**C++ Calculator**

User’s Manual

Version 1.0

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 12/1/23 | 1.0 | Creation and Declaration of User Manual | Nabeel Ahmad, Shero Baig, Arnav Jain, Zonaid Prithu, Omar Mohammed, Yaeesh Mukadam, Humza Qureshi |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Purpose 4

2. Introduction 4

3. Getting started 4

4. Advanced features 4

5. Troubleshooting 4

6. Example of uses 4

7. Glossary 5

8. FAQ 5

**Test Case**

# Purpose

Welcome to Calculon!

# Introduction

**Overview**

An easy-to-use, lightweight, and effective command-line tool for arithmetic operations is called Calculon. With its robust and adaptable design, it works with the majority of operating systems and is a useful tool for making quick calculations as well as an instructional guide for learning about C++ programming and CLI application development. To install Calculon simply download the given folder, unzip it, and then run calculon. Then, open your command line interface and type “calculon” to launch Calculon. This will start the calculator so that input may be entered.

**Installation and Setup**

Download the folder given, unzip it, and then run it.

# Getting started

**User Input Assumptions**

It is expected of users to know the fundamentals of mathematical expressions. Users must submit mathematically sound expressions onto the calculator.

# Advanced features

**RPN Conversion**For effective evaluation, Calculon internally translates infix expressions to Reverse Polish Notation (RPN).

**Error Handling**With the help of unambiguous error signals, the system effectively addresses mistakes like division by zero, improper syntax, and mismatched parenthesis.

**Parsing and Evaluation**Calculon handles parentheses in parsing arithmetic expressions and makes sure the evaluation is done in the right sequence.

**Future Enhancements**Future versions may include additional operators or support for floating-point numbers.

**Numeric Constants**The calculator can currently perform calculations using integers, but future updates will add support for floating-point values.

**CLI Interaction and Input Validation**Users are prompted to enter common arithmetic expressions using the simple CLI, which accepts operators, parentheses, and integers. For ease, spaces in inputs are disregarded. To reduce security threats, the interface provides strong error handling and input validation.

**Screen Format and Response**Calculon guides the user back to the input prompt in the event of an error after displaying the computed result or a detailed error message upon submission of the expression.

**Architectural Goals and Constraints**

**Functionality and Reliability**The architecture prioritizes efficiency and component reusability to achieve accurate and dependable expression evaluation.

**Security and Safety**The architecture incorporates security features to prevent arbitrary code execution through expression inputs.

**Extensibility and Quality Assurance**

**Modular Design for Future Enhancements**Future changes, such as additional operators or support for floating-point numbers, can be added to the code with minimum impact thanks to the modular design. ​​.

**Performance Optimization**Even for complex expressions, optimal performance is ensured via efficient data structures like stacks.

# Troubleshooting

**Common Issues and Solutions**

**Invalid Syntax**: Ensure the expression follows the correct arithmetic format.

**Runtime Errors**: Check for divisions by zero and correct the expression.

# Examples

**Sample Calculations**

* **3 + 5 \* 2**: Calculates to **13**.
* **(3 + 5) \* 2**: Calculates to **16**.
* **3 / 0**: Returns an error message.

# Glossary of terms

# Arithmetic Expression: A mathematical notation involving numbers and operation symbols that denote a single mathematical operation.

# 

# CLI (Command-Line Interface): A user interface where users interact with software by typing commands in a text-based terminal.

# 

# Evaluator Module: The component in Calculon responsible for processing the tokenized input according to operator precedence and parenthesis handling.

# 

# Expression Parsing Process: The sequence of actions from accepting user input to generating a tokenized representation of the expression.

# 

# Expression Tree: A data structure used to represent parsed expressions hierarchically, with nodes for operators and leaves for operands.

# 

# Infix Notation: The conventional notation for writing arithmetic expressions where operators are placed between operands (e.g., 3 + 4).

# 

# Modular Design: An architectural approach that divides a system into separate modules, each with a distinct functionality.

# 

# Operator Precedence (PEMDAS): The rule dictating the order in which operations in arithmetic expressions are carried out, represented by the acronym PEMDAS (Parentheses, Exponents, Multiplication and Division, Addition and Subtraction).

# 

# Parser Module: The component responsible for tokenizing the input expression and checking for syntactical correctness.

# 

# RPN (Reverse Polish Notation): A mathematical notation wherein operators follow their operands, used internally by Calculon for efficient expression evaluation.

# 

# Stack-Based Evaluation: A method of evaluating expressions where operators and operands are pushed onto and popped from a stack to compute the result.

# 

# Tokenization: The process of converting an input string into discrete units called tokens for easier parsing and processing.

# 

# Unary Operations: Operations that involve only one operand, such as negation.

# 

# Utility Package: A set of auxiliary functionalities used across Calculon's subsystems, providing common operations like string manipulation and numerical conversions.

# FAQ

**Q: How does Calculon handle operator precedence in complex expressions?**

A: Calculon adheres to the PEMDAS rule, ensuring operations like multiplication and division are prioritized over addition and subtraction. Parentheses are also respected, allowing users to define their own precedence in expressions.

**Q: Can I use Calculon for educational purposes to learn C++ and arithmetic expressions?**

A: Yes, Calculon is an excellent tool for educational purposes. It not only helps in understanding arithmetic expressions but also demonstrates good practices in C++ programming and CLI application development.

**Q: Is it possible to extend or modify Calculon for personal use or academic projects?**

A: Calculon's modular design allows for potential extensions or modifications. However, it's important to adhere to the software's license terms and conditions when making any changes.

**Q: Does Calculon support floating-point arithmetic currently?**

A: As of now, Calculon supports integer-based calculations. Support for floating-point arithmetic is considered for future enhancements.

**Q: How does Calculon ensure the security and safety of its operations?**

A: Calculon's architecture strictly validates input expressions to prevent arbitrary code execution, thereby mitigating potential security risks. It's designed to handle standard arithmetic expressions safely.

**Q: What kind of errors can Calculon detect and how does it respond to them?**

A: Calculon is equipped to detect errors like invalid syntax, division by zero, and unmatched parentheses. In such cases, it provides descriptive error messages and guides the user to correct the input.

**Q: Can Calculon handle expressions with nested parentheses?**

A: Yes, Calculon can handle expressions with multiple levels of nested parentheses, correctly interpreting the order of operations as per standard mathematical rules.

**Q: Is there a limit to the complexity or length of expressions that Calculon can handle?**

A: While Calculon is designed to handle standard arithmetic expressions efficiently, extremely long or complex expressions might impact performance due to inherent memory and computational constraints.

**Q: How does the RPN conversion enhance Calculon's performance?**

A: The RPN (Reverse Polish Notation) conversion streamlines the evaluation process by eliminating the need for repeated parsing and reducing the computational complexity, thus enhancing performance, especially for complex expressions.

**Q: Are there any plans to introduce a graphical user interface for Calculon?**

A: Currently, Calculon is solely a CLI-based tool. A graphical user interface (GUI) is outside the project's scope for now, but this could be considered for future development based on user feedback and project evolution**.**