

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт искусственного интеллекта

Кафедра проблем управления

**КУРСОВАЯ РАБОТА**

по дисциплине **Основы программирования систем управления**

**Тема курсовой работы:** «Поиск файлов на жестких дисках и портативных устройствах»

**Студент группы КРБО-03:** Мошков С. А. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Руководитель курсовой работы:** ст.преп.Морозов А. А. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Работа представлена к защите: «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Допущен к защите«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Москва 2024



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт искусственного интеллекта

Кафедра проблем управления

**ЗАДАНИЕ**

**на выполнение курсовой работы** по дисциплине

«Основы программирования систем управления»

|  |
| --- |
| Утверждаю  Заведующий кафедрой ПУ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Романов М.П.  Подпись  «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

Студент: Мошков С. А. Группа: КРБО-03-23

**Тема: «**Разработка калькулятора»

**Исходные данные:**

1. Язык программирования Си/Си++

2. Калькулятор.

3. Ввод значений в виде строки

**Перечень вопросов, подлежащих обработке, и обязательного графического материала:**

1. Разбор строки символов.

2. Приоритет операций.

3. Блок-схема алгоритма программы

**Срок представления к защите курсовой работы: до** «\_\_\_\_» \_\_\_\_\_\_\_ 2024 г.

**Задание на курсовую работу выдал** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Морозов А.А**.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**Задание на курсовую работу получила** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Мошков С. А.**

Содержание

[Аннотация 4](#_Toc167642457)

[Введение 5](#_Toc167642458)

[Методология 7](#_Toc167642459)

[Структура программы 10](#_Toc167642460)

[Демонстрация возможностей программы на тестовом примере 12](#_Toc167642461)

[Заключение 14](#_Toc167642462)

[Источники 15](#_Toc167642463)

# Аннотация

В данном отчете представлен процесс разработки и реализации программного калькулятора, способного выполнять сложные математические вычисления с использованием стеков. Основное внимание уделено архитектуре программы, методам обработки математических выражений и точности вычислений.

Калькулятор разработан на языке программирования C++ и включает в себя поддержку базовых арифметических операций, а также расширенных функций, таких как тригонометрия, логарифмы, экспоненты и возведение в степень. Программа использует структуру данных стек для хранения операндов и операторов, что позволяет точно и последовательно обрабатывать выражения в соответствии с правилами математики.

Отчет подробно описывает каждый компонент программы, включая функции округления, тригонометрические вычисления и обработку ошибок. Также приводятся примеры кода, демонстрирующие реализацию ключевых функций калькулятора.

# 1 Введение

В современном мире, где данные и информация играют ключевую роль, возможность быстро и точно обрабатывать числовые данные становится все более важной. Математические калькуляторы, которые мы используем каждый день, от простых устройств до сложных компьютерных программ, являются неотъемлемой частью нашей жизни. Они помогают нам в решении задач, от базовых арифметических операций до сложных инженерных расчетов.

Целью данной курсовой работы является разработка стекового калькулятора на языке программирования C++, который способен обрабатывать сложные математические выражения, включая тригонометрические функции, возведение в степень, логарифмы и другие операции. Особенностью данного калькулятора является использование структуры данных "стек" для хранения чисел и операций, что позволяет учитывать приоритет операций и корректно обрабатывать скобки в выражениях.

* 1. **Краткий обзор функциональности калькулятора**

Разработанный стековый калькулятор на C++ представляет собой мощный инструмент, способный выполнять широкий спектр математических операций. Основная функциональность калькулятора включает в себя:

* **Арифметические операции:** Калькулятор поддерживает все основные арифметические операции, такие как сложение ++, вычитание −−, умножение ××, деление ÷÷, и возведение в степень ^.
* **Тригонометрические функции:** Пользователи могут вычислять значения синуса, косинуса, тангенса, а также их обратные функции.
* **Логарифмы и экспоненты:** Калькулятор обеспечивает вычисление натуральных логарифмов, а также экспоненциальных функций.
* **Обработка скобок:** Стековая структура позволяет корректно обрабатывать выражения со скобками, учитывая приоритет операций.
* **Обработка ошибок:** Калькулятор включает систему обработки ошибок, предотвращающую некорректный ввод и вычисления, такие как деление на ноль или неверный ввод тригонометрических функций.

Этот калькулятор не только обеспечивает точность вычислений, но и предлагает строковой интерфейс, который делает процесс ввода выражений простым и интуитивно понятным.

**1 .2 Значение корректной обработки выражений в вычислительной технике**

Корректная обработка математических выражений является одним из фундаментальных аспектов вычислительной техники. Это не просто вопрос точности, но и вопрос эффективности, безопасности и надежности программного обеспечения.

В контексте разработки стекового калькулятора корректная обработка выражений позволяет не только гарантировать точность результатов, но и обеспечивает уверенность в том, что калькулятор будет работать надежно и эффективно в различных условиях и сценариях использования. Это особенно важно при решении сложных математических задач, где каждый шаг вычисления должен быть выполнен безошибочно. Таким образом, корректная обработка выражений является ключевым элементом, который определяет успех и полезность вычислительных инструментов в целом.

1. **Методология**

**2.1 Объяснение структуры данных стек**

Стек — это одна из фундаментальных структур данных в информатике, которая используется для хранения коллекции элементов. Основная особенность стека заключается в порядке доступа к его элементам, который следует принципу "последним пришёл — первым ушёл" (LIFO - Last In, First Out). Давайте подробнее рассмотрим ключевые аспекты стека:

* **Операции со стеком**: Основные операции, которые можно выполнять со стеком, это push (добавление элемента в стек) и pop (удаление последнего добавленного элемента из стека). Также часто используется операция peek или top, которая позволяет посмотреть на верхний элемент стека без его удаления.
* **Применение стека:** Стеки широко используются в программировании для реализации функций, обработки рекурсивных вызовов, синтаксического анализа выражений и во многих других задачах, где необходимо обратиться к последнему добавленному элементу.
* **Пример стека:**

Дно стека -> [Элемент1, Элемент2, Элемент3] <- Вершина стека

Если мы выполним операцию название\_стека.push(Элемент4), стек изменится следующим образом:

Дно стека -> [Элемент1, Элемент2, Элемент3, Элемент4] <- Вершина стека

После операции название\_стека.pop(), стек вернётся к предыдущему состоянию:

Дно стека -> [Элемент1, Элемент2, Элемент3] <- Вершина стека

* **Преимущества и недостатки:** Стеки обладают простотой реализации и эффективностью выполнения операций. Однако они ограничены в функциональности по сравнению с другими структурами данных, такими как очереди или деревья, и не подходят для задач, требующих доступа к произвольным элементам коллекции.

Стеки играют важную роль в вычислительной технике, обеспечивая удобный и эффективный способ управления данными в ситуациях, когда последний добавленный элемент должен быть обработан первым.

**2.2 Обсуждение Реализованных Функций**

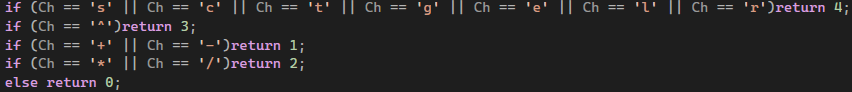
В ходе разработки калькулятора был реализован ряд функций для выполнения математических операций. Основное внимание было уделено обеспечению точности вычислений и соответствия математическим стандартам.

Калькулятор поддерживает все базовые арифметические операции: сложение, вычитание, умножение и деление. Эти операции являются фундаментальными и были реализованы с использованием стандартных операторов языка C++.

Для расширения функциональности калькулятора были добавлены дополнительные математические операции, такие как возведение в степень, извлечение квадратного корня, вычисление тригонометрических функций и нахождение натурального логарифма. Для этого была подключена библиотека <cmath> и создана функция Math, которая берет символ из стека с операциями и проверяет его на принадлежность к тому или иному типу:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Операция в строке | + | - | \* | / | sin | cos | exp | tg | ln | sqrt |
| Название операции в стеке | + | - | \* | / | s | c | e | t | l | r |

Если операция не принадлежит ни к одному из типов, то выводится сообщение с ошибкой и программа прекращает работать.

 Также одной из ключевых особенностей калькулятора является поддержка приоритета операций и использование скобок для изменения порядка вычислений. Для этого была создана функция getPriority(от англ. get – получит, priority – приоритет). Для каждой операции был установлен приоритет, как показано на изображении 1.

Изображение 1 – Приоритеты операций

Для корректной работы калькулятора была добавлена система обработки ошибок, которая предупреждает пользователя о невозможности выполнения операции, например, при делении на ноль или вводе отрицательного под корень. Это повышает надежность и удобство использования программы.

Также для избежания ошибок, связанных с неточностью встроенных функций sin и cos, были созданы функции Sin и Cos.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

**2.3 Описание Структуры 'Leksema'**

Структура 'Leksema', являющаяся ключевым элементом данного калькулятора, представляет собой структуру данных, которая используется для хранения и обработки математических выражений, введенных пользователем.

'Leksema' определяется как структура, содержащая информацию о типе элемента (операнд, оператор, скобка) – ‘Stack\_o’ и его значении – ‘Stack\_n’. Это позволяет программе корректно интерпретировать каждый элемент входного выражения и обеспечивать правильную последовательность вычислений.

В процессе анализа математического выражения каждый символ преобразуется в 'Leksema', что позволяет эффективно управлять приоритетами операций и скобками.

После преобразования входного выражения в последовательность 'Leksem', калькулятор использует стек для вычисления результата. 'Leksema' извлекается из стека в порядке, определенном правилами математического приоритета, что гарантирует точность вычислений.

Использование структуры 'Leksema' обеспечивает несколько преимуществ:

* Гибкость: 'Leksema' позволяет легко добавлять новые операции и функции в калькулятор.
* Эффективность: Структура данных оптимизирует процесс вычисления, минимизируя количество необходимых операций.

1. **Структура программы**

**3.1 Описание работы программы**

1. Инициализация констант и функций:
   * const double Pi = acos(-1); - определяет значение числа Пи через арккосинус от -1.
   * double Sin(double x) и double Cos(double x) - функции для вычисления синуса и косинуса угла x, заданного в градусах, с округлением до 7 знаков после запятой.
2. Структура Leksema:
   * Описывает либо число (type равен '0'), либо операцию (например, '+' для сложения). Для операций value всегда равно 0.
3. Функция Maths:
   * Принимает два стека: Stack\_n для чисел и Stack\_o для операций, а также элемент item.
   * Выполняет математические операции в зависимости от типа операции на вершине стека Stack\_o.
4. Функция getPriority:
   * Возвращает приоритет операции для определения порядка выполнения в выражении.
5. Функция main:
   * Запрашивает у пользователя ввод выражения.
   * Использует stringstream для чтения и обработки введенного выражения.
   * Использует стеки Stack\_n и Stack\_o для хранения чисел и операций соответственно.
   * Обрабатывает введенное выражение, распознавая числа, операции и функции.
   * Выполняет вычисления с помощью функции Maths до тех пор, пока не будет достигнут конец выражения или не будут обработаны все операции.
   * Выводит результат вычисления выражения.

Код реализует алгоритм обратной польской записи для вычисления выражений, используя стеки для управления порядком операций и чисел. Ошибки в вычислениях обрабатываются через вывод сообщений в cerr и возвращение false из функции Maths, что приводит к завершению программы. Обработка специфических функций, таких как логарифмы и тригонометрические функции, также поддерживаются данным кодом.

**3.2 Описание алгоритма разбора и вычисления выражений**

Алгоритм разбора и вычисления математических выражений — это процесс, который позволяет преобразовать строку, содержащую математическое выражение, в значение, которое можно вычислить и использовать в дальнейших расчётах. Этот процесс включает в себя несколько ключевых этапов:

1. Лексический анализ (Токенизация):  
   На этом этапе входная строка разбивается на последовательность токенов. Токены представляют собой числа, операторы, скобки и функции. Например, выражение 3+4∗23+4∗2 будет образно говоря разбито на токены: [3, '+', 4, '\*', 2].
2. Синтаксический анализ (Парсинг):  
   Следующий шаг — это преобразование последовательности токенов в структуру, которая отражает иерархию операций.
3. Преобразование в постфиксную форму (Обратная польская запись):  
   Для упрощения вычислений инфиксное выражение может быть преобразовано в постфиксную форму. Это делается с помощью стека, где операторы добавляются в стек в зависимости от их приоритета, а затем извлекаются для формирования постфиксного выражения.
4. Вычисление выражения:  
   После преобразования в постфиксную форму, выражение вычисляется с помощью стека. Когда встречается число, оно помещается в стек. Когда встречается оператор, из стека извлекаются два последних числа, с ними выполняется операция, и результат помещается обратно в стек.
5. Обработка ошибок:  
   Важной частью алгоритма является обработка возможных ошибок, таких как некорректные символы, неправильное количество операндов для оператора, деление на ноль и т.д.

**3.3 Блок-схема кода**

# Изображение выглядит как диаграмма, текст, зарисовка, План Автоматически созданное описание4 Демонстрация возможностей программы на тестовом примере

sin(45)^2 + cos(45)^2 + ((ln(1) + sqrt(4)) \* tg(45)) \* 20 + 555/5 – данное выражение содержит все требуемые операции, поэтому оно подходит для проверки работоспособности данной программы.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Данный калькулятором ответ соответствует реальности:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

* 1. **Заключение**

В результате выполнения курсовой работы был разработан калькулятор, который поддерживает такие функции, как сложение, вычитание, умножение, деление, возведение в степень, извлечение корня, нахождения синуса, косинуса и тангенса угла, вычисление натурального логарифма. Для упрощения вычисления сложных математических выражений была использована технология стеков.

* 1. **Источники**

<https://devdocs.io/cpp/> - документация к языку С++

<https://github.com/Berendei-Jr/Calculator> - социальная сеть для разработчиков