Week 3 Mini Project Project Documentation

Jude Eschete

February 17, 2025

1 Introduction

SymbolBalance is a C++ console application designed to check for balanced symbols in a user-entered expression and convert valid infix expressions into postfix form. It supports parentheses `()`, braces {}, brackets `[]`, and block comments `/* */`. The code is organized so that new C++ learners can easily understand the structure and the key algorithms involved.

2 Design and Pseudocode

The primary tasks of this project are:

1. Read the user input.
2. Check for balanced symbols.
3. Convert the expression to postfix if symbols are balanced.

The core data structure used is a stack, which helps match opening and closing symbols. Below is high-level pseudocode for the main tasks:

Listing 1: SymbolBalance Pseudocode

```
function parseInput():
read entire line into inputStr

function checkSyntax(inputStr):
create an empty stack
for each character (or pair for block comments) in inputStr:
if opening symbol -> push onto stack
if closing symbol -> check if top of stack matches
if mismatch or stack empty -> error
else pop from stack
if stack not empty -> error (unclosed opening)
return success or failure

function postfixExpress(inputStr):
```

```
remove all block comments from inputStr
remove whitespace
use an operator stack to convert from infix to postfix
for each token in the expression:
if operand -> add to output
if '(' or '{' or '[' -> push onto stack
       if ')' or '}' or ']' -> pop until matching opening is found
if operator -> pop from stack while top has >= precedence
pop remaining operators
return postfix expression
```

3 Data Structures

- **Stack:** Used to store opening symbols (`(`, {, `[`, `/*`), and to compare them against closing symbols for validation. Also used for operators when converting infix to postfix.
- **Strings:** Used to capture user input, remove block comments, and build the output postfix expression.

4 API Specifications

`parseInput()` Prompts the user and reads a complete expression into a member string.

`checkSyntax()` Validates if the input has balanced symbols. Returns `true` if balanced, otherwise prints an error and returns `false`.

`postfixExpress()` Converts the (already-validated) infix input into a postfix expression. Removes block comments, strips whitespace, and outputs the resulting postfix expression as a string.

5 Example Screenshots

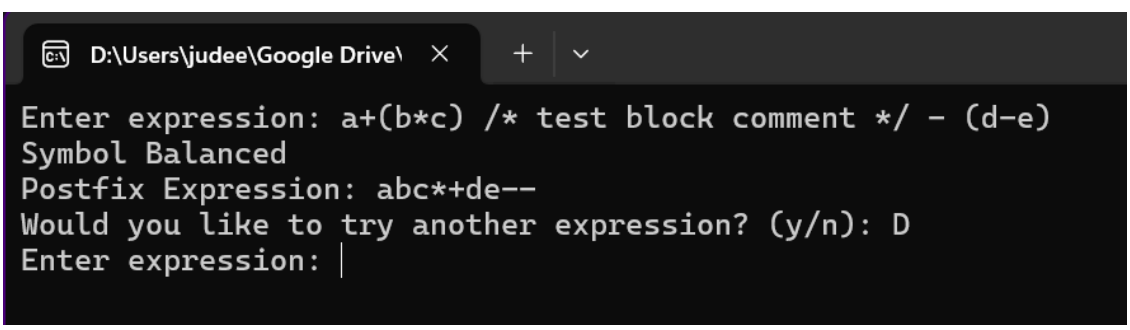

Figure 1: Sample run with balanced parentheses and a comment block.

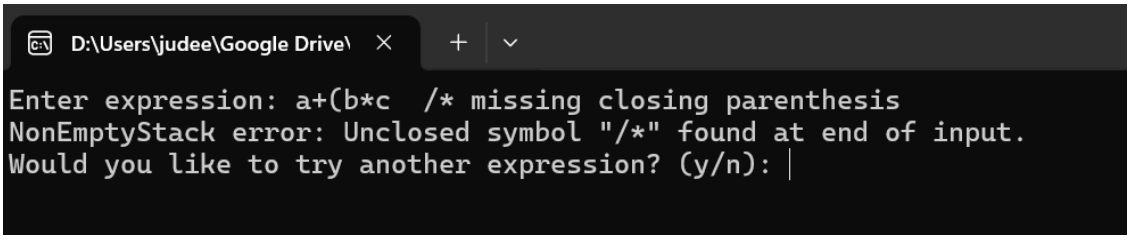

Figure 2: Example of an unbalanced bracket leading to an error message.

6 Additional Information

- The program will prompt you repeatedly for expressions until you choose `n` or `N` to exit.
- Block comments `/* ... */` can appear anywhere in the expression; they are treated as opening and closing tokens on the stack.
- If a user enters a closing comment without a corresponding opening comment, the program will report an *EmptyStack error*.
- The code uses C++ STL features and avoids advanced libraries to maintain simplicity for beginners.