Postfix Expression Evaluator – Project Report

Overview

The **Postfix Expression Evaluator** is a Windows GUI-based application written in C that allows users to input mathematical expressions in infix notation, automatically converts them to postfix (Reverse Polish Notation), and evaluates them. It provides real-time error feedback using dialog boxes and logs a step-by-step stack trace to help users understand the evaluation process. This application is designed with an intuitive user interface and robust backend logic to enhance understanding of expression evaluation and stack-based computation.

Core Features

Infix to Postfix Conversion:

- Supports standard arithmetic operators: +, -, *, /, and ^.
- Handles parentheses and operator precedence.
- Right-associative operator support (e.g., ^ for exponentiation).

Postfix Evaluation:

- Utilizes a stack to evaluate postfix expressions.
- Logs each operation and stack state to a file for traceability.
- Supports integer calculations, including exponentiation via math.h.

GUI Interface:

- Input field for entering expressions.
- "Evaluate" button for processing input.
- Output area displaying the conversion and evaluation steps.

Error Handling:

- Dialog-based (MessageBox) error notifications for:
  - Invalid characters
  - Parentheses mismatch
  - Division by zero
  - Stack overflow/underflow

Logging:

- Stack states and final result are saved and displayed from a temporary file.

Workflow Overview

The application follows a structured sequence of steps to process and evaluate infix arithmetic expressions entered by the user. Let's walk through the complete workflow using the sample expression: **2 + 3^5 - 4 / 5 * 7**

User Input

The user types an infix expression **(e.g., 2 + 3^5 - 4 / 5 * 7)** into the input text box of the GUI.

Infix to Postfix Conversion

The application converts the infix expression to postfix (Reverse Polish Notation) using a stack-based algorithm, respecting operator precedence and associativity.

Converted Postfix Expression: 2 3 5 ^ + 4 5 / 7 * -

Postfix Evaluation with Stack Logging

Each token in the postfix expression is evaluated in order:

- Operands (numbers) are pushed onto a stack.
- Operators pop the necessary number of operands, apply the operation, and push the result.
- The process is logged at each step in a temporary file for transparency and debugging.

Example Stack Trace:

- After processing number '2': Stack: 2
- After processing number '3': Stack: 2 3
- After processing number '5': Stack: 2 3 5

- After processing '^': Stack: 2 243
- After processing '+': Stack: 245
- After processing number '4': Stack: 245 4
- After processing number '5': Stack: 245 4 5
- After processing '/': Stack: 245 0
- After processing number '7': Stack: 245 0 7
- After processing '*': Stack: 245 0
- After processing '-': Stack: 245
- Final Result: 245

Output Display

After evaluation:

- The contents of the log file (infix, postfix, stack trace, and result) are read.
- This output is displayed in the multiline output window of the GUI for user reference.

This process ensures the application handles operator precedence (including exponentiation), parenthesis validation, and stack-based evaluation with full transparency and user-friendly error reporting.

Usage Requirements

Operating System: Windows (Windows 7 or later recommended)
Compiler: Any C compiler that supports the Win32 API (e.g., MinGW, MSVC)
Build Output: Standalone .exe file (e.g., PostfixEv.exe)
No external dependencies are required. The executable runs as a standalone GUI application.

Conclusion

This project demonstrates the implementation of an arithmetic expression evaluator using both stack-based algorithms and the Windows GUI framework in C. The application serves as both a functional tool and an educational aid, showing how infix expressions are parsed, converted, and evaluated. With detailed logging and error handling, it ensures robustness and user friendliness. Future enhancements could include variable support, floating-point precision, or a more advanced parser with syntax tree visualization.