МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой «ЭВМ»
_____ Д. В. Топольский
«___»_____ 2022 г.

КРОССПЛАТФОРМЕННЫЙ КАЛЬКУЛЯТОР ФИЗИЧЕСКИХ ВЕЛИЧИН

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2022.256 ПЗ ВКР

Руководитель работы,
к. т. н., доцент кафедры «ЭВМ»
_____ Е. С. Ярош
«___»_____ 2022 г.

Автор работы,
студент группы КЭ-406
_____ В. Ф. Ушаков
«___»_____ 2022 г.

Нормоконтролер,
к. п. н., доцент кафедры «ЭВМ»
_____ М. А. Алтухова
«___»_____ 2022 г.

Челябинск-2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой «ЭВМ»
_____ Д. В. Топольский
«___»_____ 2022 г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра
студенту группы КЭ-406
Ушакову Владиславу Федоровичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Кроссплатформенный калькулятор физических величин» утверждена приказом по университету от ___ декабря 2021 г. № _____.

2. **Срок сдачи студентом законченной работы:** 1 июня 2022 г.

3. **Исходные данные к работе:**

– обеспечить вычисление математических выражений, в которых операндами являются числа с единицами измерения, в том числе комплексные;

– реализовать функцию показа результата во время ввода;

– обеспечить автоматическое приведение в одну систему единиц операндов с разными единицами измерения;

– предусмотреть обнаружение несовместимых физических величин;

– реализовать функцию ввода постоянных;

– реализовать функцию добавления пользовательских единиц измерения и постоянных;

– предусмотреть возможность локализации разработки для англоязычных пользователей;

– обеспечить возможность работы на операционных системах Android версии не ниже 9.0 (уровень API не ниже 28) и Windows не ниже 10.

4. **Перечень подлежащих разработке вопросов:**

– проанализировать существующие калькуляторы, вычисляющие выражения с единицами измерения;

– проанализировать существующую модель величин;

– разработать собственную модель величин;

– разработать собственный калькулятор, вычисляющий выражения с единицами измерения.

5. **Дата выдачи задания:** 1 декабря 2021 г.

Руководитель работы _____/*Е. С. Ярош*/

Студент _____/*В. Ф. Ушаков*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	10.03.2022	
Разработка модели, проектирование (обеспечить автоматическое приведение в одну систему единиц операндов с разными единицами измерения; предусмотреть обнаружение несовместимых физических величин; реализовать функцию добавления пользовательских единиц измерения и постоянных; разработать собственную модель величин)	21.03.2022	
Реализация системы (обеспечить вычисление математических выражений, в которых операндами являются числа с единицами измерения, в том числе комплексные; реализовать функцию показа результата во время ввода; реализовать функцию ввода постоянных)	04.04.2022	
Тестирование, отладка, эксперименты (предусмотреть возможность локализации разработки для англоязычных пользователей; обеспечить возможность работы на ОС Android версии не ниже 9.0 (уровень API не ниже 28) и Windows не ниже 10)	25.04.2022	
Компоновка текста работы и сдача на нормоконтроль	16.05.2022	
Подготовка презентации и доклада	24.05.2022	

Руководитель работы _____/*Е. С. Ярош*/

Студент _____/*В. Ф. Ушаков*/

АННОТАЦИЯ

Ушаков В. Ф. Кроссплатформенный калькулятор физических величин. – Челябинск: ЮУрГУ, ВШ ЭКН; 2022, 53 с., 23 ил., библиогр. список – 14 наим., 2 прил.

В рамках выпускной квалификационной работы производится анализ современных технологий для вычисления физических величин. Организуется разработка программного комплекса для вычисления физических величин с использованием среды разработки кроссплатформенных приложений Qt. Рассматриваются преимущества и недостатки как модели вычислений физических величин в целом, так и конкретных программных комплексов. Доказывается способность предлагаемой архитектуры к обеспечению успешного вычисления физических величин.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире калькуляторы являются незаменимым инструментом в таких областях науки, как физика, химия и электротехника. Однако далеко не все калькуляторы обладают достаточным для этих областей функционалом. В частности, большинство калькуляторов не имеет единиц измерения.

Актуальность данной темы обусловлена тем, что при решении задач необходимо вручную учитывать единицы измерения, что может как занять некоторое время, так и привести к ошибкам.

Целью настоящей работы является разработка приложения, ускоряющего решение задач, в которых используются единицы измерения, а также снижающего количество ошибок в них.

Для достижения поставленной цели необходимо решить следующие *задачи*:

– проанализировать существующие калькуляторы, вычисляющие выражения с единицами измерения;

– проанализировать существующую модель величин;

– разработать собственную модель величин;

– разработать собственный калькулятор, вычисляющий выражения с единицами измерения.

Работа состоит из пяти глав. В первой главе описаны существующие на данный момент аналоги и обоснован выбор средств реализации. Во второй главе определены функциональные и нефункциональные требования разрабатываемого программного продукта. В третьей главе описана архитектура предлагаемого решения, приведены алгоритмы решения выполняемых задач, описание базы данных и описание окон графического интерфейса. В четвертой главе представлена реализация классов, графического интерфейса и файловой структуры приложения. В пятой главе произведено тестирование поведения программы при различных действиях пользователя.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

Калькулятор Google – это «удобное приложение для математических вычислений» [1]. Оно является встроенным в большинство устройств под управлением операционной системы Android. Это приложение обладает удобным интерфейсом, однако его функционал ограничен – отсутствуют комплексные числа и единицы измерения.

Достоинства:

– показ результата во время ввода.

Недостатки:

– нет кроссплатформенности;

– нет комплексных чисел;

– нет единиц измерения.

Photomath – «это приложение для изучения математики <…>. Оно предоставляет пошаговые объяснения для задач по алгебре и математическому анализу <…>» [2]. Это приложение доступно для мобильных устройств под управлением Android и iOS. Оно популярно среди физиков, так как обладает широким функционалом, в том числе работой с комплексными числами, однако не поддерживает работу с единицами измерения.

Достоинства:

– показ результата во время ввода (с относительно большой задержкой);

– кроссплатформенность (только Android и iOS);

– комплексные числа.

Недостатки:

– нет единиц измерения.

SMath Studio – это «компактная, но мощная математическая программа с графическим редактором и полной поддержкой единиц измерения» [3]. В ней

имеется большая часть физических постоянных и выводятся ошибки о несовместимости единиц измерения. Интерфейс этой программы на мобильных устройствах достаточно запутан.

Достоинства:

– кроссплатформенность;

– комплексные числа;

– единицы измерения;

– физические постоянные (символ которых не совпадает с единицами измерения);

– вывод ошибок о несовместимости единиц измерения;

– добавление пользовательских постоянных (только в пределах одного файла).

Недостатки:

– запутанный интерфейс на мобильных устройствах.

Mathcad – «это инженерное математическое программное обеспечение, которое позволяет выполнять, анализировать и обмениваться важными расчетами» [4]. Оно обладает всем необходимым функционалом для работы с единицами измерения, однако доступно только для операционной системы Windows.

Достоинства:

– комплексные числа;

– единицы измерения;

– физические постоянные;

– вывод ошибок о несовместимости единиц измерения;

– добавление пользовательских постоянных (только в пределах одного файла).

Недостатки:

– отсутствие кроссплатформенности.

Wolfram|Alpha – «это уникальный движок для решения задач и предоставления знаний» [5]. Он напоминает поисковый движок (при вводе выражения показывается результат и похожие задачи), однако таковым не является. Имеет полную поддержку единиц измерения.

Достоинства:

– показ результата во время ввода (зависит от скорости подключения к интернету);

– кроссплатформенность (веб-страница);

– комплексные числа;

– единицы измерения;

– физические постоянные;

– вывод ошибок о несовместимости единиц измерения.

Недостатки:

– нет возможности добавления собственных единиц измерения.

Лучший конвертер единиц измерения «является самым популярным и функциональным приложением среди всех конвертеров величин в Google Play» [6]. Несмотря на то, что он не является калькулятором, он имеет немаловажную функцию добавления собственных единиц измерения.

Достоинства:

– показ результата во время ввода;

– единицы измерения;

– возможность добавления пользовательских единиц измерения.

Недостатки:

– нельзя вычислить выражение;

– отсутствие кроссплатформенности.

Подведем итоги обзора аналогов, отобразив их функции в таблице 1.

Таблица 1 – Функции аналогов

Функция	**Калькулятор Google**	**Photomath**	**SMath Studio**	**Mathcad**	**Wolfram|Alpha**	**Лучший конвертер единиц измерения**
Показ результата во время ввода	+	±	–	–	±	+
Кроссплатформенность	–	±	+	–	+	–
Комплексные числа	–	+	+	+	+	–
Единицы измерения	–	–	+	+	+	+
Добавление пользовательских единиц измерения	–	–	–	–	–	+
Физические постоянные	–	–	±	+	+	–
Добавление пользовательских постоянных	–	–	±	±	–	–
Вывод ошибок о несовместимости единиц измерения	–	–	+	+	+	–

Как видно из таблицы 1, ни один из аналогов не обладает всеми функциями. Это связано с особенностями их интерфейса и с их назначением.

1.2 Выбор средств реализации

Работа калькулятора связана со сложными расчетами. Чтобы они производились быстрее, код должен выполняться непосредственно на процессоре устройства, то есть быть нативным. Следовательно, не стоит рассматривать такие языки программирования, как Java и C#, так как их код выполняется на виртуальных машинах [7–8].

С другой стороны, для представления модели физических величин нужен высокий уровень абстракции, а следовательно, объектно-ориентированный подход.

Таким образом, язык программирования должен быть как нативным, так и объектно-ориентированным. Этим критериям удовлетворяет C++.

Отталкиваясь от выбранного языка программирования, а также от требования по кроссплатформенности калькулятора, необходимо выбрать интегрированную среду разработки кроссплатформенных приложений на C++.

Таковой является Qt Creator, которая позволяет разрабатывать приложения для любых платформ, при этом исходный код для всех них одинаков [9].

Ввиду выбора Qt Creator, будет использоваться библиотека Qt, которая обладает классами элементов графического интерфейса, однако их использование является крайне неудобным, так как до компиляции и запуска приложения невозможно увидеть, что получилось.

Средство для создания графических интерфейсов Qt Designer решает эту проблему, позволяя редактировать их по принципу «что видите, то и получите» («what-you-see-is-what-you-get») [10].

Так как в дальнейшем планируется перевод текста графического интерфейса с русского языка на английский, необходимо средство для перевода текста, каковым и является Qt Linguist [10].

Его выбор связан с тем, что оно нацелено на работу с элементами графического интерфейса из библиотеки Qt.

Чтобы связать приложение с базой физических величин, необходимо использовать средство для интеграции баз данных. Библиотека Qt обладает модулем Qt SQL, который содержит драйвер для системы управления базами данных, программный интерфейс приложения для SQL и пользовательский интерфейс [10].

Qt SQL поддерживает следующие системы управления базами данных и интерфейсы [10]:

– IBM DB2 (версии 7.1 и выше);

– MySQL или MariaDB (версии 5.6 и выше);

– использующие Oracle Call Interface Driver (версии 12.1 и выше);

– использующие Open Database Connectivity;

– PostgreSQL (версии 7.3 и выше);

– SQLite (версии 3).

Система управления базами данных должна удовлетворять ряду критериев. Она должна быть:

– встраиваемой;

– компактной;

– высокопроизводительной.

Этим критериям удовлетворяет SQLite, которая к тому же является самой используемой в мире [11].

Таким образом, калькулятор должен обладать всеми функциями, перечисленными в таблице 1, то есть сочетать достоинства как обычных калькуляторов, так и систем компьютерной алгебры по части работы с единицами измерения. Это сделает калькулятор удобным для использования и с мобильных устройств, и с персональных компьютеров. Новизна разработки калькулятора будет заключаться не только в сочетании существующих в аналогах функций, но и в добавлении новой: добавление пользовательских единиц измерения и постоянных. Калькулятор будет написан на C++ в Qt Creator с использованием таких средств, как Qt Designer, Qt Linguist и Qt SQL с драйвером для SQLite.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1 Функциональные требования

Система должна обладать следующими функциями:

– вычисление выражения с единицами измерения или без них;

– мгновенное получение результата вычисления выражения;

– приведение всего выражения к одной системе единиц;

– проверка совместимости физических величин;

– ввод постоянных;

– добавление пользовательских единиц измерения и постоянных;

– вывод сообщения об ошибке (неверный синтаксис, несовместимые физические величины).

2.2 Нефункциональные требования

Система должна соответствовать следующим требованиям:

– размерности физических величин в калькуляторе должны соответствовать международной системе единиц;

– кроссплатформенность (Android и Windows).

3 РАЗРАБОТКА МОДЕЛИ, ПРОЕКТИРОВАНИЕ

3.1 Архитектура предлагаемого решения

Архитектура предлагаемого решения будет состоять из трех уровней:

– алгоритмы вычисления выражения;

– база данных;

– графический интерфейс пользователя.

3.2 Алгоритмы решения задачи

Калькулятор должен решать следующую задачу: вычисление выражения. Алгоритмы решения задачи, то есть алгоритмы вычисления выражения, будут основаны на обратной польской записи.

Обратная польская запись (или постфиксная запись) – это форма записи выражений, в которой операнды расположены перед знаками операций. Такая запись удобна для вычисления выражения компьютером, так как операнды и операции добавляются в стек и извлекаются из него по мере необходимости.

Для человека более удобна инфиксная запись, то есть форма записи, в которой знаки операций расположены между операндами. В этой форме будет производиться ввод выражения пользователем.

Чтобы вычислить введенное выражение, необходимо перевести его из инфиксной записи в обратную польскую. Для этого будет использоваться алгоритм Дейкстры.

Алгоритм Дейкстры (или алгоритм сортировочной станции) заключается в следующем [12]:

– проходим исходную строку;

– при нахождении числа, заносим его в выходную строку;

– при нахождении оператора, заносим его в стек;

– выталкиваем в выходную строку из стека все операторы, имеющие приоритет выше рассматриваемого;

– при нахождении открывающейся скобки, заносим ее в стек;

– при нахождении закрывающейся скобки, выталкиваем из стека все операторы до открывающейся скобки, а открывающуюся скобку удаляем из стека.

3.3 Описание данных

Калькулятор будет хранить данные в базе данных. База данных будет содержать информацию о следующих объектах:

– базовые размерности;

– размерности величин;

– величины;

– единицы измерения;

– постоянные.

Размерности величин, согласно международному бюро мер и весов, имеют математическую модель [13], для которой существует специальная формула (1).

$$\dim Q = \mathsf{T}^{\alpha} \mathsf{L}^{\beta} \mathsf{M}^{\gamma} \mathsf{I}^{\delta} \Theta^{\varepsilon} \mathsf{N}^{\zeta} \mathsf{J}^{\eta}, \quad (1)$$

где T, L, M, I, Θ, N и J – базовые размерности;

α, β, γ, δ, ε, ζ и η – показатели базовых размерностей.

Существующее представление размерностей величин в компьютере напоминает предложенную международным бюро мер и весов математическую модель: оно содержит массив, состоящий из показателей всех базовых размерностей [14]. Одна из возможных реализаций существующего представления размерностей величин показана в листинге 1.

Листинг 1 – Существующее представление размерностей величин

```
enum class BasicDimension { T, L, M, I, O, N, J };

using Dimension = std::array<int, 7>;
```

У представления, показанного в листинге 1, есть ряд недостатков. Во-первых, все семь показателей базовых размерностей хранятся в памяти постоянно, даже если они равны нулю. Во-вторых, если добавить еще несколько базовых размерностей, то размер массива увеличится, и это приведет к большему использованию памяти, что говорит о плохой масштабируемости. В-третьих, добавление базовых размерностей возможно только на этапе разработки, то есть во время выполнения приложения добавить их не получится.

Предлагаемое в данной работе представление размерностей в компьютере заключается в хранении вместо массива всех семи показателей словаря, ключом которого является идентификатор базовой размерности в базе данных, а значением – соответствующий ей показатель, но только если он не равен нулю. То есть если величина будет безразмерной, то словарь будет пуст. Реализация предлагаемого представления размерностей величин показана в листинге 2.

Листинг 2 – Предлагаемое представление размерностей величин

```
using Dimension = std::map<int, int>;
```

Такое представление позволит добавить базовые размерности во время выполнения приложения без увеличения потребления памяти и упростить связь между размерностями и величинами.

Величины являются основой базы данных, так как на них завязаны оба вводимых пользователем в выражение объекта: единицы измерения и постоянные.

На основе величин будет производиться вычисление выражения и определение результирующей единицы измерения. Стоит учесть, что операции

сложения и вычитания невозможны между несовместимыми величинами, что будет выявляться и докладываться пользователю.

Единицы измерения будут основываться на величинах. Единицы измерения одной и той же величины будут отличаться *отношением* к стандартной единице международной системы единиц. Так, например, стандартная единица измерения длины в международной системе – метр. Таким образом, сам метр будет иметь отношение 1:1, километр – отношение 1000:1, сантиметр – отношение 0,01:1, а дюйм – отношение 0,0254:1. При вычислении выражения все единицы будут автоматически приводиться в одну систему единиц, используя отношение.

Если при вычислении выражения какой-то из единиц измерения не окажется в базе данных, то сгенерируется составная единица измерения с наименьшим количеством составляющих.

Постоянные не будут иметь прямой связи с единицами измерения, а будут основаны, подобно самим единицам измерения, на величине. Это связано с тем, что часть постоянных имеет составные единицы измерения. Например, постоянная Планка $h = 6{,}62607015 \cdot 10^{-34}$ кг·м^2·с$^{-1}$ (Дж·с).

Схема базы данных, содержащая все вышеперечисленные объекты, представлена на рисунке 1.

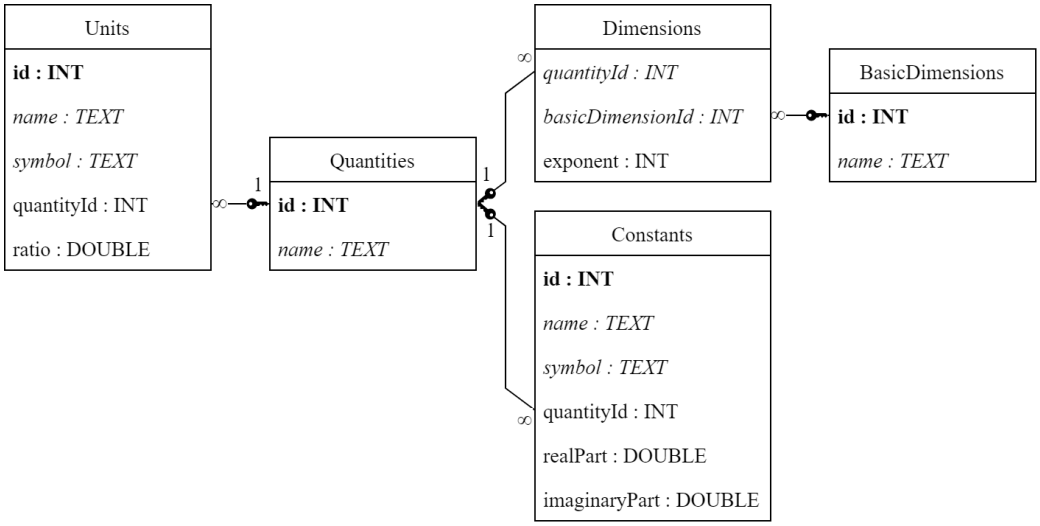

Рисунок 1 – Схема базы данных

На рисунке 1 **жирным** выделены поля, которые являются первичными ключами, *курсивом* выделены уникальные поля (в таблице Dimensions уникальна пара quantityId+basicDimensionId). Скрипты создания таблиц в базе данных представлены в приложении А.

Назначение таблиц указано в таблице 2.

Таблица 2 – Назначение таблиц базы данных

Наименование таблицы	**Назначение**
BasicDimensions	Хранение словаря базовых размерностей
Quantities	Хранение словаря величин
Dimensions	Распределение размерностей (массивов показателей базовых размерностей) по величинам
Units	Хранение данных о единицах измерения
Constants	Хранение данных о константах

Поля таблиц базы данных, указанных в таблице 2, описаны в таблице 3.

Таблица 3 – Назначение полей таблиц базы данных

Наименование поля	**Назначение**
BasicDimensions	
id	Идентификатор базовой размерности
name	Название базовой размерности
Quantities	
id	Идентификатор величины
name	Название величины
Dimensions	
quantityId	Идентификатор величины
basicDimensionId	Идентификатор базовой размерности
exponent	Показатель базовой размерности

Окончание таблицы 3

Units	
id	Идентификатор единицы измерения
name	Название единицы измерения
symbol	Символ единицы измерения, который будет отображаться в поле ввода
quantityId	Идентификатор величины
ratio	Отношение единицы измерения к базовой
Constants	
id	Идентификатор константы
name	Имя константы
symbol	Символ константы, который будет отображаться в поле ввода
quantityId	Идентификатор величины
realPart	Значение вещественной части комплексного числа
imaginaryPart	Значение мнимой части комплексного числа

Как видно из таблицы 3, все таблицы базы данных построены на идентификаторах. С помощью этих идентификаторов будут конструироваться объекты классов.

3.4 Графический интерфейс пользователя

Графический интерфейс пользователя будет состоять из следующих окон:

– главное окно;

– окно добавления единиц измерения;

– окно добавления постоянных.

Главное окно будет содержать три вкладки: с основным функционалом, с единицами измерения и с постоянными.

Вкладка с основным функционалом будет включать все доступные в калькуляторе операции и цифровой блок.

Вкладки с единицами измерения и с постоянными будут отображать список всех имеющихся в базе данных соответствующих объектов. С этих вкладок также можно будет добавить новые объекты в базу данных.

Окна добавления объектов будут содержать все доступные в базе данных поля, кроме идентификатора.

Окно добавления единиц измерения позволит ввести для нового объекта название, символ, величину и отношение.

У полей ввода окна добавления единиц измерения есть ряд ограничений.

Название может быть произвольным, но оно должно быть уникальным. Символ может содержать только буквы. В поля ввода названий и символов можно вставить скопированный текст, что позволит пользователю ввести, например, греческие буквы. Величина будет выбираться из выпадающего списка, данные для которого будут браться из базы данных. Отношение может быть только действительным числом.

Окно добавления постоянных позволит ввести для нового объекта название, символ, величину и значение.

У полей ввода окна добавления постоянных те же ограничения, что и у соответствующих полей ввода предыдущего окна, а в поля ввода вещественной и мнимой частей комплексного числа можно вводить только числа.

Схема связи всех вышеперечисленных окон представлена на рисунке 2.

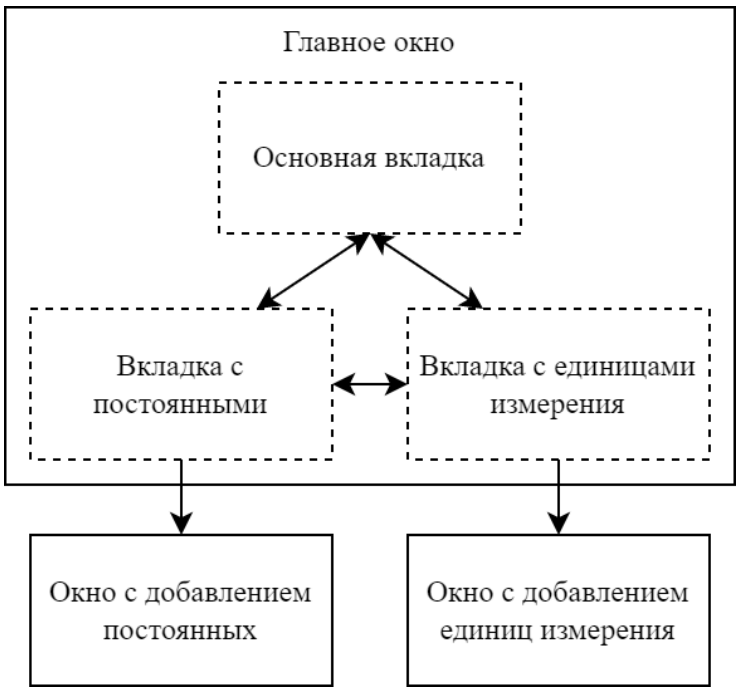

Рисунок 2 – Схема связи окон

Как видно из рисунка 2, приложение будет иметь три различных окна, где главное будет иметь три вкладки.

4 РЕАЛИЗАЦИЯ СИСТЕМЫ

4.1 Реализация классов

На рисунке 3 изображена диаграмма основных классов калькулятора.

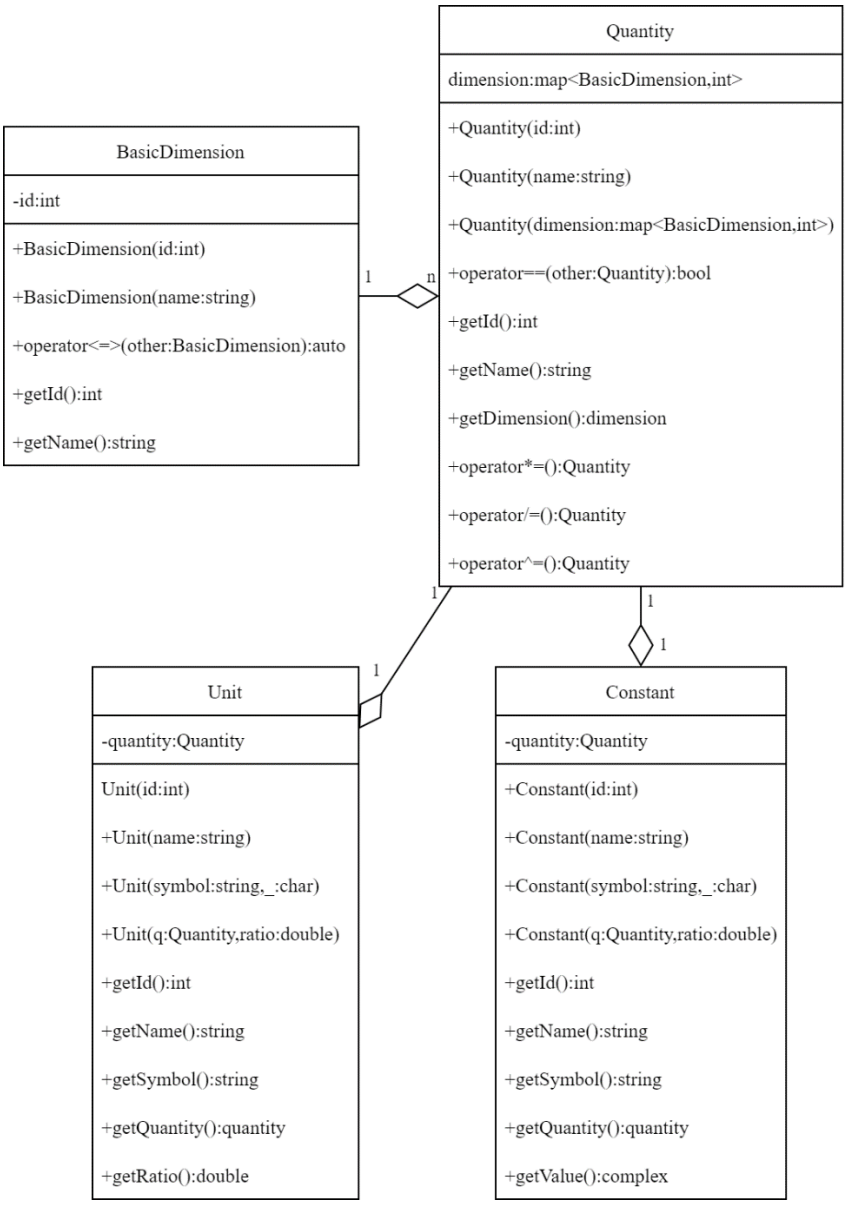

Рисунок 3 – Диаграмма основных классов

Как видно из рисунка 3, диаграмма классов очень похожа на схему базы данных, изображенную на рисунке 1, так как здесь величины также являются центральным звеном. Назначение классов указано в таблице 4.

Таблица 4 – Описание классов

Класс	Назначение
BasicDimension	Ключ словаря показателей базовых размерностей
Quantity	Работа со словарем показателей базовых размерностей
Unit	Хранение свойств единиц измерения
Constant	Хранение свойств постоянных, операнд

Описание методов классов из таблицы 4 приведено в таблице 5.

Таблица 5 – Описание методов классов

Метод	Назначение
BasicDimension	
`static void create()`	Создание таблицы «BasicDimensions» в базе данных
`static void insert(const QString &name)`	Вставка объекта в таблицу
`BasicDimension(const int &id = 0)`	Конструктор объекта по идентификатору из базы данных
`BasicDimension(const char *name)`	Конструктор объекта по имени из базы данных
`auto operator<=>(` `const BasicDimension &) const`	Оператор сравнения для возможности установки объекта в качестве ключа словаря
`const int &getId() const`	Получение идентификатора объекта в базе данных
`QString getName() const`	Получение имени объекта из базы данных
Quantity	
`static void create()`	Создание таблиц «Quantities» и «Dimensions» в базе данных

Продолжение таблицы 5

`static void insert(`
`const QString &name,`
`const Dimension &dimension)`
	Вставка объекта в таблицу (Dimension – это синоним класса словаря: `using Dimension = std::map<BasicDimension, int>`)

`Quantity(const int &id = 0)`	Конструктор объекта по идентификатору из базы данных
`Quantity(const char *name)`	Конструктор объекта по имени из базы данных
`Quantity(const Dimension &dimension)`	Конструктор объекта по словарю показателей базовых размерностей
`bool operator==(const Quantity &other) const`	Оператор сравнения
`int getId() const`	Получение идентификатора объекта из базы данных
`QString getName() const`	Получение имени объекта из базы данных
`const Dimension &getDimension() const`	Получение словаря показателей базовых размерностей
`Quantity &operator*=(` `const Quantity &other)`	Оператор умножения
`Quantity &operator/=(` `const Quantity &other)`	Оператор деления
`Quantity &operator^=(` `const double &value)`	Оператор возведения в степень
Unit	
`static void create()`	Создание таблицы «Units» в базе данных

`static void insert(const QString &name,`
`const QString &symbol,`
`const Quantity &quantity,`
`const double &ratio)`
	Вставка объекта в таблицу

`Unit(const int &id = 0)`	Конструктор объекта по идентификатору из базы данных
`Unit(const char *name)`	Конструктор объекта по имени из базы данных
`Unit(const char *symbol, const char &)`	Конструктор объекта по символу из базы данных

Окончание таблицы 5

`Unit(const Quantity &quantity, const double &ratio)`	Конструктор объекта по величине и отношению
`int getId() const`	Получение идентификатора объекта из базы данных
`QString getName() const`	Получение имени объекта из базы данных
`QString getSymbol() const`	Получение символа объекта из базы данных
`const Quantity &getQuantity() const`	Получение величины
`const double &getRatio() const`	Получение отношения
Constant	
`static void create()`	Создание таблицы «Constants» в базе данных
`static void insert(const QString &name, const QString &symbol, const Quantity &quantity, const Complex &complex)`	Вставка объекта в таблицу
`Constant(const int &id = 0)`	Конструктор объекта по идентификатору из базы данных
`Constant(const QString &name)`	Конструктор объекта по имени из базы данных
`Constant(const QString &symbol, const char &)`	Конструктор объекта по символу из базы данных
`Constant(const Quantity &quantity, const Complex &complex)`	Конструктор объекта по величине и значению
`int getId() const`	Получение идентификатора объекта из базы данных
`QString getName() const`	Получение имени объекта из базы данных
`QString getSymbol() const`	Получение символа объекта из базы данных
`const Quantity &getQuantity() const`	Получение величины
`const Complex &getValue() const`	Получение значения

Как видно из таблицы 5, все классы жестко привязаны к базе данных. Это позволяет искать объекты в базе данных с помощью запросов как по их

идентификатору, так и по имени без задействования сторонних классов, которым был бы необходим доступ к закрытым полям.

4.2 Ввод и вычисление выражения

Ввод выражения будет производиться со встроенной в приложение клавиатуры. Особенностью его синтаксиса будет являться то, что постоянные и единицы измерения будут автоматически обрамляться в фигурные и квадратные скобки соответственно. Это сделано как для улучшения форматирования выражения, а именно выделения постоянных и единиц измерения цветом, так и для обнаружения их при вычислении.

Вычисление выражения будет делиться на несколько этапов:

– разбиение выражения на токены;

– проверка синтаксиса;

– перевод в обратную польскую запись и вычисление.

Разбиение выражения на токены упростит как проверку синтаксиса, так и перевод выражения в обратную польскую запись. Токенами будут считаться операнды, операторы и идентификаторы.

Схема алгоритма разбиения на токены изображена на рисунке 4.

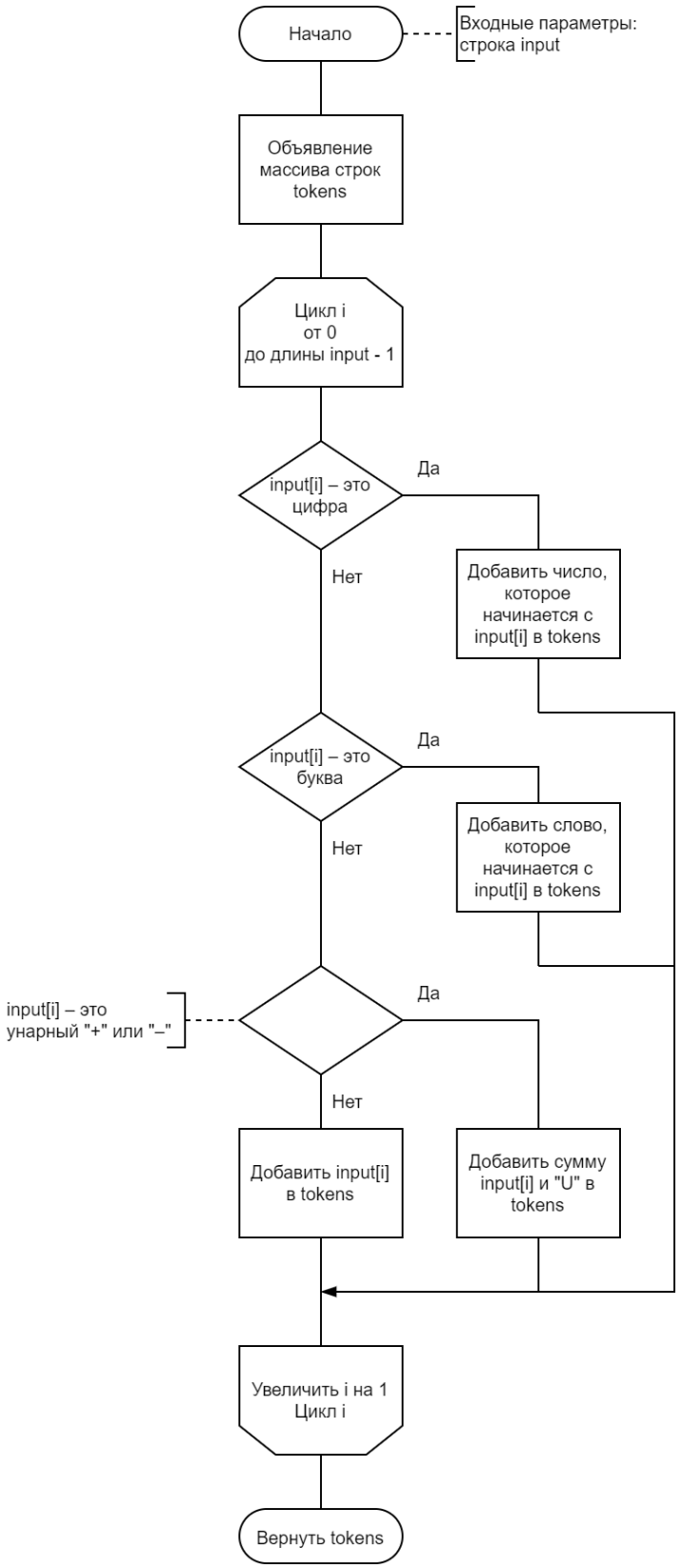

Рисунок 4 – Схема алгоритма разбиения на токены

Алгоритм разбиения на токены, соответствующий схеме алгоритма разбиения на токены из рисунка 4, на языке C++ представлен в листинге 3.

Листинг 3 – Алгоритм разбиения выражения на токены

```
QVector<QString> parse(const QString &input)
{
    QVector<QString> tokens;

    for (int i = 0; i < input.length(); i++)
    {
        QString token;

        // Если токен начинается с цифры:
        if (input[i].isDigit())
        {
            // Запомнить его начальный индекс;
            int position = i;

            // Идти по строке, пока она не закончится, либо пока продолжается число
            // (учтены записи чисел вида "#", "#.#", "#E#", "#.#E#", "#E-#", "#.#E-#", где "#" –
            // произвольный набор идущих подряд цифр, "." – десятичный разделитель, "E" – *10^);
            while (i + 1 < input.length() && (
                   input[i + 1].isDigit() ||
                   input[i + 1] == QLocale().decimalPoint() ||
                   input[i + 1].toLower() == 'e' || (
                   input[i + 1] == '-' && input[i].toLower() == 'e'))) {
                i++;
            }

            // Запомнить токен;
            token = input.mid(position, i - position + 1);
        }
        // Если токен начинается с буквы, проделать аналогичные действия, допуская только буквы;
        else if (input[i].isLetter())
        {
            int position = i;

            while (i + 1 < input.length() && (
                   input[i + 1].isLetter())) {
                i++;
            }

            token = input.mid(position, i - position + 1);
        }
        // Если токен начинается с символа:
        else
        {
            // Запомнить токен;
            token = input.mid(i, 1);

            // Если токен является унарным плюсом или минусом:
            if ((token == "+" || token == "-") && (i == 0 || (
                !input[i - 1].isDigit() && input[i - 1] != ')' && input[i - 1] != '!'))) {
                // Добавить к токену "U";
                token += "U";
            }
        }

        // Добавить токен в массив токенов.
        tokens.append(token);
    }

    return tokens;
}
```

Как видно из листинга 3, разбиение на токены разделяет строку на массив строк, что значительно упростит проверку выражения.

Проверка выражения полностью построена на токенах: сначала они разбиваются на группы (начало выражения, функция, операнд, бинарный

оператор, унарный оператор, постфиксный оператор, начало единицы измерения или постоянной, их символ, их конец, левая скобка, правая скобка), а затем на основе предыдущего типа токена задаются ожидаемые. В листинге 4 приведена карта ожидаемых токенов на основе предыдущего.

Листинг 4 – Карта ожидаемых токенов

```
QMap<QString, QVector<QString>> expectedTokens {
    { "BEGIN",    { "FUNCTION", "OPERAND", "LEFTBR", "UNARY", "IDBEGIN" } },
    { "FUNCTION", { "LEFTBR" } },
    { "OPERAND",  { "BINARY", "POSTFIX", "IDBEGIN", "END", "RIGHTBR" } },
    { "BINARY",   { "BEGIN", "FUNCTION", "OPERAND", "LEFTBR", "UNARY", "IDBEGIN" } },
    { "UNARY",    { "BEGIN", "FUNCTION", "OPERAND", "LEFTBR", "IDBEGIN" } },
    { "POSTFIX",  { "BINARY", "POSTFIX", "IDBEGIN", "END", "RIGHTBR" } },
    { "IDBEGIN",  { "ID" } },
    { "ID",       { "IDEND" } },
    { "IDEND",    { "BINARY", "POSTFIX", "END", "RIGHTBR" } },
    { "LEFTBR" ,  { "FUNCTION", "OPERAND", "LEFTBR", "UNARY", "IDBEGIN" } },
    { "RIGHTBR" , { "BINARY", "POSTFIX", "IDBEGIN", "END", "RIGHTBR" } }
};
```

Как видно из листинга 4, например, в начале выражения ("BEGIN") ожидаются только функции, операнды, левая скобка, унарные операторы и начало единицы измерения или постоянной, а после операндов ("OPERAND") – бинарный оператор, постфиксный оператор, начало единицы измерения или постоянной, конец выражения и правая скобка.

Благодаря карте можно будет расширить количество операторов, не изменяя при этом алгоритм проверки: достаточно будет указать, к какой группе принадлежит тот или иной новый оператор.

Программная реализация контроля корректности синтаксиса и логики введенного пользователем задания на вычисление приведена в приложении Б.

Вычисление выражения с его переводом в обратную польскую запись реализовано по алгоритму, описанному в подразделе 2 раздела 3.

4.3 Графический интерфейс пользователя

Как было сказано в разделе 3, интерфейс калькулятора будет иметь три окна. Главное окно будет иметь три вкладки. Скриншот основной вкладки главного окна представлен на рисунке 5.

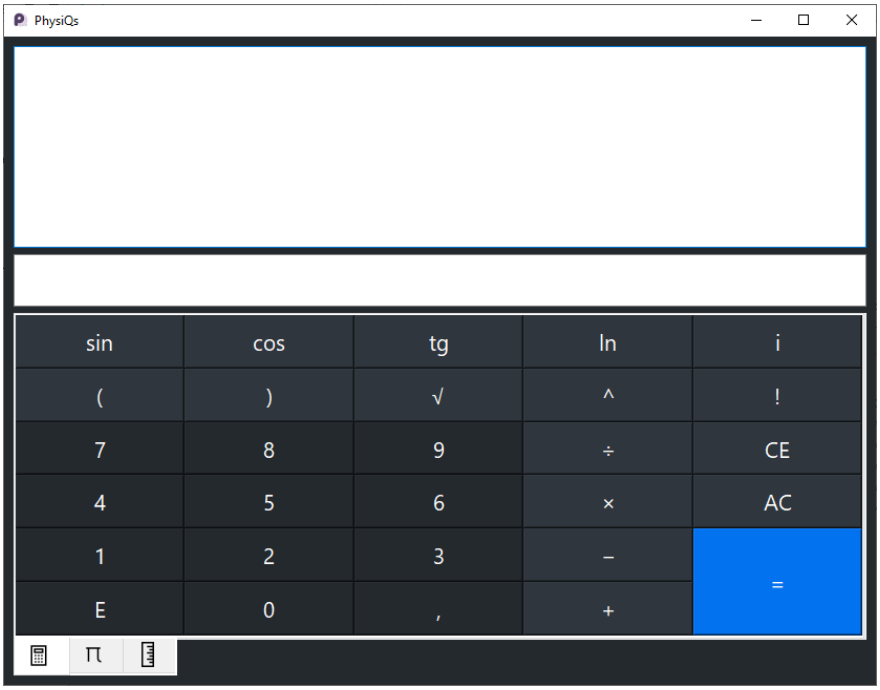

Рисунок 5 – Основная вкладка

Как видно из рисунка 5, главное окно имеет поле ввода выражения и поле вывода результата во время ввода выражения. Эти поля будут общими для всех вкладок. Основная вкладка содержит все имеющиеся в калькуляторе математические операторы, а также цифровой блок. С нее можно перейти на вкладки с единицами измерения и постоянными.

На рисунке 6 изображена вкладка с постоянными.

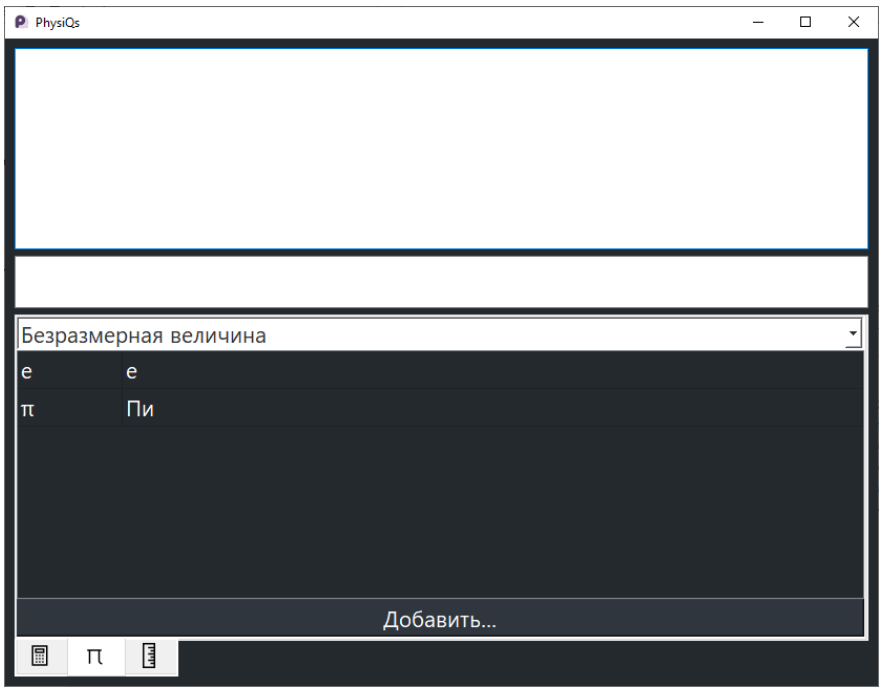

Рисунок 6 – Вкладка с постоянными

Как видно из рисунка 6, на вкладке с постоянными есть список имеющихся в калькуляторе постоянных. При нажатии на элемент списка в поле ввода будет выводиться символ постоянной. На этой вкладке предусмотрен выпадающий список для группировки постоянных по величине. Это сделано для того, чтобы пользователь быстро находил постоянные, так как их может быть огромное количество.

Со вкладки с постоянными пользователь может перейти в окно добавления постоянных, которое изображено на рисунке 7.

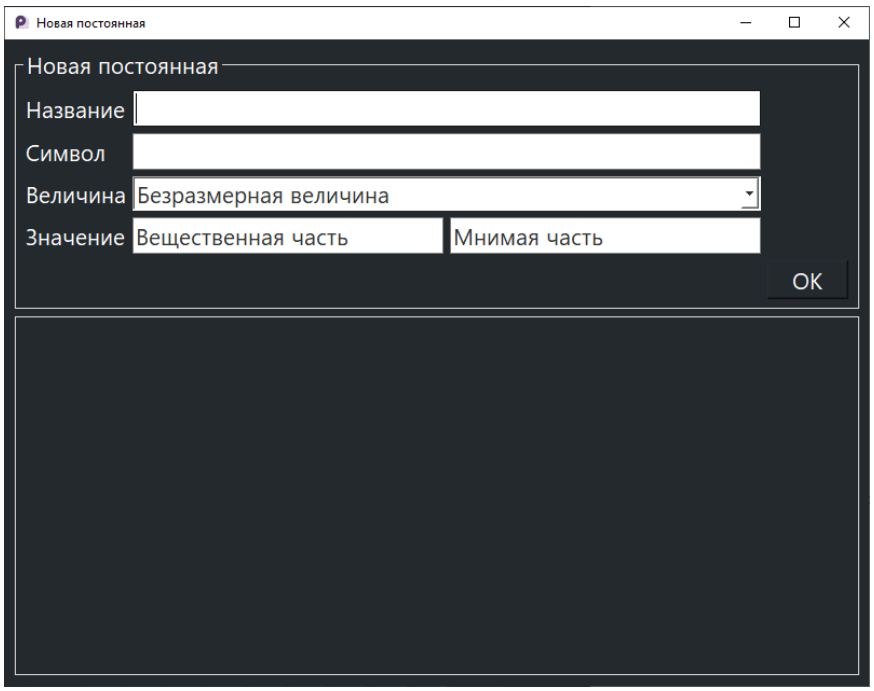

Рисунок 7 – Окно добавления постоянных

Как видно из рисунка 7, для новой постоянной можно выбрать ее название, символ, величину из выпадающего списка, а также значения обеих частей комплексного числа.

Вкладка главного окна с единицами измерения представлена на рисунке 8.

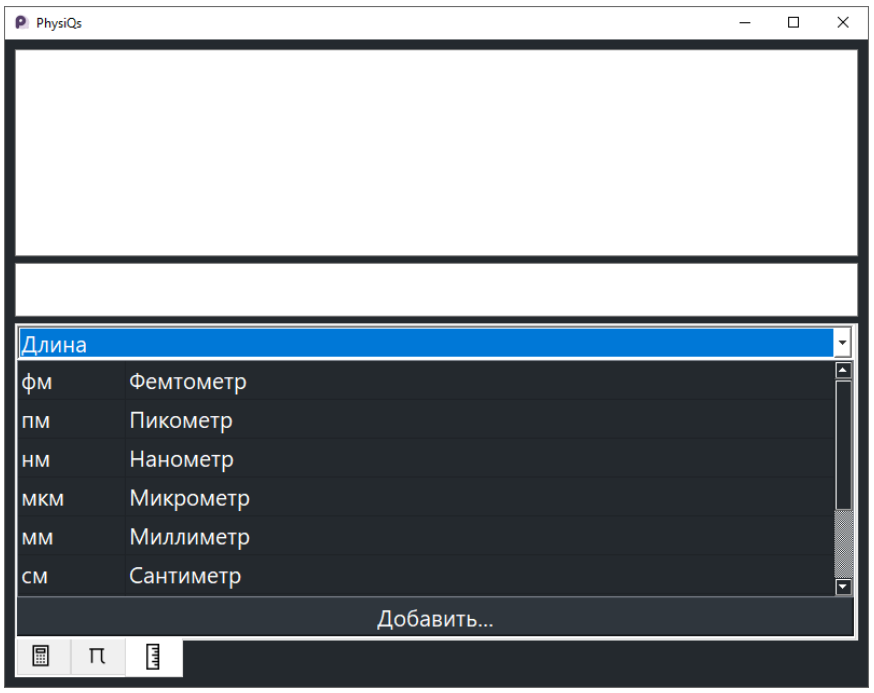

Рисунок 8 – Вкладка с единицами измерения

Как видно из рисунка 8, на вкладке с единицами измерения можно выбрать для ввода нужную единицу измерения. Так как в базе данных может быть неограниченное их число, пользователь в целях экономии времени поиска нужной может отфильтровать список по величинам. Также со вкладки с единицами измерения можно добавить новую. Добавление новых единиц измерения происходит в отдельном окне, изображенном на рисунке 9.

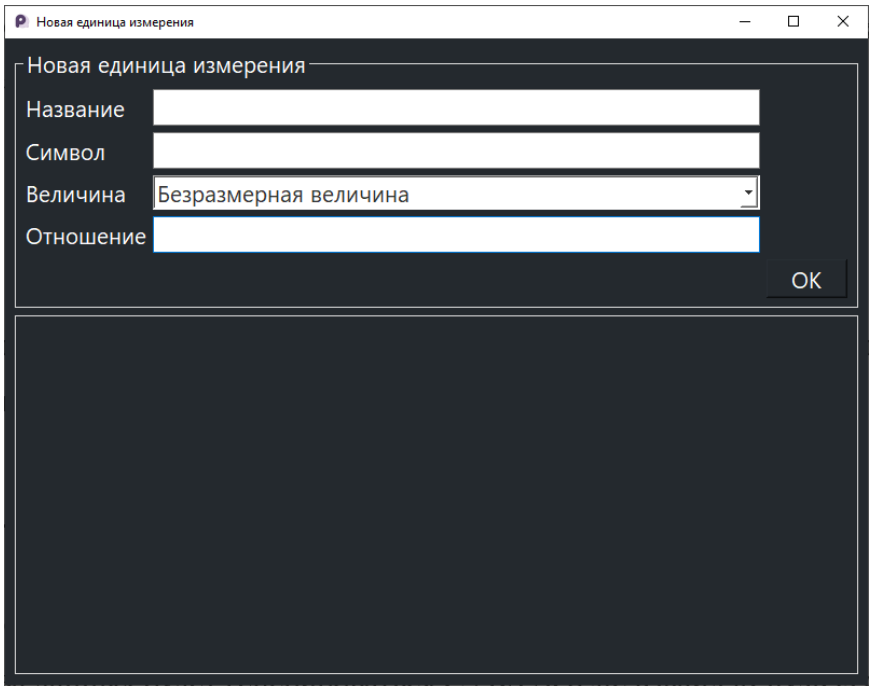

Рисунок 9 – Экран добавления единиц измерения

Как видно из рисунка 9, новой единице измерения можно задать название, символ, величину из выпадающего списка и отношение.

4.4 Файловая структура

Приложение должно хранить в постоянной памяти базу данных. Она будет генерироваться системой управления базами данных SQLite в файл «PhysiQs.db». Путь сохранения базы данных зависит от платформы. На Windows путь следующий: «%LOCALAPPDATA%\PhysiQs»; на Android: «/data/ru.susu.physiqs».

5 ТЕСТИРОВАНИЕ

Проведем тестирование калькулятора. Тестированию подлежат: ввод выражения в главном окне, ввод данных о новой постоянной и о новой единице измерения.

5.1 Тестирование ввода выражения

При вводе выражения пользователь может допустить синтаксическую или логическую ошибку, о чем он должен быть осведомлен.

Синтаксическая ошибка может заключаться в неправильной расстановке скобок, в незаконченном вводе выражения или в неправильно введенных числах. На рисунке 10 показано поведение программы при синтаксической ошибке, например, в случае, если неправильно расставлены скобки.

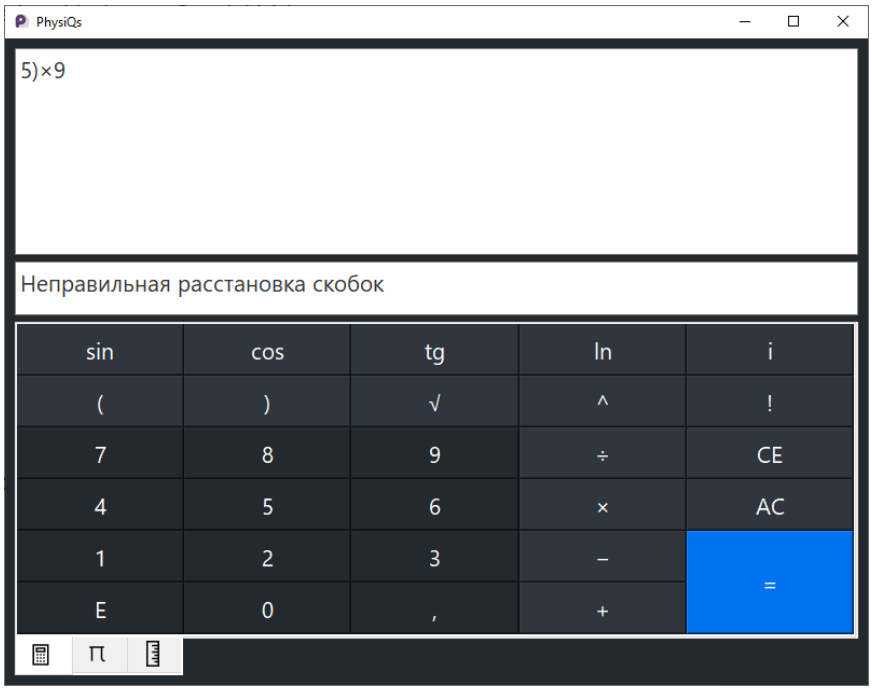

Рисунок 10 – Поведение при синтаксической ошибке

Как видно из рисунка 10, калькулятор смог выявить синтаксическую ошибку и сообщил о ней пользователю.

Логическая ошибка заключается в проведении недопустимых операций над операндами. Учтены следующие недопустимые операции: сложение или вычитание операндов с разными физическими величинами, умножение или деление операндов с одинаковыми величинами, но с единицами измерения разных систем единиц, и возведение в степень операндов с единицами измерения (приложение Б). На рисунке 11 показано поведение программы при логической ошибке, например, при сложении операндов с несовместимыми единицами измерения.

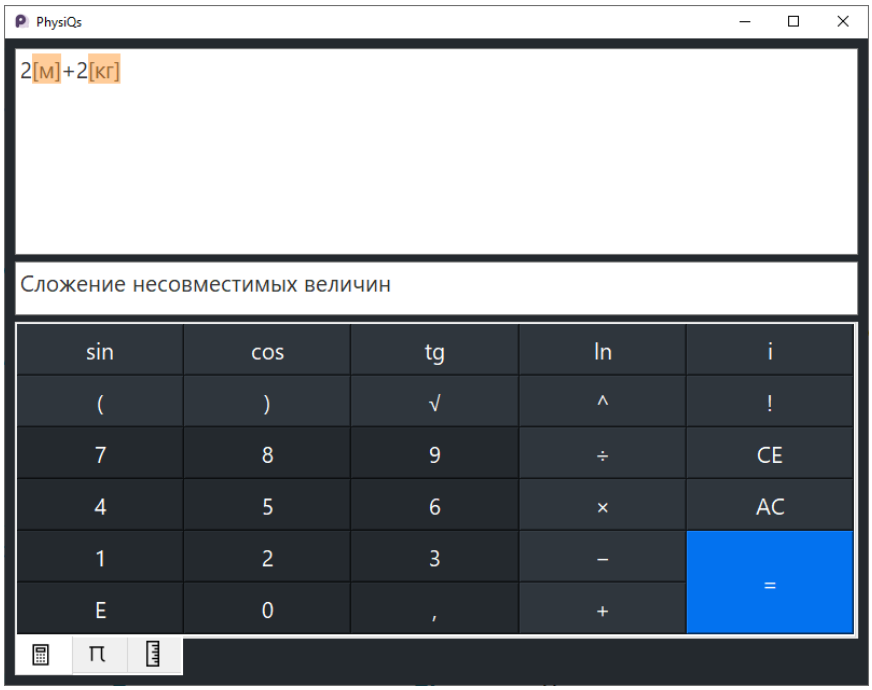

Рисунок 11 – Поведение при логической ошибке

Как видно из рисунка 11, калькулятор смог выявить логическую ошибку и сообщил о ней пользователю.

Исправив все ошибки, пользователь должен будет получить результат выражения. На рисунке 12 показано поведение при исправленных ошибках.

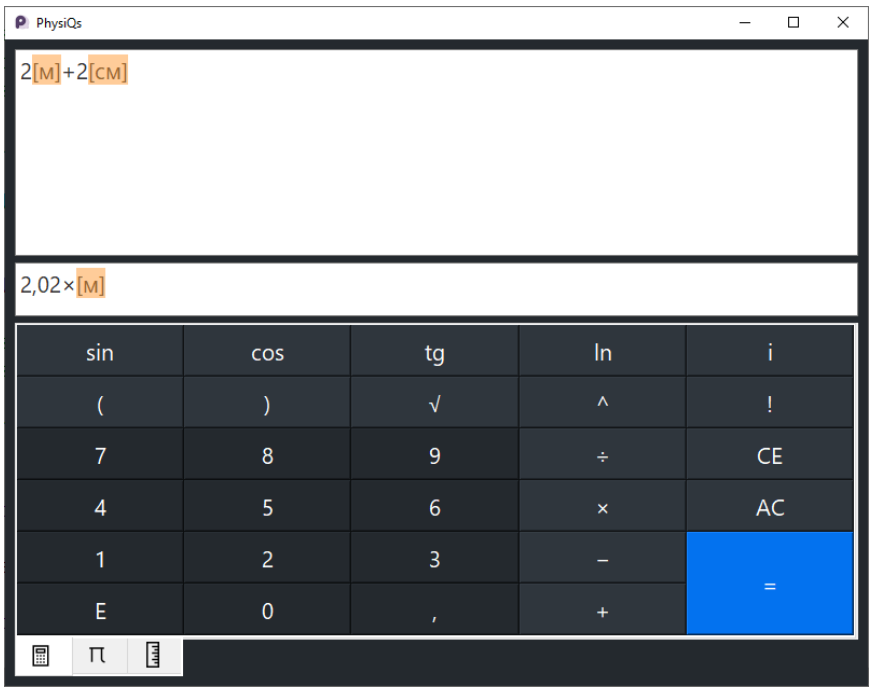

Рисунок 12 – Поведение при исправленных ошибках

Как видно из рисунка 12, калькулятор не обнаружил ошибок и мгновенно вывел ответ, приведя все в одну систему единиц.

5.2 Создание постоянной или единицы измерения

Создание новых постоянных и единиц измерения для их дальнейшего добавления в базу данных происходит в отдельных окнах.

Окно добавления постоянных имеет четыре поля ввода: поле ввода названия, символа и вещественной и мнимой частей комплексного числа. При нажатии на кнопку «OK» и успешной проверке всех полей ввода объект с введенными данными добавляется в базу данных. При неуспешной – калькулятор выведет пользователю сообщение об ошибке.

Поле ввода названия ограничений не имеет.

Поле ввода символа имеет следующее ограничение: символ может содержать только буквы. На рисунке 13 показано поведение калькулятора при неправильно введенном символе.

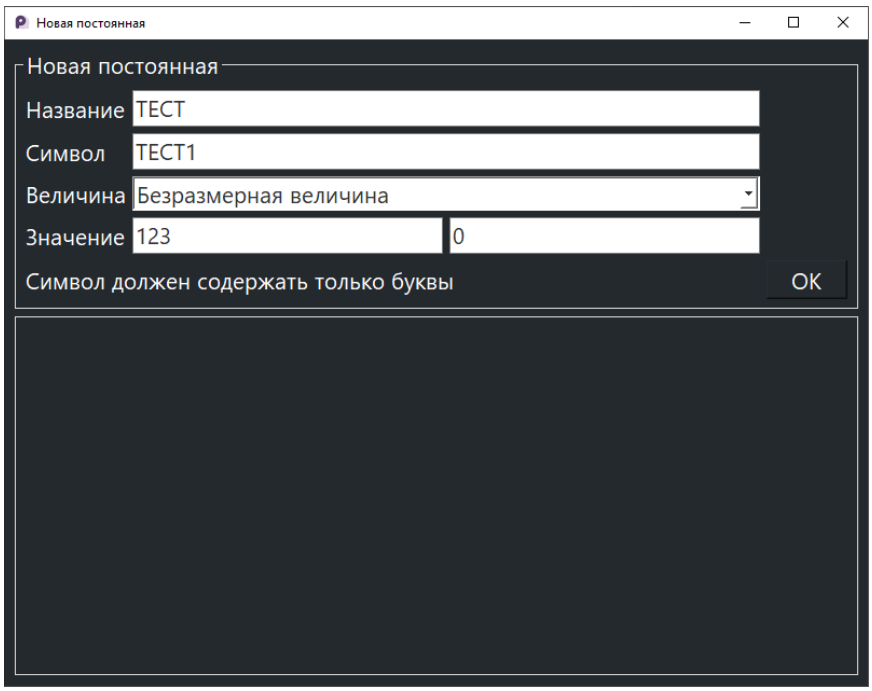

Рисунок 13 – Поведение при неправильном символе

Как видно из рисунка 13, калькулятор уведомил пользователя об ошибочном вводе символа.

Поля ввода вещественной и мнимой частей комплексного числа имеют одинаковые ограничения: они должны принимать только числа. На рисунке 14 показано поведение программы при неправильном вводе мнимой части.

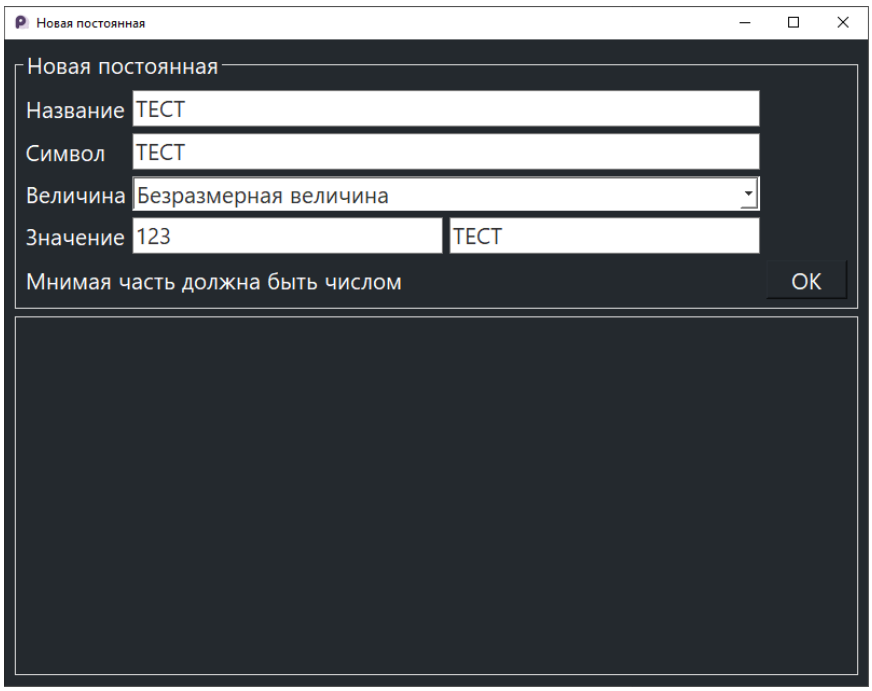

Рисунок 14 – Поведение при неправильном комплексном числе

Как видно из рисунка 14, калькулятор уведомил пользователя о необходимости исправить мнимую часть комплексного числа.

Введем верные данные, как показано на рисунке 15.

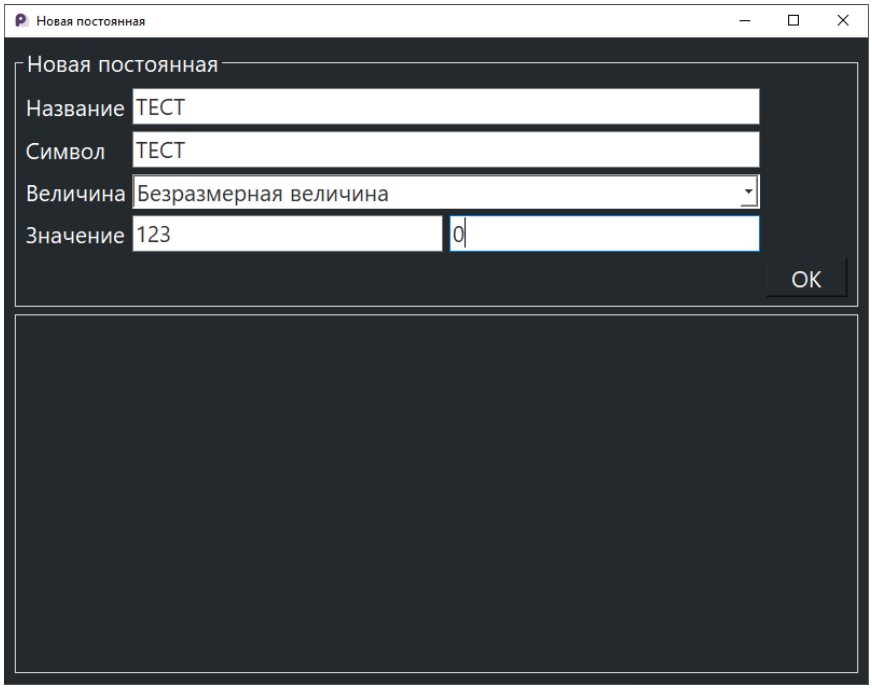

Рисунок 15 – Ввод верных данных

При нажатии на кнопку «OK» на рисунке 15 объект должен добавиться в базу данных. Проверим это и попробуем ввести его на рисунке 16.

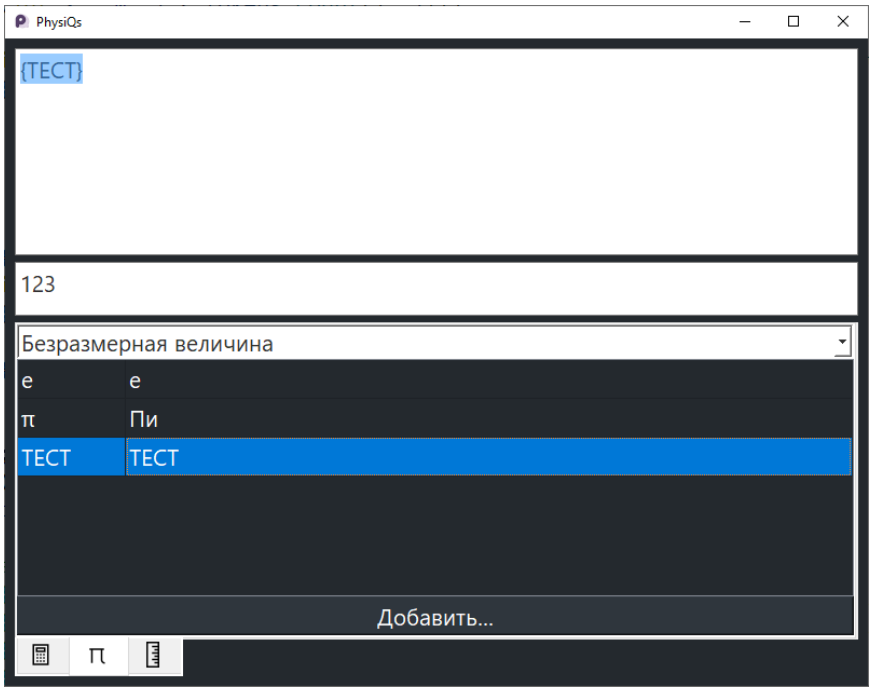

Рисунок 16 – Ввод добавленной постоянной

Как видно из рисунка 16, постоянная была добавлена в базу данных и в список постоянных, а также при ее вводе выводится ее значение.

Окно добавления единиц измерения похоже по своей структуре на окно ввода постоянных и отличается только тем, что поле отношения не может быть меньше нуля или равно нулю. На рисунке 17 показано поведение калькулятора при неправильном вводе отношения.

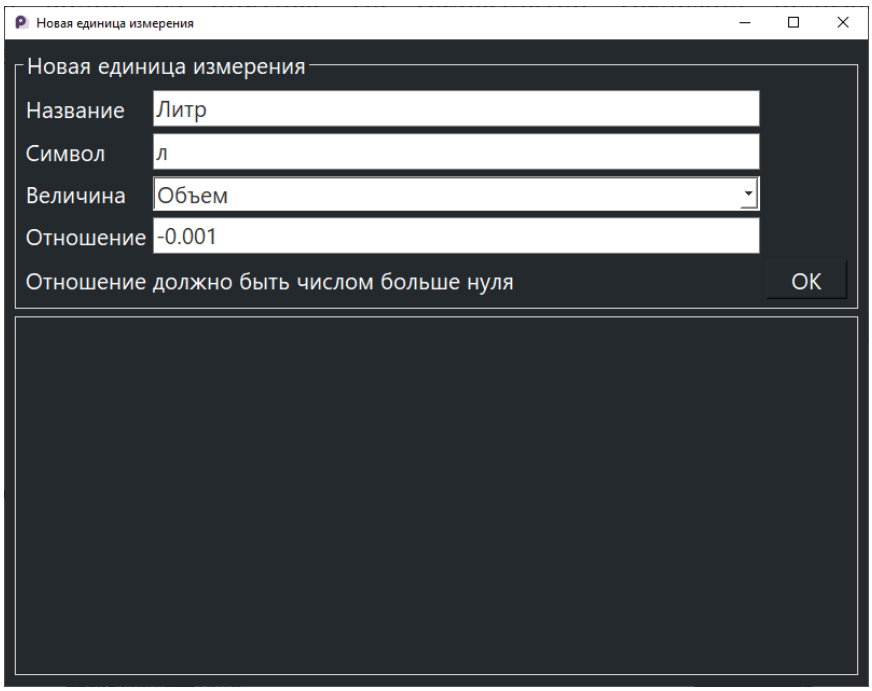

Рисунок 17 – Поведение при неверном отношении

Как видно из рисунка 17, калькулятор уведомил пользователя о необходимости исправления ошибки.

Введем верные данные, как показано на рисунке 18.

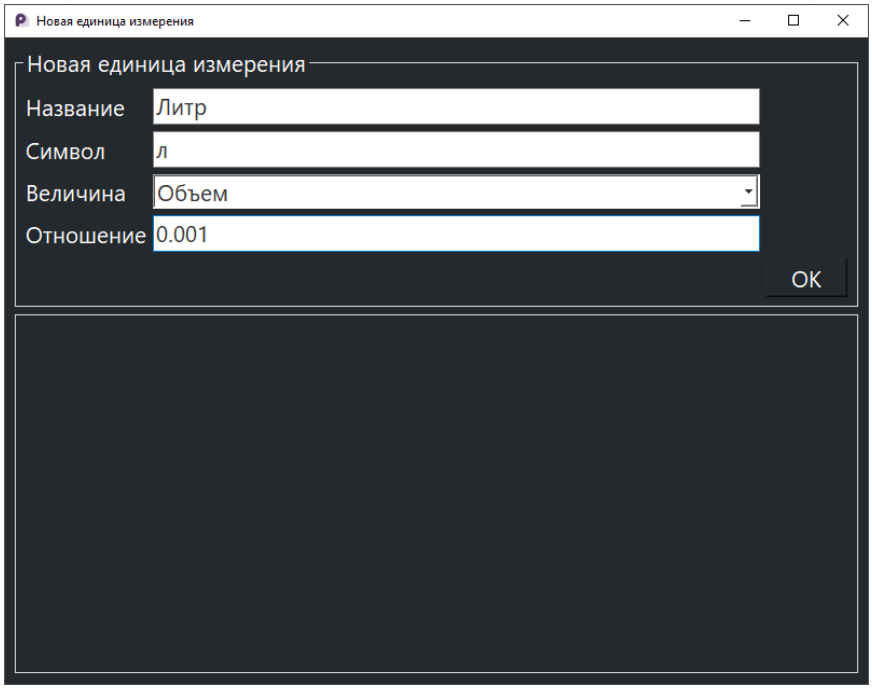

Рисунок 18 – Ввод верных данных

При нажатии на кнопку «OK» на рисунке 18 объект должен добавиться в базу данных. Проверим это и попробуем ввести его на рисунке 19.

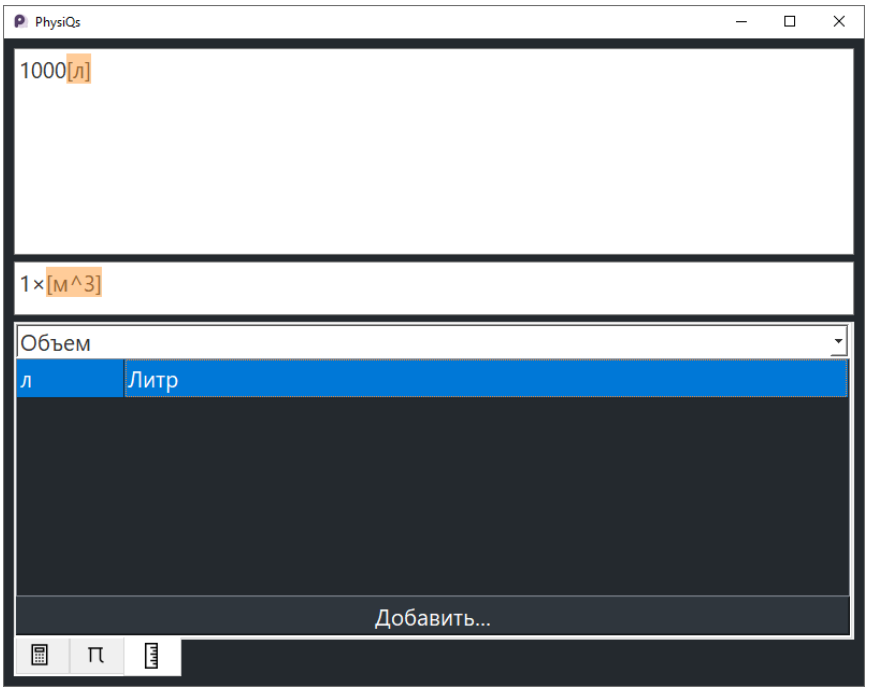

Рисунок 19 – Ввод добавленной единицы измерения

Как видно из рисунка 19, единица измерения была добавлена в базу данных и в список единиц измерения, а также при ее вводе вывелся результат со стандартной единицей международной системы единиц.

5.3 Тестирование на разных операционных системах

Так как одно из требований калькулятора – кроссплатформенность, запустим его на различных операционных системах.

В требованиях к кроссплатформенности указана операционная система Android версии не ниже 9. Отобразим сведения о тестируемом устройстве на рисунке 20.

Рисунок 20 – Сведения об устройстве на Android

Как видно из рисунка 20, Версия Android тестируемого устройства равна 9. Запустим калькулятор на этом устройстве и отобразим это на рисунке 21.

Рисунок 21 – Калькулятор на устройстве с Android 9

Как видно из рисунка 21, приложение успешно было запущено на Android версии 9.

В требованиях по кроссплатформенности также указана операционная система Windows не ниже 10. Отобразим данные о тестовом стенде на рисунке 22.

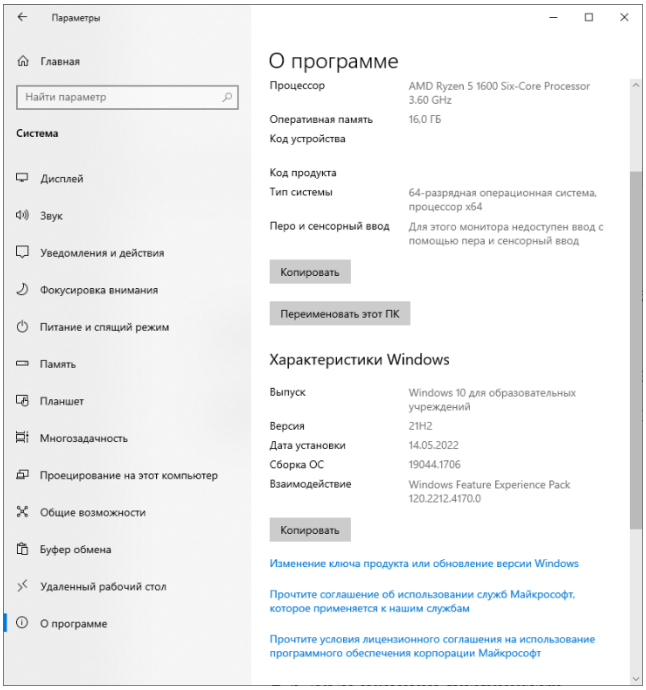

Рисунок 22 – Сведения об устройстве на Windows

Как видно из рисунка 22, на тестовом стенде установлена операционная система Windows 10. Запустим калькулятор на данном стенде и отобразим это на рисунке 23.

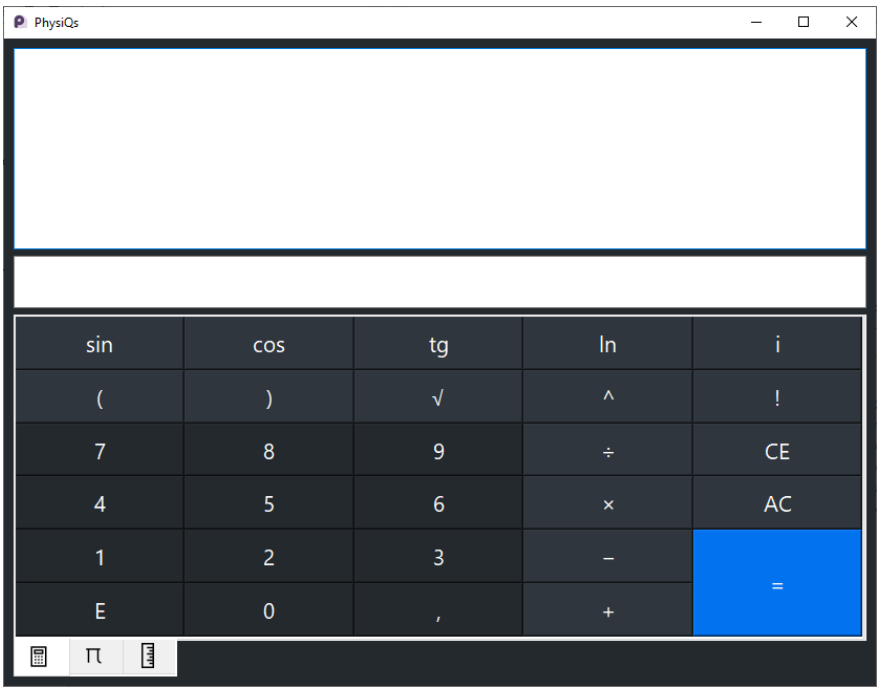

Рисунок 23 – Калькулятор на устройстве с Windows 10

Как видно из рисунка 23, приложение успешно было запущено на Windows 10.

5.4 Дальнейшая разработка

Для дальнейшей разработки предусмотрена возможность локализации приложения для англоязычных пользователей с помощью средства Qt Linguist. Локализации будут подвергаться не только надписи в графическом интерфейсе, но и десятичный разделитель («.» вместо «,») и названия функций (например, «tan» вместо «tg»).

Также при дальнейшей разработке можно добавить в калькулятор такие функции, как выбор выводимой системы единиц, обнаружение нарушения физического смысла, пошаговый перевод в одну систему единиц и историю вычислений.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы был произведен анализ современных технологий для вычисления физических величин. Была организована разработка программного комплекса для вычисления физических величин с использованием среды разработки кроссплатформенных приложений Qt. Были рассмотрены преимущества и недостатки, как модели вычислений физических величин в целом, так и конкретного программного комплекса. Была разработана архитектура предлагаемого решения и доказана ее способность к обеспечению успешного вычисления физических величин.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Приложения в Google Play – Калькулятор. – Текст : электронный // Google LLC : [сайт]. – URL: https://play.google.com/store/apps/details?id=com.google.android.calculator&hl=ru (дата обращения: 08.03.2022).

2. Home – Photomath. – Текст : электронный // Photomath, Inc. : [сайт]. – URL: https://photomath.com/ru/ (дата обращения: 08.03.2022).

3. SMath Studio – SMath. – Текст : электронный // Андрей Ивашов : [сайт]. – URL: https://ru.smath.com/обзор/SMathStudio/резюме (дата обращения: 08.03.2022).

4. Mathcad: Math Software for Engineering Calculations | Mathcad : [официальный сайт] / PTC. – URL: https://www.mathcad.com/en (дата обращения: 08.03.2022). – Текст : электронный.

5. Wolfram|Alpha Tour. – Текст : электронный // Wolfram|Alpha : [сайт]. – URL: https://www.wolframalpha.com/tour/ (дата обращения: 09.03.2022).

6. Лучший конвертер единиц измерения - Google Play. – Текст : электронный // Google LLC : [сайт]. – URL: https://play.google.com/store/apps/details?id=com.kuzmin.konverter&hl=ru (дата обращения: 24.05.2022).

7. Краткий обзор языка C#. – Текст : электронный // Microsoft : [сайт]. – URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (дата обращения: 09.03.2022).

8. Oracle Java. – Текст : электронный // Oracle : [сайт]. – URL: https://www.oracle.com/ru/java/ (дата обращения: 09.03.2022).

9. Cross Platform IDE Qt Creator. – Текст : электронный // The Qt Company : [сайт]. – URL: https://www.qt.io/product/development-tools (дата обращения: 09.03.2022).

10. Qt Documentation Home. – Текст : электронный // The Qt Company : [сайт]. – URL: https://doc.qt.io/ (дата обращения: 01.06.2022).

11. SQLite Home Page : [официальный сайт] / SQLite. – URL: https://www.sqlite.org/index.html (дата обращения: 09.03.2022). – Текст : электронный.

12. Обратная польская нотация или как легко распарсить алгебраическое выражение / Хабр. – Текст : электронный // Хабр : [сайт]. – 21 декабря 2021. – URL: https://habr.com/ru/post/596925/ (дата обращения: 30.04.2021).

13. International System of Units (SI) / Международное бюро мер и весов. – 9-е издание. – 2019. – 216 с. – ISBN 978-92-822-2272-0. – Текст : электронный // Международное бюро мер и весов : [официальный сайт]. – Севр, Франция. – URL: https://www.bipm.org/documents/20126/41483022/SI-Brochure-9-EN.pdf (дата обращения: 30.11.2021).

14. IfcDimensionalExponents. – Текст : электронный // buildingSMART International, Ltd. : [сайт]. – URL: https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC2/HTML/schema/ifcmeasureresource/lexical/ifcdimensionalexponents.htm (дата обращения: 08.03.2022).

ПРИЛОЖЕНИЕ А
Скрипты для создания таблиц в базе данных

```
CREATE TABLE IF NOT EXISTS BasicDimensions (
      id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
      name TEXT NOT NULL,
UNIQUE (name))

CREATE TABLE IF NOT EXISTS Quantities (
      id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
      name TEXT NOT NULL,
UNIQUE (name))

CREATE TABLE IF NOT EXISTS Dimensions (
      quantityId INTEGER NOT NULL REFERENCES Quantities(id),
      basicDimensionId INTEGER NOT NULL REFERENCES BasicDimensions(id),
      exponent INTEGER NOT NULL,
UNIQUE (quantityId, basicDimensionId))

CREATE TABLE IF NOT EXISTS Units (
      id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
      name TEXT NOT NULL,
      symbol TEXT NOT NULL,
      quantityId INTEGER NOT NULL REFERENCES Quantities(id),
      ratio REAL NOT NULL,
UNIQUE (name),
UNIQUE (symbol))

CREATE TABLE IF NOT EXISTS Constants (
      id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
      name TEXT NOT NULL,
      symbol TEXT NOT NULL,
      quantityId INTEGER NOT NULL REFERENCES Quantities(id),
      realPart REAL NOT NULL,
      imaginaryPart REAL NOT NULL,
UNIQUE (name),
UNIQUE (symbol))
```

ПРИЛОЖЕНИЕ Б
Программная реализация контроля корректности введенного пользователем задания на вычисление

```
/* Проверка синтаксиса */

QString checkSyntax(QVector<QString> tokens)
{
    // Замена токенов на их тип:
    for (auto &token : tokens)
    {
        // Числовой операнд;
        if (token.front().isDigit())
        {
            bool ok;
            QLocale().toDouble(token, &ok);
            if (!ok)
                return "Неверное число";

            token = "OPERAND";
        }
        // Токены, начинающиеся с букв:
        else if (token.front().isLetter())
        {
            // Комплексный операнд;
            if (token == "i")
                token = "OPERAND";
            // Функция;
            else if (token=="sin" || token=="cos" || token=="tan" || token=="tg" || token=="ln")
                token = "FUNCTION";
            // Идентификатор постоянной или единицы измерения;
            else
                token = "ID";
        }
        // Токены-символы:
        else if (token == "!")
            token = "POSTFIX";
        else if (token == "√")
            token = "FUNCTION"
        else if (token == "+" || token == "-" || token == "*" || token == "/" || token == "^")
            token = "BINARY";
        else if (token == "+U" || token == "-U")
            token = "UNARY";
        else if (token == "[" || token == "{")
            token = "IDBEGIN";
        else if (token == "]" || token == "}")
            token = "IDEND";
        else if (token == "(")
            token = "LEFTBR";
        else if (token == ")")
            token = "RIGHTBR";
        else
            return "Неизвестный символ";
    }

    // Карта ожидаемых токенов;
    QMap<QString, QVector<QString>> expectedTokens {
        { "BEGIN",    { "FUNCTION", "OPERAND", "LEFTBR", "UNARY", "IDBEGIN" } },
        { "FUNCTION", { "LEFTBR" } },
        { "OPERAND",  { "BINARY", "POSTFIX", "IDBEGIN", "END", "RIGHTBR" } },
        { "BINARY",   { "BEGIN", "FUNCTION", "OPERAND", "LEFTBR", "UNARY", "IDBEGIN" } },
        { "UNARY",    { "BEGIN", "FUNCTION", "OPERAND", "LEFTBR", "IDBEGIN" } },
        { "POSTFIX",  { "BINARY", "POSTFIX", "IDBEGIN", "END", "RIGHTBR" } },
        { "IDBEGIN",  { "ID" } },
        { "ID",       { "IDEND" } },
        { "IDEND",    { "BINARY", "POSTFIX", "END", "RIGHTBR" } },
        { "LEFTBR" ,  { "FUNCTION", "OPERAND", "LEFTBR", "UNARY", "IDBEGIN" } },
        { "RIGHTBR" , { "BINARY", "POSTFIX", "IDBEGIN", "END", "RIGHTBR" } }
    };
```

```
    // Переменная, контролирующая расстановку скобок;
    int bracketdiff = 0;

    // Добавление конечного токена в конец списка токенов;
    tokens.append(«END»);

    // Задание начальных ожидаемых токенов;
    Qvector<Qstring> expected = expectedTokens[“BEGIN”];

    for (const auto &token : tokens)
    {
        if (token == “LEFTBR”)
            bracketdiff++;
        else if (token == “RIGHTBR”)
            bracketdiff--;

        if (bracketdiff < 0)
            return «Неправильная расстановка скобок»;

        if (!expected.contains(token))
            return «Неправильная расстановка операторов»;

        expected = expectedTokens[token];
    }
    if (bracketdiff)
        return «Несоответствие количества скобок»;

    return 0;
}

/* Проверка логики (внутри класса с операндами) */

class Operand
{
    // …

    // Оператор сложения;
    Operand &operator+=(const Operand &other)
    {
        // Если величины не совпадают:
        if (unit.getQuantity() != other.unit.getQuantity())
            throw std::runtime_error(«Сложение несовместимых величин»);

        value += other.value * other.unit.getRatio();

        return *this;
    }

    // Оператор вычитания;
    Operand &operator-=(const Operand &other)
    {
        // Если величины не совпадают:
        if (unit.getQuantity() != other.unit.getQuantity())
            throw std::runtime_error(«Вычитание несовместимых величин»);

        value -= other.value * other.unit.getRatio();

        return *this;
    }

    // Оператор умножения;
    Operand &operator*=(const Operand &other)
    {
        // Если величины совпадают, а единицы измерения не совпадают:
        if (unit.getRatio() != other.unit.getRatio() && unit.getQuantity() ==
other.unit.getQuantity())
            throw std::runtime_error(«Умножение равных величин в разных единицах»);

        unit = Unit(unit.getQuantity() * other.unit.getQuantity(), 1.0);
        value *= other.value * other.unit.getRatio();

        return *this;
    }
```

```
    // Оператор деления;
    Operand &operator/=(const Operand &other)
    {
        // Если величины совпадают, а единицы измерения не совпадают:
        if (unit.getRatio() != other.unit.getRatio() && unit.getQuantity() ==
other.unit.getQuantity())
            throw std::runtime_error("Умножение равных величин в разных единицах");

        unit = Unit(unit.getQuantity() / other.unit.getQuantity(), 1.0);
        value /= other.value * other.unit.getRatio();

        return *this;
    }

    // Оператор возведения в степень;
    Operand &operator^=(const Operand &other)
    {
        // Если показатель является числом с единицей измерения
        if (unit.getQuantity() != Quantity() && other.unit.getQuantity() != Quantity())
            throw std::runtime_error("Показатель степени с единицей измерения");

        unit = Unit(unit.getQuantity() ^ other.value, 1.0);
        value = std::pow(value, other.value);

        return *this;
    }

    // ...
};
```