# Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству

## Анализ прототипов

Перед началом разработки важно изучить существующие аналогичные программные продукты. Это поможет понять, какие функции уже реализованы в других приложениях, и определить, какие из них можно использовать в качестве ориентира.

### Примеры прототипов:

Калькулятор Windows – стандартное приложение, которое выполняет базовые арифметические операции. Однако оно не поддерживает обработку сложных математических выражений и построение графиков.

GeoGebra – программа для визуализации математических функций и геометрических построений. Она поддерживает построение графиков, но имеет сложный интерфейс и избыточный функционал для задач курсовой.

Desmos – онлайн-калькулятор с возможностью построения графиков функций. Простой и интуитивно понятный интерфейс, но требует подключения к интернету.

Вывод: Необходимо разработать приложение, которое будет сочетать простоту интерфейса (как у калькулятора Windows) с возможностью обработки выражений и построения графиков (как в Desmos или GeoGebra).

## Литературные источники

### Основы программирования на Delphi:

Фаронов В. В. "Программирование на языке Delphi".

Канту М. "Delphi для начинающих".

Математические вычисления:

Корн Г., Корн Т. "Справочник по математике для научных работников и инженеров".

Бахвалов Н. С. "Численные методы".

### Построение графиков:

Павловская Т. А. "Программирование на языке Delphi".

Райтман М. А. "Разработка графических интерфейсов в Delphi".

## Формирование требований к программному средству

На основе анализа прототипов и изучения литературы сформированы требования к проектируемому программному средству.

### Функциональные требования

#### Обработка математических выражений:

* Поддержка базовых арифметических операций (+, -, \*, /).
* Поддержка скобок для задания приоритета операций.
* Поддержка математических функций (sin, cos, tg, ln, exp и др.).
* Возможность вычисления выражений с переменными (например, "2\*x + 3").

#### Построение графиков функций:

* Возможность построения графиков функций одной переменной (например, y = x^2).
* Настройка диапазона значений по осям X и Y.
* Отображение сетки на графике.

#### Работа с файлами:

* Сохранение графиков в формате изображения (например, PNG или JPEG).

### Интерфейс пользователя:

* Простой и интуитивно понятный интерфейс.
* Поле для ввода выражения.
* Кнопки для выполнения операций (вычисление, построение графика).
* Область для отображения графика.

### Нефункциональные требования

* Производительность:
* Приложение должно быстро обрабатывать выражения и строить графики.
* Удобство использования:
* Интерфейс должен быть простым и понятным для пользователя.
* Кроссплатформенность (по возможности):
* Приложение должно работать на Windows (основная платформа для Delphi).
* Технические требования
* Язык программирования: Delphi.
* Среда разработки: Embarcadero RAD.

### Библиотеки:

* Для математических вычислений: стандартные функции Delphi.
* Для построения графиков: компонент TChart или аналогичный.
* Для работы с файлами: стандартные функции Delphi для работы с текстовыми файлами и изображениями.

# Анализ требований к ПС и разработка функциональных требований

## Теоретический анализ и математическое обоснование

### **Обработка математических выражений:**

**Парсинг выражений:** Введенное пользователем выражение (например, 2 \* (3 + sin(π/2))) должно быть преобразовано в структуру данных, удобную для вычислений. Для этого используется алгоритм рекурсивного спуска или библиотеки для парсинга (например, MathParser в Delphi).

#### **Поддержка функций и операторов**:

* Базовые операции: сложение (+), вычитание (-), умножение (\*), деление (/), возведение в степень (^).
* Тригонометрические функции: sin(x), cos(x), tan(x).
* Логарифмические функции: ln(x), log10(x).
* Другие функции: sqrt(x), abs(x), exp(x).

.

#### Работа с переменными:

Пользователь должен иметь возможность задавать переменные (например, x, y и т. д.) и использовать их в выражениях.

### Построение графиков функций:

#### **Дискретизация функции**:

Для отображения графика функция вычисляется в N точках на заданном интервале [Xmin, Xmax]. Шаг дискретизации определяется как:

где N — количество точек (рекомендуемое значение: 1000 для плавного графика).

#### Масштабирование и панорамирование:

Преобразование координат графика в пиксели экрана выполняется по формулам:

### Обработка ошибок:

* ПС должно корректно обрабатывать ошибки ввода (например, деление на ноль, некорректные символы).
* Для каждой ошибки должно выводиться понятное сообщение (например, «Ошибка: деление на ноль»).

## Описание функциональности ПС

**Описание ключевых вариантов использования:**

Акторы:

* **Пользователь** – человек, взаимодействующий с программным средством.

**Варианты использования:**

### Ввод и вычисление выражения:

Пользователь вводит выражение в текстовое поле.

ПС проверяет синтаксис, вычисляет результат и выводит его.

Пример: Ввод 2\*(3+sin(π/2)) → Результат: 8.0.

### Построение графика функции:

Пользователь задает функцию (например, y = x^2), диапазоны по осям X и Y.

ПС генерирует график, отображает оси, сетку и легенду.

Настройка внешнего вида графика:

Пользователь выбирает цвет линии, тип графика (сплошная, пунктирная), толщину линии.

ПС обновляет отображение в реальном времени.

### Экспорт графика:

Пользователь сохраняет график как изображение (PNG/JPEG) или копирует в буфер обмена.

### Обработка ошибок:

При вводе некорректного выражения (например, 2+\*3) ПС выводит сообщение: «Ошибка синтаксиса: неверный оператор».



Диаграмма вариантов использования

## Спецификация функциональных требований

### Обработка математических выражений

#### Пользователь вводит выражение в текстовое поле интерфейса.

#### ПС поддерживает:

* Базовые операции: +, -, \*, /, ^ (степень).
* Функции: sin(x), cos(x), tan(x), ln(x), log10(x), sqrt(x).
* Константы: π (3.1415), e (2.7182).
* Скобки для задания приоритета.

Результат вычисления отображается с точностью до 4 знаков после запятой.  
При обнаружении ошибки (деление на ноль, неверный синтаксис) выводится сообщение с указанием типа ошибки.

### Построение графиков функций

#### Пользователь задает функцию в формате y = f(x) (например, y = 2\*sin(x) + x^2).

#### Настройка диапазонов:

* По умолчанию масштабируется в зависимости от координат границ функции.
* Пользователь может задать произвольно

# Анализ требований к ПС и разработка функциональных требований



Схема 1:Основная схема

1. Начало: Программа запускается и переходит к выбору действия.
2. Запуск по: Пользователю предлагается выбрать одно из трех действий:
   1. Выражение: Обработка математического выражения.
   2. Функция: Построение графика функции.
   3. Выход: Завершение работы программы.
3. Выбор действия: В зависимости от выбора пользователя программа выполняет соответствующее действие:
   1. Обработка выражений: Программа переходит к модулю обработки математических выражений.
   2. Построение графика: Программа переходит к модулю построения графиков функций.
   3. Конец: Программа завершает свою работу.



Схема 2:Обработка выражения

1. Начало обработки выражений: Алгоритм начинает выполнение.
2. Ввод выражения пользователем: Пользователь вводит математическое выражение в текстовое поле.
3. Лексический анализ: Выражение разбивается на токены (числа, операторы, функции, скобки). Это позволяет выделить отдельные элементы выражения для дальнейшей обработки.
4. Синтаксический анализ: На основе токенов строится абстрактное синтаксическое дерево (AST). Это дерево представляет структуру выражения и позволяет корректно интерпретировать порядок операций.
5. Проверка корректности выражения: Проверяется правильность расстановки скобок и операторов. Если обнаружены ошибки, программа выводит соответствующее сообщение.
6. Рекурсивный обход дерева для вычисления результата: Программа обходит AST, вычисляя значение выражения. Этот процесс включает выполнение операций в правильном порядке (с учетом приоритетов операторов и скобок).
7. Вывод результата: Результат вычисления отображается пользователю.
8. Конец обработки выражений: Алгоритм завершает свою работу.



Схема 3:Построение графика

1. Начало построения графика:
   * Начало алгоритма.
2. Ввод функции:
   * Пользователь вводит функцию (например, y = sin(x)).
3. Установка диапазона:
   * Пользователь задает диапазоны по осям X и Y (например, X ∈ [-10, 10], Y ∈ [-1, 1]).
4. Дискретизация:
   * Функция вычисляется в N точках на интервале [Xmin, Xmax]. Шаг дискретизации:
5. Отрисовка графика:
   * Точки графика преобразуются в пиксели и отображаются на экране.
6. Настройка визуализации:
   * Пользователь может выбрать цвет линии, тип графика (сплошная, пунктирная) и толщину линии.
7. Экспорт графика:
   * Пользователь может сохранить график в формате PNG/JPEG или скопировать его в буфер обмена.
8. Конец построения графика:
   * Завершение работы алгоритма.



Схема 4: Данные

* Модуль ввода:
  1. Получает данные от пользователя (математическое выражение или функцию для построения графика).
  2. Передает данные в модуль обработки выражений или модуль графики.
* Модуль обработки выражений:
  1. Выполняет лексический и синтаксический анализ выражения.
  2. Строит абстрактное синтаксическое дерево (AST).
  3. Передает AST в модуль вычислений.
* Модуль вычислений:
  1. Рекурсивно обходит AST и вычисляет результат выражения.
  2. Передает результат в модуль вывода.
* Модуль графики:
  1. Получает функцию и диапазоны от модуля ввода.
  2. Выполняет дискретизацию и отрисовку графика.
  3. Передает график в модуль вывода.
* Модуль вывода:
  1. Отображает результат вычисления или график пользователю.