Project Report

on

Calculator Project

Submitted by

Mr. Nont Arayarungsarit st124335

Present to

Prof. Phan Minh Dung

Mr. Akraradet Sinsamersuk

This report is part of the Programing Language and Compilers

Asian Institute of Technology

March 2024

The grammar of the language

<infixToPrefix> ::= <operators> <operands>

<operators> ::= <operator> | <operator> <operators>

<operands> ::= <operand> | <operand> <operands>

<operator> ::= '(' | ')' | '+' | '-' | '*' | '/'

<operand> ::= <character>

<infixToPostfix> ::= <operators> <operands>

<operators> ::= <operator> | <operator> <operators>

<operands> ::= <operand> | <operand> <operands>

<operator> ::= '(' | ')' | '+' | '-' | '*' | '/'

<operand> ::= <character>

*In this calculator, ‘character’ can be a-z 0-9 and A-Z

Type of the parser

In this calculator project uses top-down parsing that is a parsing-method where a input is parsed starting from the root of the parse tree, working recursively down to the leaves of the tree.

Starting by recursive descent parsing, the parser starts from the top-level grammar rule and recursively descends through the input, matching tokens against the grammar rules. Each non-terminal of the grammar corresponds to a function or method in the parser code.

Method of translation

The means of translation, which have been taken are infix to prefix and infix to postfix conversion. These methods are devoted to expressing infix operators (the ones that are written in the middle of operands) either in a prefix form (the operators are written ahead of their operands) or postfix form (the operators are subsequently written after their operands).

Infix to Prefix Conversion (infixToPrefix)

-Using this approach the algorithm reads the infix expression sequentially from left to right.

- It uses two stacks: operator for one (one) and another operand (operands).

- With an operand passing the data onto the operands stack.

When an operator is encountered :

-If it has an opening parenthesis '(', it will pushed onto the operators stack.

-If it has a closing parenthesis ')', operators are popped from the operators stack and operands from the operands stack until an opening parenthesis '(' is encountered. Then, the popped operators and operands are combined in prefix order and pushed back onto the operands stack.

-If it has any other operator, while the stack is not empty and the current operator has lower precedence than or equal precedence to the top operator in the operators stack, operators are popped from the operators stack and operands from the operands stack, combined in prefix order, and pushed back onto the operands stack. Then, the current operator is pushed onto the operators stack.

-If it has any other operator, while the stack is not empty and the current operator has lower precedence than or equal precedence to the top operator in the operators stack, operators are popped from the operators stack and operands from the operands stack, combined in prefix order, and pushed back onto the operands stack. Then, the current operator is pushed onto the operators stack.

-After processing the entire expression, any remaining operators in the operators stack are popped, operands are popped, and they are combined in prefix order and pushed back onto the operands stack.

-The result, which is the prefix expression, is obtained from the operands stack.

Infix to Postfix Conversion (infixToPostfix)

For Infix to Postfix is similar as infix to prefix translation.

Integration of parser and translation

Integration of the translation will happen when user click '=', after parsing input expression, it will call infixToPrefix and infixToPostfix to translate input to output as the name of the function suggest. Infix notation is the standard arithmetic notation where operators are placed between operands, e.g., 2 + 1 * 3. Prefix notation places operators before operands, e.g., *+213. This will do 2 + 1 first because calculator was set using + is high precedence than *, then it will do *3 after already finished 2+1. Postfix notation places operators after operands, e.g., 21+3*, this is the similar mechanism like infixToPrefix as well.

Test

If user puts input through calculator, then calculator can show the output as follows

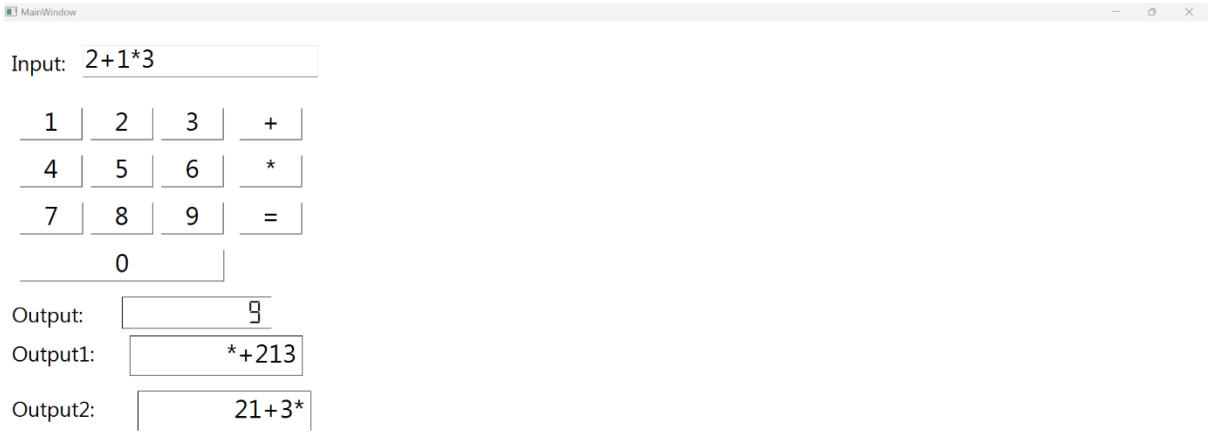

Figure 1 : calculation that + has higher precedence than *

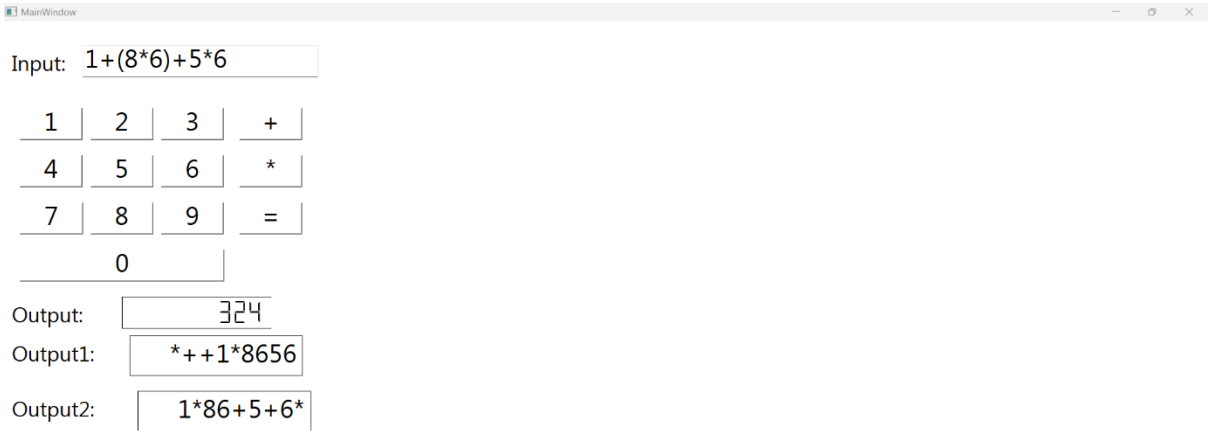

Figure 2 : calculation that () has higher precedence than + and * (optional)

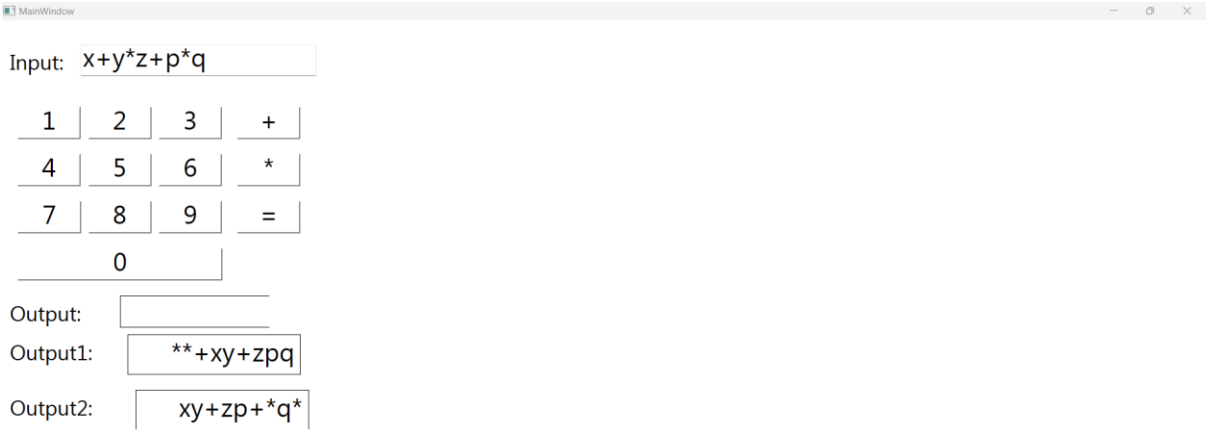

Figure 3 : calculator can take A-Z and a-z as input

Source code

https://github.com/Nont18/Dung-calculator