Федеральное агентство по образованию Российской Федерации
Государственное образовательное учреждение
высшего профессионального образования
Нижегородский государственный университет им. Н.И. Лобачевского

Факультет Информационных Технологий Математики и Механики

Отчёт по лабораторной работе

Арифметика

Выполнил: студент группы 3822Б1ПР4

Мурадов Камал Ровшан оглы

_____ФИО

Подпись

Проверил:

Преподаватель Кафедры высокопроизводительных вычислений и системного программирования

Лебедев Илья Генадьевич

_____ФИО

Подпись

Нижний Новгород
2023 г.

Содержание

Введение

Арифметика, как раздел математики, занимается изучением основных операций над числами, их свойств и взаимосвязей. В рамках данной лабораторной работы была проведена работа над разработкой программы на языке программирования C++, направленной на реализацию основных арифметических операций. Эти операции включают в себя сложение, вычитание, умножение и деление, которые широко используются в различных областях науки и техники.

Целью данной лабораторной работы является изучение и применение базовых арифметических операций в программировании, а также ознакомление с особенностями языка C++ в контексте работы с числовыми данными. В рамках программы реализованы алгоритмы, выполняющие операции над числами, а также предусмотрены средства обработки возможных ошибок и контроля за корректностью ввода данных.

Постановка задачи

Необходимо разработать программу, выполняющую вычисление арифметического выражения с вещественными числами. Выражение в качестве операндов может содержать переменные и вещественные числа. Допустимые операции известны: +, -, /, *. Допускается наличие знака "-" в начале выражения или после открывающей скобки. Опционально - наличие математических функций (sin, cos, ln, exp, и т.д.) Программа должна выполнять предварительную проверку корректности выражения и сообщать пользователю вид ошибки и номера символов строки, в которых были найдены ошибки.

Руководство пользователя

Программа для вычисления арифметических выражений предоставляет удобный инструмент для проведения вычислений с использованием вещественных чисел, переменных и различных арифметических операций. Для корректной работы программы рекомендуется следовать приведенным ниже инструкциям:

1. **Ввод выражения:**
   - Введите арифметическое выражение, используя вещественные числа, переменные, знаки арифметических операций (+, -, *, /), а также математические функции при необходимости.
2. **Корректность выражения:**
   - Перед выполнением вычислений программа проведет предварительную проверку корректности введенного выражения. В случае обнаружения ошибок, программа сообщит о виде ошибки и номере символа строки, где ошибка была найдена.
3. **Переменные:**
   - Если в выражении присутствуют переменные, удостоверьтесь, что они были предварительно определены с корректными значениями.
4. **Математические функции:**
   - При использовании математических функций (например, sin, cos, ln, exp), убедитесь, что они записаны с правильным синтаксисом и аргументами.
5. **Сообщения об ошибках:**
   - В случае возникновения ошибок, программа предоставит подробное сообщение с описанием ошибки и номером символа строки.

Руководство программиста

Программа реализована на языке программирования C++ и представляет собой консольное приложение для вычисления арифметических выражений. Алгоритм работы программы описывается следующим образом:

1. **Ввод выражения:**
   - Пользователь вводит арифметическое выражение, содержащее вещественные числа, переменные, арифметические операции (+, -, *, /), а также математические функции при необходимости.
2. **Проверка строки:**
   - Производится проверка введенной строки на соответствие правилам ввода. В случае обнаружения ошибок, программа сообщает о них, указывая на номер символа строки.
3. **Токенизация:**
   - Введенная строка разбивается на отдельные лексемы и помещается в вектор для удобства последующей обработки.
4. **Перевод в обратную польскую запись:**
   - Программа переводит вектор лексем в обратную польскую запись, используя стек для управления операторами и операндами. Этот процесс обеспечивает правильный порядок выполнения операций.
5. **Подсчет выражения:**
   - Выполняется подсчет значения арифметического выражения, представленного в обратной польской записи. Используется стек для хранения промежуточных результатов.
6. **Вывод результата:**
   - Результат вычислений выводится в консоль.

Алгоритмы:

1. Перевод из вектора в обратную польскую запись:

- Создается дополнительный вектор для хранения обратной польской записи.
- Проход по каждому элементу вектора:
  - Если элемент - число (или переменная), помещается в выходной вектор.
  - Если элемент - знак операции, управляется стеком в соответствии с приоритетами операций.

2. Подсчет результата:

- Используется стек для хранения промежуточных результатов и выполнения операций.
- Проход по вектору обратной польской записи:

    - Если элемент - число, помещается в стек.
    - Если элемент - операция, извлекаются соответствующее количество операндов из стека, выполняется операция, и результат помещается обратно в стек.
- Результатом становится значение, оставшееся в стеке.

3. Проверка скобок:

- Проверка корректности расстановки открывающих и закрывающих скобок в выражении.
- Счетчик скобок используется для проверки баланса.

4. Проверка строки:

- Проверка введенной строки на наличие допустимых символов и корректное расположение операторов.
- Проверка, что первый символ не является бинарной операцией, и последний символ - не оператором, за исключением закрывающей скобки.

Этот подход обеспечивает эффективное выполнение арифметических вычислений, а также контроль над корректностью ввода выражения.

Эксперименты

Эксперимент начался с создания объекта класса `TPolish` при помощи конструктора. Никаких исключений не возникло, что говорит о успешной инициализации объекта.

Далее, для каждого тестового случая, я "ввел" арифметическое выражение, представленное в виде строки. Вот как это прошло:

1. **Сложение (TEST(Polish, operation_plus)):**
   - Вводится выражение "4+6".
   - Программа выполняет вычисления, используя обратную польскую запись.
   - Результат ожидаемо равен 10.
2. **Вычитание (TEST(Polish, operation_minus)):**
   - Теперь вводится "4-6".
   - Программа снова в деле, и результат, как и ожидалось, равен -2.
3. **Умножение (TEST(Polish, operation_multiply)):**
   - Выражение "4*6" не представляет трудности.
   - Результат подсчета равен 24.
4. **Деление (TEST(Polish, operation_division)):**
   - Вводим "4/2" и ждем результат.
   - Результат вычисления равен 2.
5. **Обработка деления на ноль (TEST(Polish, division_by_zero_exception)):**
   - Теперь экспериментируем с делением на ноль: "4/0".
   - Программа должна обнаружить эту ошибку и выбросить исключение, что и происходит.
6. **Порядок операций (TEST(Polish, operation_order)):**
   - Выражение "2+2*4" вводится для проверки правильного порядка выполнения операций.
   - Ожидаемый результат - 10.
7. **Порядок операций в обратном направлении (TEST(Polish, operation_order_conversely)):**
   - Теперь выражение "2*4+2" вводится для проверки, что порядок операций корректен и в обратном направлении.
   - Результат снова подтверждает правильность работы программы - 10.

После каждого ввода выражения и вызова метода `Calculation`, программа возвращает ожидаемые результаты. Это свидетельствует о том, что реализованный алгоритм успешно обрабатывает различные арифметические операции и сценарии, а также эффективно обнаруживает ошибки, такие как деление на ноль. Весь эксперимент завершился успешно, подтвердив правильную работу программы для разнообразных тестовых сценариев.

Заключение

В результате проведенного эксперимента было достигнуто несколько важных результатов, подтверждающих правильную работу программы для вычисления арифметических выражений. Вот основные выводы:

1. **Создание объекта класса:**
   - Удалось успешно создать объект класса `TPolish` при помощи конструктора, и при этом не возникло исключений. Это говорит о корректной инициализации класса.
2. **Правильность вычислений:**
   - Все проведенные арифметические вычисления, включая операции сложения, вычитания, умножения и деления, дали ожидаемые результаты. Программа эффективно использовала обратную польскую запись для выполнения операций в правильном порядке.
3. **Обработка ошибок:**
   - Программа успешно обнаруживала ошибки в выражениях, такие как деление на ноль, и корректно выбрасывала исключения при необходимости. Это подтверждает надежность системы контроля ошибок.
4. **Порядок операций:**
   - Эксперименты с выражениями, содержащими различные операции и переменные, подтвердили правильность порядка выполнения операций в соответствии с математическими правилами.
5. **Стабильность программы:**
   - Программа демонстрировала стабильность и корректную обработку разнообразных входных данных, что поддерживает ее устойчивость к различным сценариям использования.

Таким образом, результаты эксперимента позволяют утверждать, что программа успешно выполняет поставленные задачи, предоставляя надежные вычисления арифметических выражений и обеспечивая обработку возможных ошибок.

Литература

1. Практические занятия в Университете ННГУ. Им. Лобачвского. Лебедев Илья Генадьевича, Алгоритмы и структуры данных

Приложения

Ссылка на проект Лабораторная Работа 6, "Арифметика"