|  |  |
| --- | --- |
|  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Институт искусственного интеллекта | | |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры | | |

|  |  |
| --- | --- |
| **КУРСОВАЯ РАБОТА**  **по теме**  **«**Реализация программного обеспечения для вычисления производных с использованием функциональной парадигмы программирования**»** | |
| **по дисциплине** | |
| **«**Функциональное программирование**»** | |
|  | |
| Студент 1-го курса магистратуры  Группы | Минеев С. А. |
| Преподаватель | Милонов Г. А. |
|  |  |
|  |  |

Москва 2024

Оглавление

[1 **Общие сведения** 2](#_Toc168938170)

[1.1 Сведения об функциональных ограничениях на применение 3](#_Toc168938171)

[1.2 Программное и аппаратное обеспечение, необходимое для функционирования программы 5](#_Toc168938172)

[1.3 Используемые языки программирования 5](#_Toc168938173)

[1.4 Краткое описание средств языка в области поддержки функциональной парадигмы 6](#_Toc168938174)

[**2** **Структура программного обеспечения** 7](#_Toc168938175)

[2.1 Алгоритм работы программы 7](#_Toc168938176)

[2.2 Используемые методы 11](#_Toc168938177)

[2.3 Структура программы 13](#_Toc168938178)

[2.4 Связи программы с другими программами 13](#_Toc168938179)

[**3** **Структуры данных** 14](#_Toc168938180)

[3.1 Характер, организация и предварительная подготовка входных данных 14](#_Toc168938181)

[3.2 Формат, описание и способ кодирования входных данных 15](#_Toc168938182)

[3.3 Характер и организация выходных данных 18](#_Toc168938183)

[3.4 Формат, описание и способ кодирования выходных данных 18](#_Toc168938184)

[**4** **Методика и результаты тестирования** 0](#_Toc168938185)

[4.1 Состав и структуру технических средств, необходимых для проведения тестирования 0](#_Toc168938186)

[4.2 Состав и структуру программного обеспечения, необходимого для проведения тестирования 0](#_Toc168938187)

[4.3 Программа (последовательность) тестирования, указывающую те свойства программного обеспечения, которые будут проверяться и очередность, в которой эти проверки будут производиться 1](#_Toc168938188)

[4.4 Результаты тестирования 0](#_Toc168938189)

# 1 **Общие сведения**

**Аннотация**

Данная работа посвящена реализации программы для вычисления производных от одного аргумента с использованием функциональной парадигмы программирования.

**Наименование программы**:

– Система вычисления производных.

**Назначение программы:**

Вычисление производных вещественных функций от одного вещественного аргумента.

## 1.1 Сведения об функциональных ограничениях на применение

Данное программное обеспечение разработано на основе следующих функциональных требованиях:

Таблица № 1 – Функциональные требования и ограничения

|  |  |  |
| --- | --- | --- |
| **№** | **Описание требования** | **Примечание** |
| 1 | **Требование к backend** | |
|  | Наличие универсального метода `*deriv*`, позволяющего на вход принять математическое выражение в виде инфиксной последовательности символов | Реализуется на стороне guile (т. е. в файле guile.scm) |
|  | Наличие макрос-правил, позволяющих правильно и однозначно оценить и сопоставить с определенной лексемой в выражении результат производной |
|  | Учет возможного присутствия в выражениях следующих математических операций:  Сложение (), Вычитание (), Умножение (, Деление (), Возведение в степень () |
|  | Выражения могут содержать следующие элементарные функции от одного аргумента:  Линейные функции произвольной степени тригонометрические (sin, cos, tg, ctg), обратные тригонометрические (asin, acos, atg, actg), гиперболические (sh, csh, th, cth), логарифмические ( - Десятичный логарифм, – Натуральный логарифм, – логарифм по основанию ), показательные (), экспоненциальные , рациональные() |
|  | Наличие правил вычисления производных по следующим основным законам вычисления производных:   1. (Производная от константы равна нулю). 2. (Производная от переменной равна единицы). 3. (Правило вычисления производной от рациональной функции) 4. (Правило вычисления производной от сложной функции.) |
|  | Наличие функциональности для организации и взаимодействия с guile |  |
|  | Наличие реализованных переменных аргументов функций в guile: . | Реализуется на стороне guile (т. е. в файле guile.scm) |
| 2 | **Требование к frontend (GUI)** | |
|  | Наличие кнопки открытия/закрытия соединения с guile на порту 37146 host: localhost (http:127.0.0.1) утилиты telnet |  |
|  | Наличие консольного окна в GUI-программе для взаимодействия c guile |
|  | Наличие метода `*print`* в guile для отправки результатов в формате const char\* в программу | Реализуется на стороне guile (т. е. в файле guile.scm) |
|  | Наличие дисплея отображения результатов вычисления производной в математической форме, подобной latex. |  |
|  | Наличие установки настроек размера и стиля шрифта отображения результатов, полученных с guile. | По умолчанию ***размер*** шрифта должен быть равен ***14 кегель***, ***стиль*** шрифта ***Contarell***. |
|  | Наличие графического компонента для занесения полученных результатов в буфер обмена. |  |
|  | Наличие привычного оформления программы. (Расположения кнопок развертывания окна программы/ сворачивания и закрытия должно быть в правом верхнем углу с соответствующим пиктограммы оформлением) |
|  | Наличие функциональности настройки и выбора стиля оформления программы |

## 1.2 Программное и аппаратное обеспечение, необходимое для функционирования программы

Для функционирования программы помимо самой программы требуется наличие установленного и работающего guile (рекомендуется guile 3.0.9).

## 1.3 Используемые языки программирования

В качестве языков разработки использовались следующие языки программирования:

* C++ (Qt Creator and Qt 6.6.0/6.7.0) – Графическое оформление программы.
* Guile (Scheme) 3.0.9 – Реализация функциональной части вычисления производных.

## 1.4 Краткое описание средств языка в области поддержки функциональной парадигмы

В данном разделе (подразделе) описывается кратко средства функционального программирования Guile Scheme.

Scheme – функциональный язык программирования, один из трёх наиболее популярных диалектов Лиспа (наряду с Common Lisp и Clojure).

Создан в середине 1970-х годов исследователями Массачусетского технологического института Гаем Стилом и Джеральдом Сассменом.

Обладает минималистичным дизайном. Содержит минимум примитивных конструкций и позволяет выразить всё необходимое путём надстройки над ними. Например, использует всего два механизма организации циклов — хвостовую рекурсию и итеративный подход.

Реализация guile scheme строится на основе макроса правил по следующей сигнатуре:

(define-syntax name-macros

( (sin ln)

((\_ sin e) ()) ;

((\_ ln e\* …) ()) ;

)

)

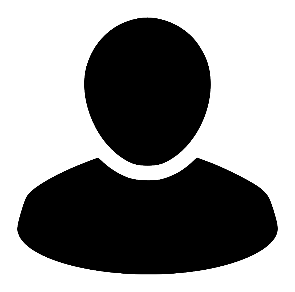
Рис. 1 – Демонстрация макрос правил.

Макрос-правила[macros-rules] состоят из паттерна (pattern) и шаблона (template) сопоставления. В примере на рис. 1 в качестве паттерна выступает последовательность, которая начинается с символа \_, означающего безымянную переменную. В данном случае под \_ понимается последовательность символов, стоящая перед паттерном. Также в макрос-правиле также можно заметить e\* … . Под e\* понимается буквенный или числовой символ, а под знаком ‘Эллипсис’ (…) понимается любая следующая за ним последовательность символов/операторов. Стоит отметить, что e\* не может содержать операторы, в том числе круглые скобки и оператор экранирования.

# **Структура программного обеспечения**

## Алгоритм работы программы

Представление:

Guile system

Пользователь QEvalDerivSys

Рис. 2 – Алгоритм работы программы

Описание:

Пользователь вводит в консоль Guile последовательность формата, описанного в п. 3.2. Затем нажав на клавиатуре клавишу Enter пользователь запускает анализ введенной строки и отправку ее guile по telnet соединению на указанном хосте (по умолчанию: localhost [127.0.0.1]) и порту (по умолчанию 37146). Запрос обрабатывается на стороне guile согласно предопределенным макрос правилам вычисления производной, которые загружаются предварительно при запуске программы. По итогу работы guile формируется результат строкового типа и отправляется в программу. Программа методом “print” получает результат и выводит в соответствующее представление.

Пример работы программы:



Рис. 3 – Демонстрация окна приветствия со светлой темой

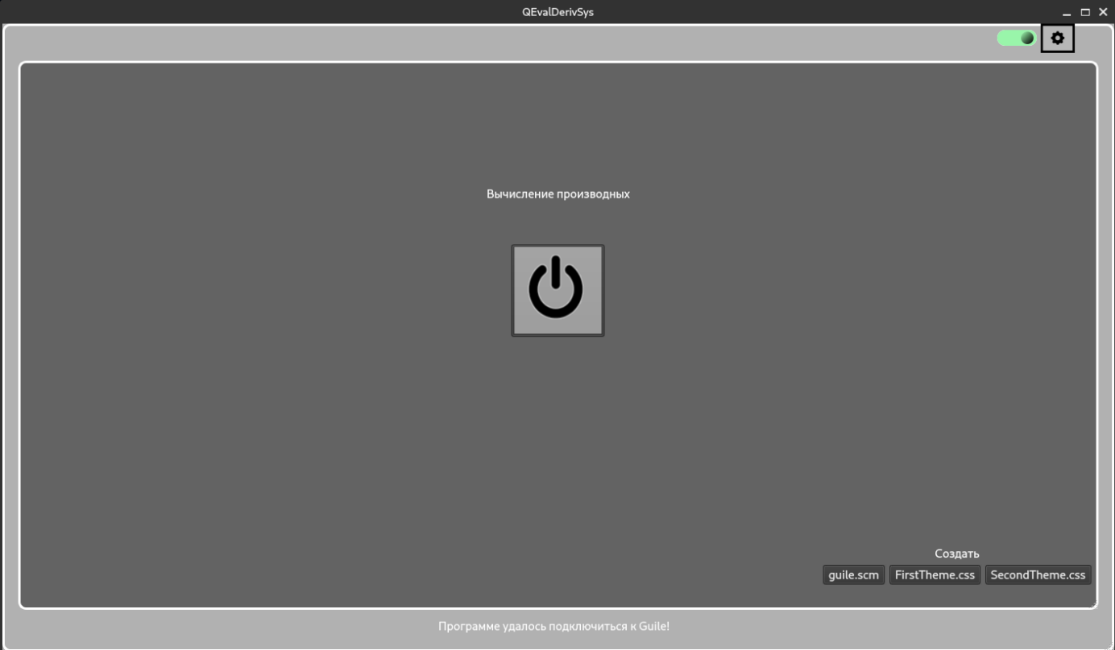


Рис. 4 – Демонстрация окна приветствия с темной темой

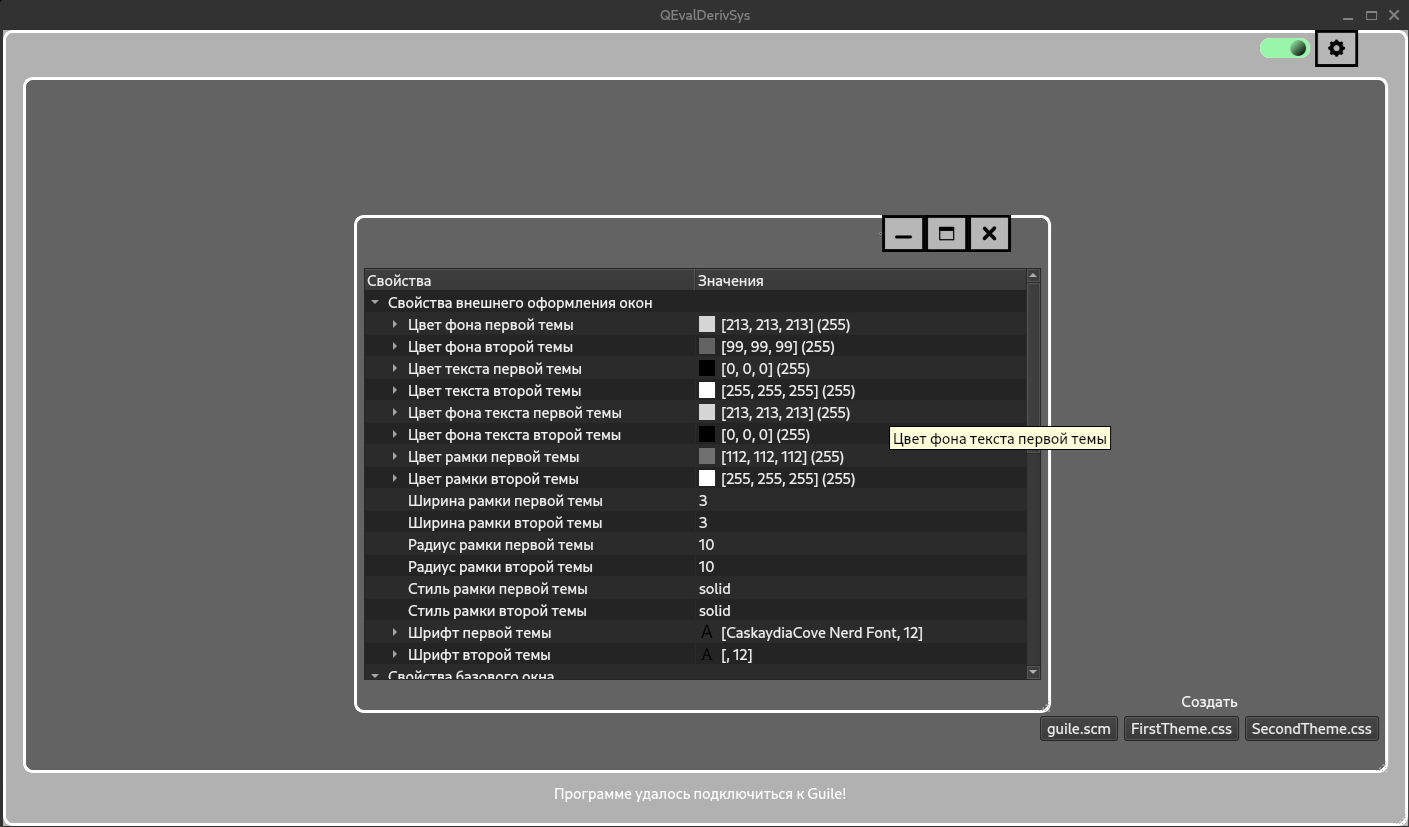


Рис. 5 – Демонстрация окна задания настроек программы

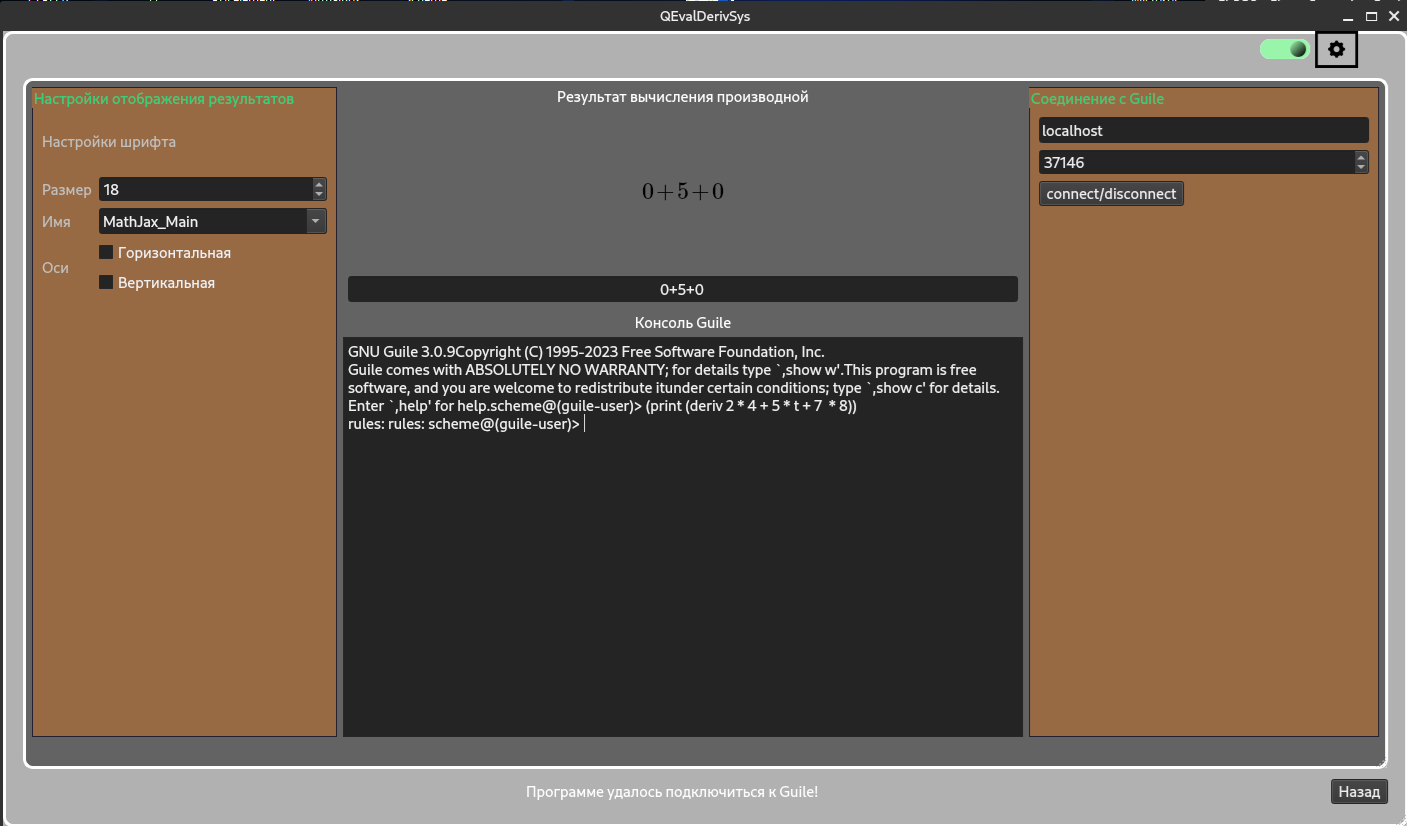


Рис. 6 – Демонстрация базового окна работы с guile при темной теме

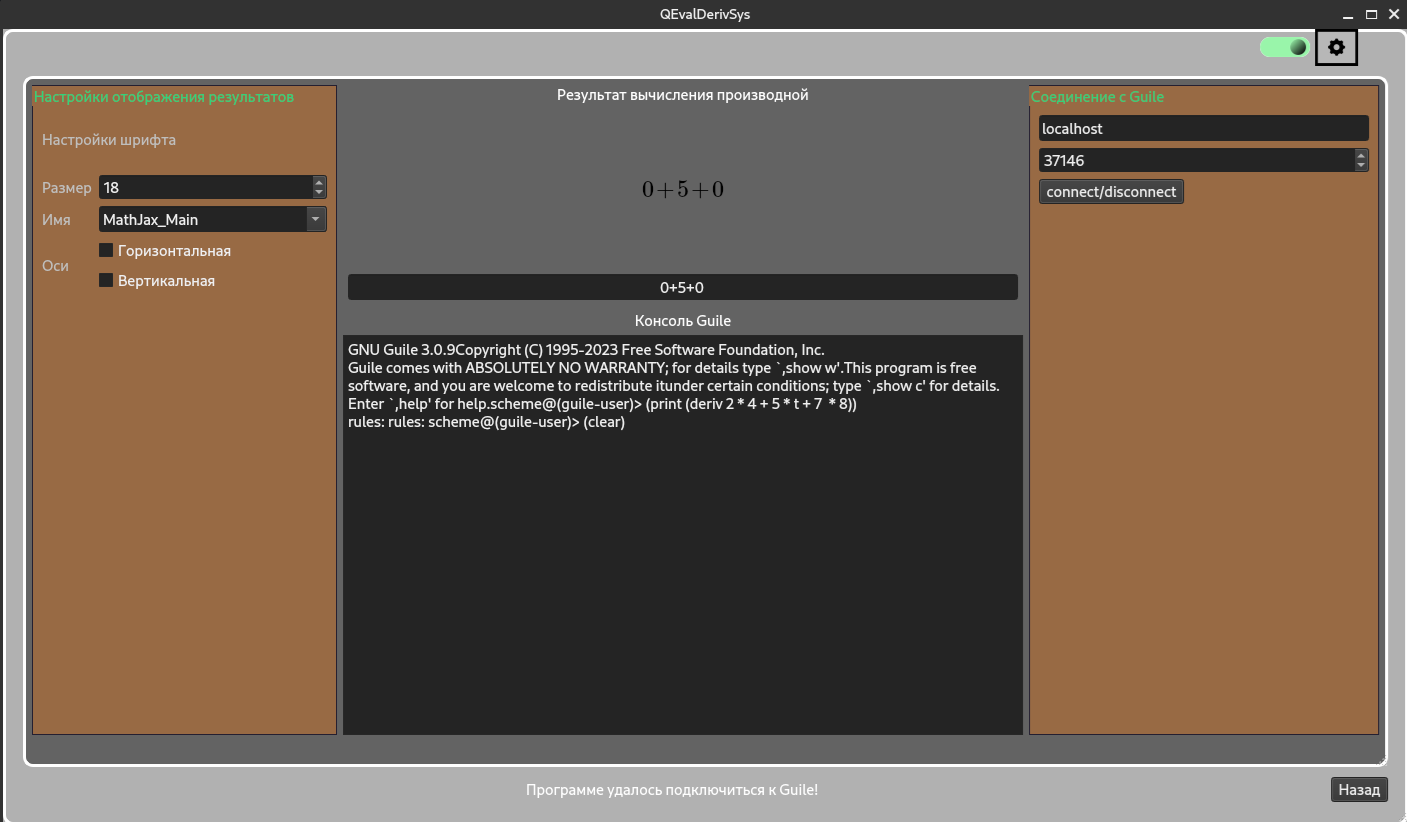


Рис. 7 – Демонстрация ввода формул и вывода результата. Очистка консоли

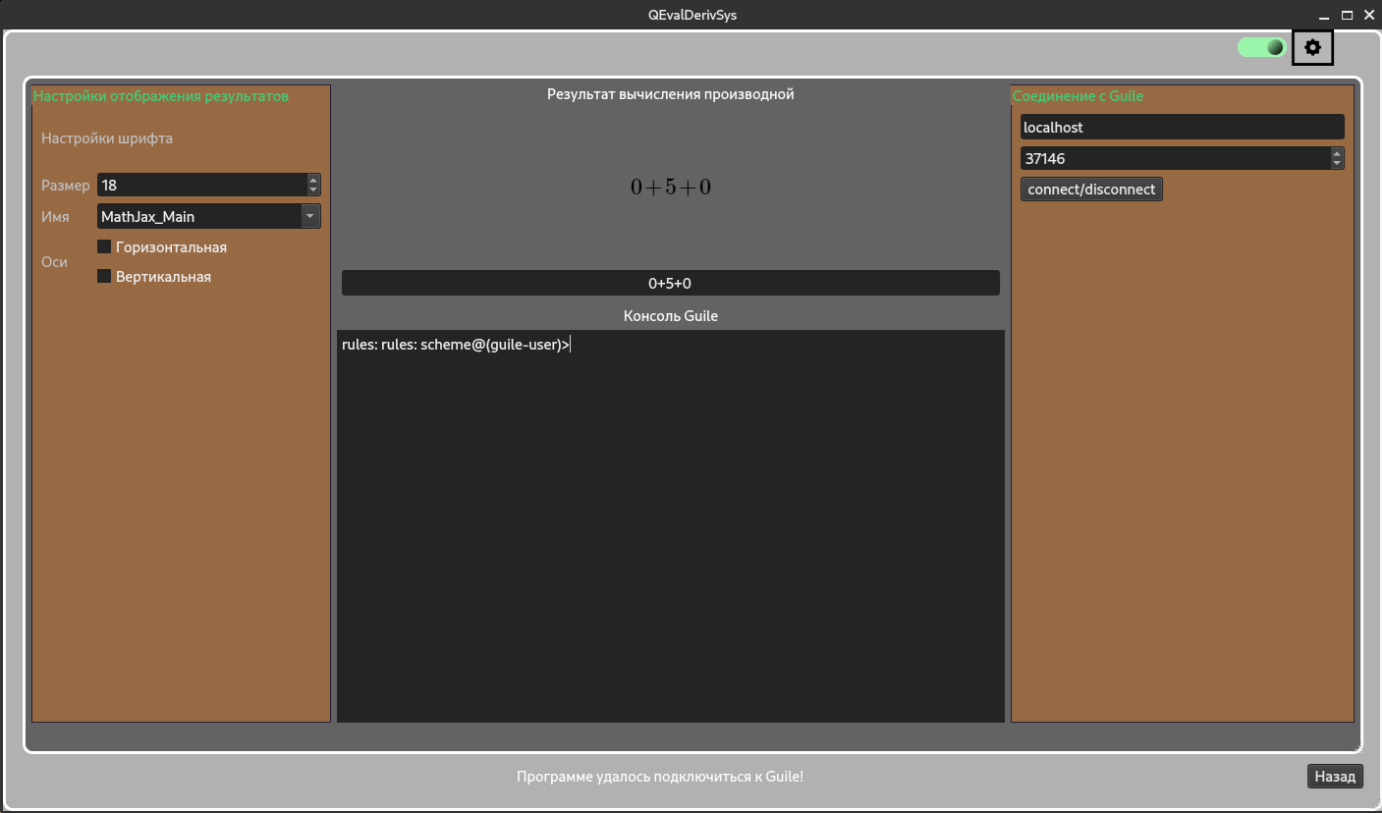


Рис 8 – Демонстрация очищенного окна консоли



Рис. 9 – Демонстрация базового окна программы cо светлой темой

## Используемые методы

Далее в таблице № 2 приводятся сведения об основных методах, используемых в программе.

Таблица № 2 – Основные используемые методы

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Компонет** | **Метод** | **Описание** |
| 1 | QMainWindow |  |  |
| 1.1 |  | bool connectGuile() | Подключение к Guile |
| 1.2 |  | existGuileFile | Проверка на существования файла guile.scm в директории приложения |
| 1.3 |  | loadCodeSchemeInGuile | Загрузка када guile/ |
| 1.4 |  | createFileGuile | Создание файла guile |
| 2 | QBaseWindow |  |  |
| 2.1 |  | static void\* register\_functions(void \*); | Регистрация метода print класса QBaseWindow в guile |
| 2.2 |  | static SCM print(SCM msg); | Метод получения результатов. |
| 2.3 |  | void setFontSize(int size); | Изменение размера шрифта |
| 2.4 |  | void setFontStyle(QString style\_name); | Изменение стиля шрифта |
| 2.5 |  | void addTextTelnet(const char \*msg, int count); | Передача данных в guile |
| 3 | WSettings |  |  |
| 3.1 |  | updateCurrentTheme() | Обновление текущей темы оформления |
| 3.2 |  | expanded() | Расширение окна программы |
| 3.3 |  | void collapse() | Сворачивания окна программы |
| 4 | QTelnet |  |  |
| 4.1 |  | sendData() | Передача данных в guile по порту |
| 4.2 |  | connectToHost(const QString &host, quint16 port) | Подключение к guile |

## Структура программы

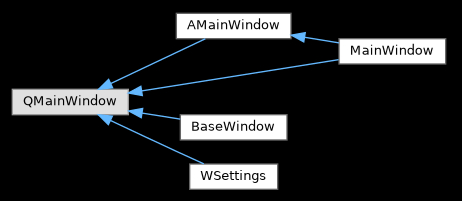


Рис. 10 – Структура программы

Описание:

Программа имеет следующую структуру:

В основе находится **MainWindow**, который наследует внешнее оформление от абстрактного класса **АMainWindow** и вкладывает в него свой пользовательский интерфейс. При нажатии на кнопку работы с guile окно **MainWindow** делается скрытным и подменяется на **BaseWindow**.

**BaseWindow** представляет собой окно, которое вложено в **AMainWindow** и содержит функционал по работе с **Guile**.

Помимо двух этих окон в программе предусмотрено ещё одно окно для настройки стилей оформления приложения - **WSettings**.

## Связи программы с другими программами

Guile

Рис. 11 – Структура связи c Guile

Описание:

Программа **QEvalDerivSys** осуществляет соединение с guile по средствам **QTelne**t соединения на порту 37146 localhost.

# **3 Структуры данных**

## Характер, организация и предварительная подготовка входных данных

При работе с программой в *лучшем случае* никакой предварительной подготовки данных не требуется. В случае же отсутствия необходимых файлов для работы программы имеется реализация для их генерации.

Отметим, что сгенерированный файл guile.scm по умолчанию располагается рядом с программой, а файлы стилей ThemeFirst.css и ThemeSecond.css в директорию qss, которая также располагается в месте установки программы.

В качестве входных данных консоли guile выступает последовательность символов строкового типа (Числа автоматически интерпретируются как строковый тип), которая оформлена согласно формату, описанному далее в п. 3.2.

## Формат, описание и способ кодирования входных данных

Окно консоли на вход принимает строку следующего формата:

(print ( ()))

Функция отвечает за перенаправление результатов вычисления в программы, для последующего отображения в ней полученных результатов в формате подобном .

Функция принимает на вход математическое выражение в виде последовательности имен функция, скобок приоритетности и переменных и позволяет вычислить значение производной от такого типа выражений. В качестве неизвестных переменных определены и .количество открывающихся скобочек должно быть равно количеству закрывающихся.

Стоит отметить, что не требуется первую функцию обрамлять скобками иначе скобки будут интерпретироваться как попытка представить поток входных аргументов единым списком, что не позволить вычислить производную:

* ***Неверное*** задание выражения:

(print ( ( ())))

* ***Верное*** задание выражения:

(print ( ()))

Для очистки окна программы используйте метод (clear), который обрабатывается на стороне QEvalDeriv.

Имена и обозначения используемых и поддерживаемых функций, а также их сигнатура приводится в следующей таблице:

Таблица № 3 – Реализованы правила вычисления производных функций

|  |  |  |
| --- | --- | --- |
| **№** | Функции | **Параметры** |
| - | Тригонометрические функции |  |
| 1 | sin(t) |  |
| 2 | cos(t) |  |
| 3 | tg(t) |  |
| 4 | ctg(t) |  |
| - | Обратные тригонометрические функции |  |
| 5 | asin(t) |  |
| 6 | acos(t) |  |
| 7 | atg(t) |  |
| 8 | actg(t) |  |
| - | Гиперболические функции |  |
| 9 | sh(t) |  |
| 10 | csh(t) |  |
| 11 | th(t) |  |
| 12 | cth(t) |  |
| - | Другие функции |  |
| 13 | log n (t) | n – степень логарифма |
| 14 | lg(t) | Десятичный логарифм |
| 15 | ln(t) | Натуральный логарифм |
| 16 | sqrt(t) |  |
| 17 | sqrt\_n n (t) | Корень n-ой степени |
| 18 | pow k (t) | Степенная функция |
| 19 | t ^ k | Степенная функция |
| 20 | k \* t []\*\* | \*\* - означает последовательность от нуля и более, означает вычисление производной от любой другой или той же самой функции |
| 21 | k ^ t []\*\* | \*\* - означает последовательность от нуля и более, означает вычисление производной от любой другой или той же самой функции |
| 22 | t ^ k []\*\* | \*\* - означает последовательность от нуля и более, означает вычисление производной от любой другой или той же самой функции |

Реализованные правила дифференцирования:

Таблица № 4 – Правила дифференцирования

|  |  |  |
| --- | --- | --- |
| **№** | **Правило дифференцирования** | **Описание производной** |
| 1 |  | Производная от константы |
| 2 |  | Производная от переменной |
|  |  | Производная показательной функции |
| 2 |  | Производная степенной функции |
| 3 |  | Производная от рациональной функции |
| 4 |  | Производная произведения |
| 5 |  | Производная сложной функции |

!!! В качестве переменной вы можете использовать t или x. Рекомендуется использовать .

!!! Не забывайте об использовании парных скобок.

## Характер и организация выходных данных

Передача и прием данных между guile и ПО осуществляется с использованием методов, которые обрабатывают строковый тип данных.

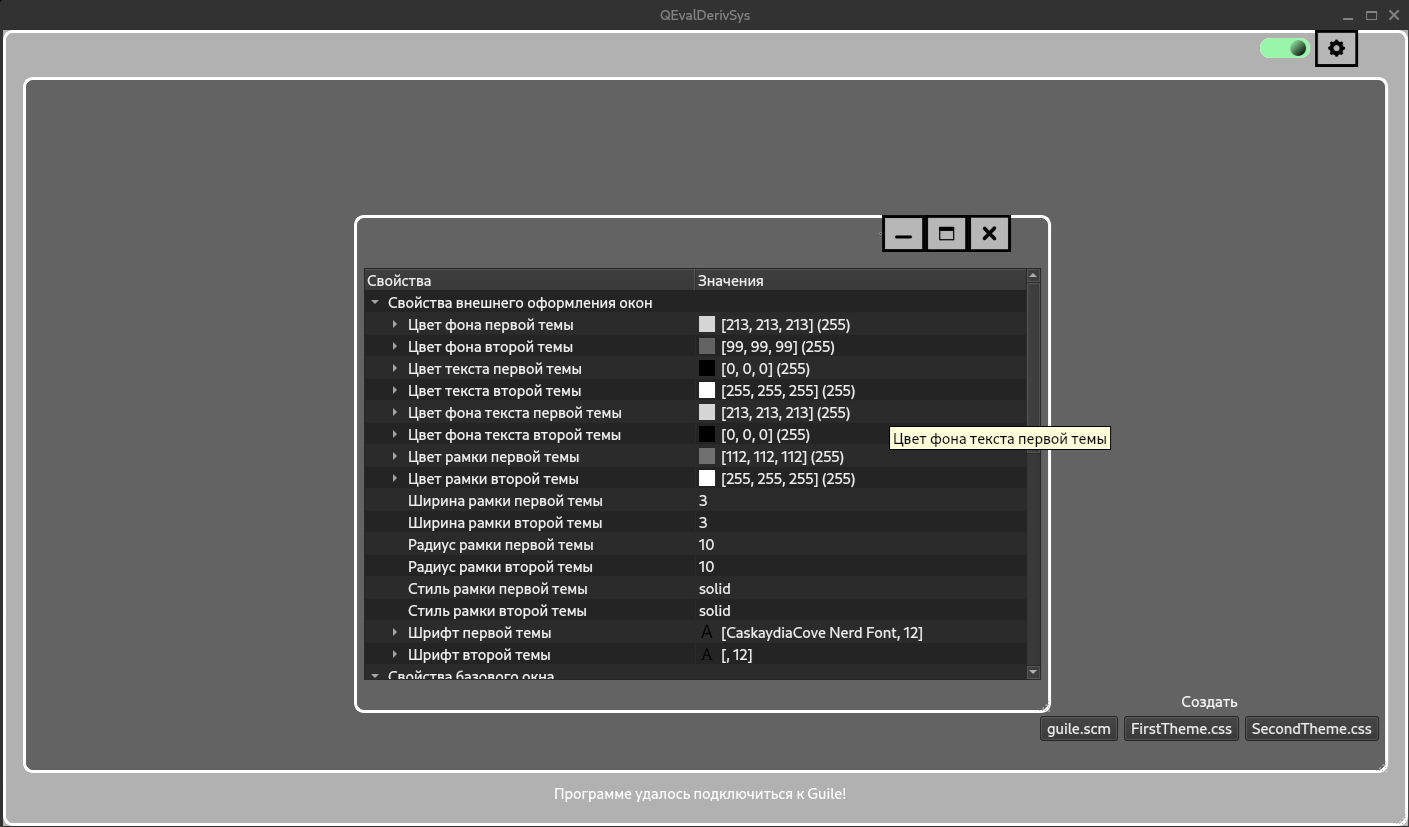
Программа на выходе выдаёт последовательность строкового типа. Guile ее принимает и обрабатывает, возвращая для всей последовательности целостный результат вычисления строкового типа.

## Формат, описание и способ кодирования выходных данных

Guile возвращает результат строкового типа в метод print класса BaseWindow, который распечатывает полученные данные в QExprWidget.

QExprWidget осуществляет синтаксический, лексический и семантический анализ математического выражения и выводит результат в latex-подобном стиле оформления.

Пользователь имеет возможность настроить размер и стиль шрифта в соответствующих настройках:

Рис. 12 - Демонстрация настройки параметров.

# **4 Методика и результаты тестирования**

## Состав и структуру технических средств, необходимых для проведения тестирования

Для проведения тестирования необходимо наличие установленного Qt Creator 6 с комплектом сборки Qt 6.6.0 или Qt 6.7.0. С другими версиями Qt гарантии успешной сборки отсутствует.

Также должен быть установлен Guile. Рекомендуемая версия 3.0.9.

В зависимости от платформы может понадобиться библиотека libxkbcommon-dev, libxkbcommon-tools или другие дополнительные зависимости. В последнем случае рекомендуется обратиться к соответствующей документации.

## Состав и структуру программного обеспечения, необходимого для проведения тестирования

Для проведения тестирования в ***состав*** разработанного ПО входит модуль ***QEvalDerivTest***, предназначенный для проведения тестирования.

Структура данного модуля включает в себя 8 основных тест-кейсов (см. п. 4.3 Последовательность тестирования).

## Программа (последовательность) тестирования, указывающую те свойства программного обеспечения, которые будут проверяться и очередность, в которой эти проверки будут производиться

Порядок проведения тестирования описывается следующей последовательностью тест-кейсов:

Таблица № 5 – Тесты

|  |  |  |
| --- | --- | --- |
| **№** | **Описание теста** | **Примечание** |
| 1 | Тест на успешное открытие/закрытие соединения с утилитой guile по telnet | Должен быть реализован метод с именем и в случае успешного открытия возвращаться истиное (true) логическое значение, в противном случае ложное значение (false). |
| 2 | Тест на отсутствие/присутствие guile.scm рядом с исполняемым файлом программы | В случае отсутствия должен создаться по умолчанию и снова осуществиться проверка на существования. |
| 3 | Тест на отсутствие/присутствие директории qss рядом с исполняемым файлом программы. и/или файлов стилей оформления программы в директории qss | - |
| 4 | Тест на настройку размера шрифта | - |
| 5 | Тест на настройку стиля шрифта | - |
| 6 | Тесты на вычисление производных | Количество тестов определяется с учетом наличия функциональной парадигмы программирования с рекурсивными макрос-правилами покрытия всех реализованных методов. |

## Результаты тестирования

Дата проведения тестирования: 06.06.24

Время проведения тестирования: 14:00

Тестирование провел: Минеев С. А.



Рис. 12 – Результаты тестирования