции (линейные, полиномиальные, тригонометрические, логарифмические, экспоненциальные и другие) и визуализировать их в виде графиков на двумерной координатной плоскости. Это позволяет пользователю наглядно изучать поведение функций в различных диапазонах значений.

1. **Анализ функций:**

Используя программу, пользователи могут исследовать особенности графиков функций: экстремумы, точки пересечения с осями, области возрастания и убывания. Визуализация графиков помогает лучше понимать свойства функций и их применение в решении различных задач.

1. **Интерактивное управление графиками:**

Программа позволяет масштабировать и перемещать графики, что упрощает изучение их поведения на разных участках. Пользователь может детально рассматривать отдельные части графика, приближая и отдаляя его, а также изменять диапазон значений.

1. **Образовательные цели:**

Программа разработана с целью облегчить процесс обучения математике и смежным дисциплинам. Она предназначена для студентов, преподавателей и научных сотрудников, которым необходимо быстро и наглядно строить графики функций для анализа и демонстрации на лекциях или практических занятиях.

1. **Сохранение и использование результатов:**

Программа предоставляет возможность сохранять графики в виде изображений (например, в форматах PNG или JPEG) для дальнейшего использования в научных работах, презентациях, учебных пособиях или личных заметках.

# Условия выполнения программы

* + 1. Системные требования

Для успешной работы программы оборудование должно соответствовать следующим минимальным характеристикам:

* **операционная система:** Windows 7 и выше;
* **процессор:** Двухъядерный с частотой 2.0 ГГц;
* **оперативная память:** 4 ГБ;
* **свободное место на диске:** 100 МБ;
* **графический адаптер:** Поддержка DirectX 11 и выше.

# ****Программные зависимости****

* **среда выполнения .NET:** Необходима установка .NET Framework 8.0;
* **поддержка WPF:** Программа требует возможности рендеринга графики через WPF, которая должна поддерживаться установленной операционной системой и графическими драйверами.

# ****Экран и разрешение****

Программа рассчитана на использование с экранами, поддерживающими минимальное разрешение 1280x720. Для более удобной работы рекомендуется использовать дисплеи с разрешением 1920x1080 и выше.

# ****Требования к пользователю****

* пользователь должен иметь базовые знания математики (линейные, тригонометрические, полиномиальные и другие);
* программа использует математическую нотацию, поэтому пользователь должен быть знаком с синтаксисом выбора математических выражений.

# ****Рабочие условия****

Программа работает в автономном режиме, не требуя подключения к интернету. Однако для получения обновлений или в случае обращения за поддержкой пользователю может понадобиться интернет-соединение.

# ****Инсталляция и запуск****

* для установки программы требуется наличие прав администратора на компьютере;
* инсталляция программы происходит через запуск установочного файла (MSI или EXE), после чего программа будет доступна через меню «Пуск» или ярлык на рабочем столе.

# ****Совместимость****

Программа разработана для работы на операционных системах семейства Windows (Windows 7 и выше), а также поддерживает различные аппаратные конфигурации при наличии соответствующих драйверов для видеокарты.

# ****Конфиденциальность****

Программа не требует доступа к личным данным пользователя и не передаёт данные через интернет, что исключает риск утечки конфиденциальной информации.

# ****Обновления****

Обновления программы устанавливаются вручную или через встроенный механизм автоматического обновления (при наличии).

# Выполнение программы

Выполнение программы:

* **запуск исполняемого файла;**
* **проверка системных ресурсов;**
* **обработка введённых данных;**
* **построение графика;**
* **интерактивное взаимодействие;**
* **сохранение результатов;**
* **освобождение ресурсов.**

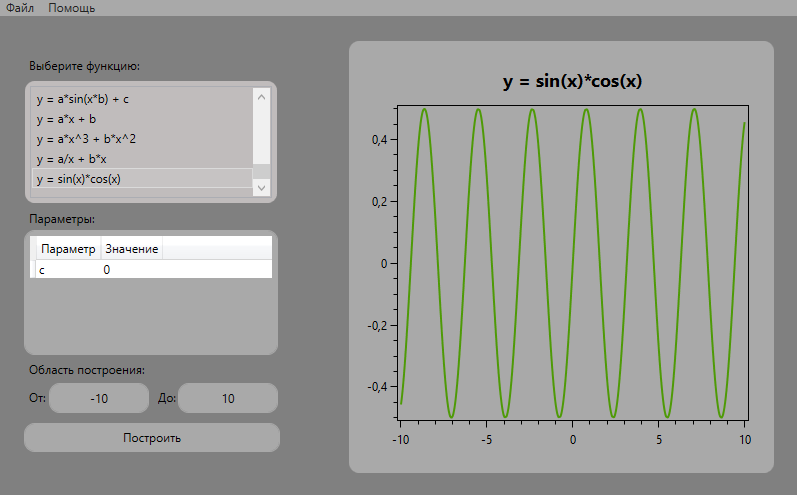


Рисунок 3 – Выполнение программы

# Сообщения оператору

Не предусмотрено в программе.

# Тестирование программы

# Программа, методика и результаты испытаний ГОСТ 19.301-79

# Объект испытаний

Объектом испытаний является программное обеспечение для построения графиков математических функций. Данное ПО позволяет визуализировать различные функции с заданными параметрами и сохранять их в виде изображений (формат PNG). Программа предоставляет пользователю возможности взаимодействия с графическим интерфейсом для ввода параметров функции, выбора диапазонов значений по оси X, а также выбора конкретной функции для построения.

# Цель испытаний

Целью испытаний программного обеспечения для построения графиков является проверка его работоспособности, корректности выполнения основных функций и соответствие заявленным требованиям. Испытания направлены на выявление потенциальных ошибок, проверку надежности и производительности программы, а также на оценку удобства использования интерфейса.

# Требования к программе

1. **Построение графиков математических функций:**

* программа должна предоставлять возможность построения графиков различных математических функций;
  + пользователь должен иметь возможность изменять диапазон значений оси X для построения графика.

1. **Интерактивный интерфейс:**

* программа должна иметь интуитивно понятный интерфейс, позволяющий пользователю легко выбирать тип функции из списка, задавать параметры и диапазоны значений;
* должна быть предусмотрена функция интерактивного изменения параметров с немедленным обновлением графика.

1. **Сохранение графиков:**

* программа должна поддерживать сохранение построенных графиков в формате PNG с возможностью указания местоположения файла;
* сохраненные графики должны быть высокого качества и сохранять все элементы, заданные пользователем (заголовки, линии графика, координатные оси).

1. **Отображение параметров функций:**

* при выборе конкретной функции программа должна автоматически выводить параметры, которые можно настроить для этой функции;
* данные должны быть представлены в виде таблицы с полями для ввода значений.

# Требования к программной документации

Документация должна содержать:

* техническое задание;
* руководство оператора;
* программа, методика и результаты испытаний.

# Состав и порядок испытаний

Состав испытаний:

* + - * **функциональное тестирование;**
      * **интерфейсное тестирование;**
      * **тестирование сохранения данных;**
      * **тестирование совместимости;**
      * **тестирование производительности;**
      * **тестирование на обработку ошибок.**

Порядок испытаний:

* + - * **подготовительный этап;**
      * **тестирование интерфейса и взаимодействия с пользователем;**
      * **тестирование функциональности программы;**
      * **тестирование сохранения графиков;**
      * **тестирование производительности;**
      * **тестирование совместимости и стабильности;**
      * **документирование результатов.**

# Методы испытаний

Методы:

* функциональное тестирование;
* тестирование пользовательского интерфейса (UI);
* тестирование производительности;
* тестирование совместимости;
* тестирование устойчивости к ошибкам (тестирование отказоустойчивости);
* ручное и автоматизированное тестирование;
* регрессионное тестирование.

# Протокол испытаний

Таблица 1 – Тест-кейс к программе

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Предусловие** | **Шаг** | **Ожидаемый результат** | **Фактический результат** | **Итог** |
| 1 | Запуск программы | Программа запущена | Запустить программу | Программа запущена | Программа запущена | + |
| 2 | Кнопка «Сохранить» | Программа запущена | Нажать на кнопку | Сохранение | Сохранение | + |
| 3 | Кнопка «Выход» | Программа запущена | Нажать на кнопку | Выход | Выход | + |
| 4 | Кнопка о программе | Программа запущена | Нажать на кнопку | О программе | О программе | + |
| 5 | Выбор функции | Программа запущена | Выбрать функцию | Функция выбрана | Функция выбрана | + |
| 6 | Изменение параметров функции | Программа запущена | Изменить параметр | Параметр изменен | Параметр неизменен | - |
| 7 | Задать область построения | Программа запущена | Задать область построения | Область построения задана | Область построения задана | + |
| 8 | Кнопка «Построить» | Программа запущена | Нажать на кнопку | Построение | Построение | + |
| 9 | Защита от некорректных значений | Некорректные данные заблокированы | - | - | - | + |

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данного курсового проекта была разработана программа для построения графиков функций с использованием технологий C# и WPF. Основной целью проекта было создание программного продукта, который позволил бы пользователю визуализировать математические функции на графике, а также обеспечить возможность динамического изменения параметров этих функций для дальнейшего построения графиков.

На начальных этапах работы была проведена детальная постановка задачи, которая включала анализ предметной области и технических требований к разрабатываемому приложению. В частности, было определено, что система должна поддерживать работу с широким спектром функций: линейные, нелинейные, тригонометрические и логарифмические функции. Кроме того, программа должна была предоставить пользователю возможность изменять параметры функций в реальном времени и пересчитывать график на основе выбранных данных.

Были сформированы требования к интерфейсу пользователя, который должен быть интуитивно понятным и удобным для работы. Важно было обеспечить минимальную нагрузку на пользователя в плане необходимости изучения функционала программы. Для этого в интерфейс были встроены такие элементы управления, как выпадающие списки для выбора функций, текстовые поля для ввода параметров и диапазонов значений по оси X, а также кнопки для быстрого построения и сохранения результатов работы.

В рамках этапа проектирования была выполнена разработка архитектуры программы, которая включает в себя несколько ключевых модулей. Основным из них является модуль графического интерфейса, обеспечивающий взаимодействие пользователя с программой, а также модуль расчёта значений функций и модуль визуализации графиков. Для построения графиков использовалась библиотека OxyPlot, которая предоставила необходимые инструменты для реализации графического компонента приложения.

Программа была спроектирована таким образом, чтобы поддерживать гибкое расширение функционала. Например, с помощью механизма обновления данных на форме можно легко добавить новые функции в программу без изменения основной логики приложения. Это позволяет в будущем значительно расширить спектр доступных для построения функций, удовлетворяя запросы более широкого круга пользователей.

Одним из важных аспектов программы является возможность сохранения графиков в формате PNG, что было успешно реализовано в ходе выполнения проекта. Это позволяет пользователям не только визуализировать функции в режиме реального времени, но и сохранять результаты для последующего использования в отчётах, учебных материалах или других целях. Таким образом, программа может быть полезна не только для студентов и преподавателей, но и для научных работников и инженеров, которым необходимо представлять математические модели в графической форме.

Важной частью работы также стало тестирование программы. В процессе тестирования были выявлены и устранены ошибки, возникающие при введении некорректных данных, а также были проведены проверки на устойчивость работы приложения при использовании большого количества точек для построения графика. Программа показала высокую стабильность и производительность, что является важным показателем для конечного продукта.

С точки зрения экономической эффективности, разработка данной программы может быть выгодной для учебных заведений и научных организаций, так как она позволяет автоматизировать процесс построения графиков функций, снижая затраты на ручную работу и повышая точность построений. Программа имеет открытый исходный код, что делает её доступной для широкого круга пользователей и облегчает её доработку и адаптацию под конкретные нужды.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* 1. **Громов, В. А.** Основы программирования на C#: Учебное пособие / В. А. Громов. — 2-е изд., испр. и доп. — М.: Юрайт, 2019. — 368 с.
  2. **Кузнецов, П. А., и Белоусов, А. С.** WPF для профессионалов. Разработка интерфейсов в среде .NET / П. А. Кузнецов, А. С. Белоусов. — СПб.: Питер, 2020. — 512 с.
  3. **Шилдт, Г.** Полный справочник по C# / Г. Шилдт. — 7-е изд. — М.: Вильямс, 2018. — 960 с.
  4. **Microsoft.** Документация по библиотеке OxyPlot [Электронный ресурс]. — URL: https://oxyplot.github.io/documentation (дата обращения: 03.10.2024).
  5. **Сайков, В. И.** Алгоритмы и структуры данных: Учебник / В. И. Сайков. — М.: Лаборатория знаний, 2021. — 430 с.
  6. **Стандарт ГОСТ Р 7.0.5-2021**. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. — М.: Стандартинформ, 2021. — 56 с.
  7. **Шеховцов, Д. Ю.** Методология тестирования программного обеспечения / Д. Ю. Шеховцов. — М.: Бином, 2017. — 320 с.
  8. **Павлов, Е. И.** Основы проектирования интерфейсов: Курс лекций / Е. И. Павлов. — М.: ДМК Пресс, 2019. — 276 с.
  9. **Панферов, И. В.** Современные подходы к разработке ПО на платформе .NET / И. В. Панферов. — СПб.: БХВ-Петербург, 2020. — 384 с.
  10. **Полторацкий, М. Н.** Тестирование и отладка программных продуктов: Практическое руководство / М. Н. Полторацкий. — М.: Юрайт, 2018. — 292 с.

# ПРИЛОЖЕНИЯ

**ПРИЛОЖЕНИЕ А**

MainWindow.xaml:

<Window x:Class="Построение\_графиков.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Построение\_графиков" xmlns:oxy="http://oxyplot.org/wpf"

mc:Ignorable="d"

Title="Построение графиков" Height="500" Width="800" ResizeMode="NoResize" WindowStartupLocation="CenterScreen" WindowStyle="None" Background="Gray" Icon="Resource/icon.png">

<DockPanel>

<Menu DockPanel.Dock="Top" Background="DarkGray">

<MenuItem Header="Файл" ToolTip="Файл">

<MenuItem Header="Сохранить" x:Name="SaveFile" Click="SaveFile\_Click" ToolTip="Сохранение графика в указанное место"/>

<Separator/>

<MenuItem Header="Выход" x:Name="Exit" Click="Exit\_Click" ToolTip="Выход из программы"/>

</MenuItem>

<MenuItem Header="Помощь" ToolTip="Раздел с информацией">

<MenuItem Header="О программе" x:Name="About" Click="About\_Click" ToolTip="Информация о программе"/>

</MenuItem>

</Menu>

<Grid Background="Gray">

<Label Content="Выберите функцию:" HorizontalAlignment="Left" Margin="25,36,0,0" VerticalAlignment="Top"/>

<Border CornerRadius="10" BorderBrush="Gray" BorderThickness="1" Background="#FFC0BCBC" Margin="25,64,521,294">

<ListBox x:Name="ChoiceListBox" SelectionChanged="ChoiceListBox\_SelectionChanged" Background="#FFC0BCBC" Margin="5,5,5,5" ToolTip="Выбор конкретной функции">

<ListBoxItem Content="y = mod(x, a)" />

<ListBoxItem Content="y = x^3" />

<ListBoxItem Content="y = a\*x^3" />

<ListBoxItem Content="y = cos(x)" />

<ListBoxItem Content="y = sqrt(x)" />

<ListBoxItem Content="y = x^a" />

<ListBoxItem Content="y = ln(x)" />

<ListBoxItem Content="y = arcsin(x)" />

<ListBoxItem Content="y = arccos(x)" />

<ListBoxItem Content="y = arctng(x)" />

<ListBoxItem Content="y = arcctg(x)" />

<ListBoxItem Content="y = log(x, a)" />

<ListBoxItem Content="y = 1/x" />

<ListBoxItem Content="y = ln(cos(x))" />

<ListBoxItem Content="y = sqrt(arctng(x))" />

<ListBoxItem Content="y = ln(sin(x))" />

<ListBoxItem Content="y = pow(exp(x), 2)" />

<ListBoxItem Content="y = exp(x)" />

<ListBoxItem Content="y = a\*x^2 + b\*x + c" />

<ListBoxItem Content="y = a\*sin(x\*b) + c" />

<ListBoxItem Content="y = a\*x + b" />

<ListBoxItem Content="y = a\*x^3 + b\*x^2" />

<ListBoxItem Content="y = a/x + b\*x" />

<ListBoxItem Content="y = sin(x)\*cos(x)" />

</ListBox>

</Border>

<Label Content="Параметры:" HorizontalAlignment="Left" Margin="25,189,0,0" VerticalAlignment="Top"/>

<Border CornerRadius="10" BorderBrush="DarkGray" BorderThickness="1" Background="DarkGray" Margin="25,214,521,143">

<DataGrid x:Name="ParametrDataGrid" AutoGenerateColumns="False" IsReadOnly="True" Margin="5,5,5,5" Background="DarkGray" BorderThickness="0" GridLinesVisibility="None" ToolTip="Параметр выбранной функции"/>

</Border>

<Label Content="Область построения:" HorizontalAlignment="Left" Margin="25,340,0,0" VerticalAlignment="Top"/>

<Label Content="От:" HorizontalAlignment="Left" Margin="25,368,0,0" VerticalAlignment="Top"/>

<Label Content="До:" HorizontalAlignment="Left" Margin="154,368,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="FromTextBox" HorizontalAlignment="Left" Margin="50,367,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="100" Height="30" Background="DarkGray" PreviewTextInput="TextBox\_PreviewTextInput" PreviewKeyDown="TextBox\_PreviewKeyDown" TextAlignment="Center" VerticalContentAlignment="Center" ToolTip="Область построения от">

<TextBox.Template>

<ControlTemplate TargetType="TextBox">

<Border Background="{TemplateBinding Background}" CornerRadius="10" BorderBrush="DarkGray" BorderThickness="1">

<ScrollViewer x:Name="PART\_ContentHost" />

</Border>

</ControlTemplate>

</TextBox.Template>

</TextBox>

<TextBox x:Name="BeforeTextBox" HorizontalAlignment="Left" Margin="179,367,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="100" Height="30" Background="DarkGray" PreviewTextInput="TextBox\_PreviewTextInput" PreviewKeyDown="TextBox\_PreviewKeyDown" TextAlignment="Center" VerticalContentAlignment="Center" ToolTip="Область построения до">

<TextBox.Template>

<ControlTemplate TargetType="TextBox">

<Border Background="{TemplateBinding Background}" CornerRadius="10" BorderBrush="DarkGray" BorderThickness="1">

<ScrollViewer x:Name="PART\_ContentHost" />

</Border>

</ControlTemplate>

</TextBox.Template>

</TextBox>

<Button x:Name="BuildButton" Click="BuildButton\_Click" Content="Построить" HorizontalAlignment="Left" Margin="25,407,0,0" VerticalAlignment="Top" Width="256" Background="DarkGray" Height="29" ToolTip="Расчет функции">

<Button.Template>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" CornerRadius="10" BorderBrush="DarkGray" BorderThickness="1">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Button.Template>

</Button>

<Border CornerRadius="10" BorderBrush="Transparent" Background="DarkGray" Margin="350,25,25,25">

<oxy:PlotView x:Name="PlotView" HorizontalAlignment="Center" Height="396" VerticalAlignment="Center" Width="406" Background="DarkGray" ToolTip="Область, где отображается функция"/>

</Border>

</Grid>

</DockPanel>

</Window>

**ПРИЛОЖЕНИЕ Б**

MainWindow.xaml.cs:

using Microsoft.Win32;

using OxyPlot;

using OxyPlot.Series;

using OxyPlot.Wpf;

using System;

using System.IO;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace Построение\_графиков

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

InitializeParameterGrid();

Height += 30;

Width += 30;

}

private void SaveFile\_Click(object sender, RoutedEventArgs e)

{

if (PlotView.Model != null)

{

SaveFileDialog dlg = new SaveFileDialog();

dlg.FileName = "График";

dlg.DefaultExt = ".png";

dlg.Filter = "PNG Files|\*.png";

Nullable<bool> result = dlg.ShowDialog();

if (result == true)

{

string filePath = dlg.FileName;

using (var stream = File.Create(filePath))

{

var pngExporter = new PngExporter { Width = 800, Height = 600 };

pngExporter.Export(PlotView.Model, stream);

}

MessageBox.Show("График успешно сохранён!", "Сохранение", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

else

{

MessageBox.Show("Нет графика для сохранения.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void Exit\_Click(object sender, RoutedEventArgs e)

{

Close();

}

private void About\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show("", "О программе", MessageBoxButton.OK, MessageBoxImage.Information);

}

private void InitializeParameterGrid()

{

ParametrDataGrid.Columns.Add(new DataGridTextColumn { Header = "Параметр", Binding = new System.Windows.Data.Binding("Name") });

ParametrDataGrid.Columns.Add(new DataGridTextColumn { Header = "Значение", Binding = new System.Windows.Data.Binding("Value") });

}

private void ChoiceListBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

var selectedItem = ChoiceListBox.SelectedItem as ListBoxItem;

if (selectedItem != null)

{

string selectedFunction = selectedItem.Content.ToString();

UpdateParameterGrid(selectedFunction);

}

}

private void UpdateParameterGrid(string selectedFunction)

{

ParametrDataGrid.Items.Clear();

if (selectedFunction.Contains("a"))

ParametrDataGrid.Items.Add(new { Name = "a", Value = 1.0 });

if (selectedFunction.Contains("b"))

ParametrDataGrid.Items.Add(new { Name = "b", Value = 1.0 });

if (selectedFunction.Contains("c"))

ParametrDataGrid.Items.Add(new { Name = "c", Value = 0.0 });

}

private void BuildButton\_Click(object sender, RoutedEventArgs e)

{

double fromX = double.Parse(FromTextBox.Text);

double toX = double.Parse(BeforeTextBox.Text);

var selectedItem = ChoiceListBox.SelectedItem as ListBoxItem;

if (selectedItem != null)

{

string selectedFunction = selectedItem.Content.ToString();

PlotGraph(selectedFunction, fromX, toX);

}

}

private void PlotGraph(string selectedFunction, double fromX, double toX)

{

double a = GetParameter("a", 1.0);

double b = GetParameter("b", 1.0);

double c = GetParameter("c", 0.0);

var plotModel = new PlotModel { Title = selectedFunction };

var lineSeries = new LineSeries();

for (double x = fromX; x <= toX; x += 0.01)

{

double y = CalculateFunction(selectedFunction, x, a, b, c);

lineSeries.Points.Add(new DataPoint(x, y));

}

plotModel.Series.Add(lineSeries);

PlotView.Model = plotModel;

}

private double CalculateFunction(string selectedFunction, double x, double a, double b, double c)

{

switch (selectedFunction)

{

case "y = x^3":

return Math.Pow(x, 3);

case "y = a\*x^3":

return a \* Math.Pow(x, 3);

case "y = cos(x)":

return Math.Cos(x);

case "y = sqrt(x)":

return Math.Sqrt(x);

case "y = x^a":

return Math.Pow(x, a);

case "y = ln(x)":

return Math.Log(x);

case "y = a\*x^2 + b\*x + c":

return a \* Math.Pow(x, 2) + b \* x + c;

case "y = a\*sin(x\*b) + c":

return a \* Math.Sin(b \* x) + c;

case "y = a\*x + b":

return a \* x + b;

case "y = sin(x)\*cos(x)":

return Math.Sin(x) \* Math.Cos(x);

default:

return 0;

}

}

private double GetParameter(string name, double defaultValue)

{

foreach (var item in ParametrDataGrid.Items)

{

var row = item as dynamic;

if (row.Name == name)

{

return double.Parse(row.Value.ToString());

}

}

return defaultValue;

}

private void TextBox\_PreviewTextInput(object sender, System.Windows.Input.TextCompositionEventArgs e)

{

TextBox textBox = sender as TextBox;

e.Handled = !IsTextAllowed(textBox, e.Text);

}

private void TextBox\_PreviewKeyDown(object sender, System.Windows.Input.KeyEventArgs e)

{

if (e.Key == Key.V && (Keyboard.Modifiers & ModifierKeys.Control) == ModifierKeys.Control)

{

e.Handled = true;

}

}

private static bool IsTextAllowed(TextBox textBox, string inputText)

{

foreach (char c in inputText)

{

if (char.IsDigit(c))

{

continue;

}

if (c == '-' && textBox.Text.Length == 0)

{

continue;

}

return false;

}

return true;

}

}

}

**ПРИЛОЖЕНИЕ В**

https://github.com/killdxspx/Cource-project-MDK-01.01.git