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

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>

Type of the parser

In this calculator project use top-down parsing that is a parsing-method where a input is parsed starting from the root of the parse tree (with the “Start” symbol), working recursively down to the leaves of the tree.

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)

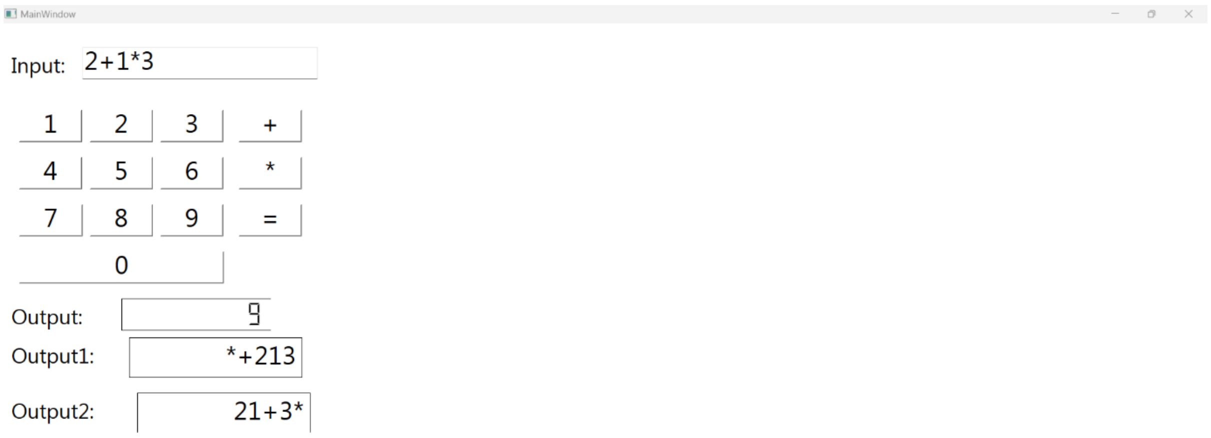
For Infix to Postfix is similar as infix to prefix translation.

<u>Integration of parser and translation</u>

The parser is given the expressions to take, which are then used to convert them to either infix or postfix notation via one of the translation methods, and then processed according to the needs of the application, for example, the ones for evaluation and further compilation. In this calculator, when user clicked equal button, the calculator will indicate 3 outputs at the same time such as A number that already calculated with plus (+) is higher precedence than multiplication (*).

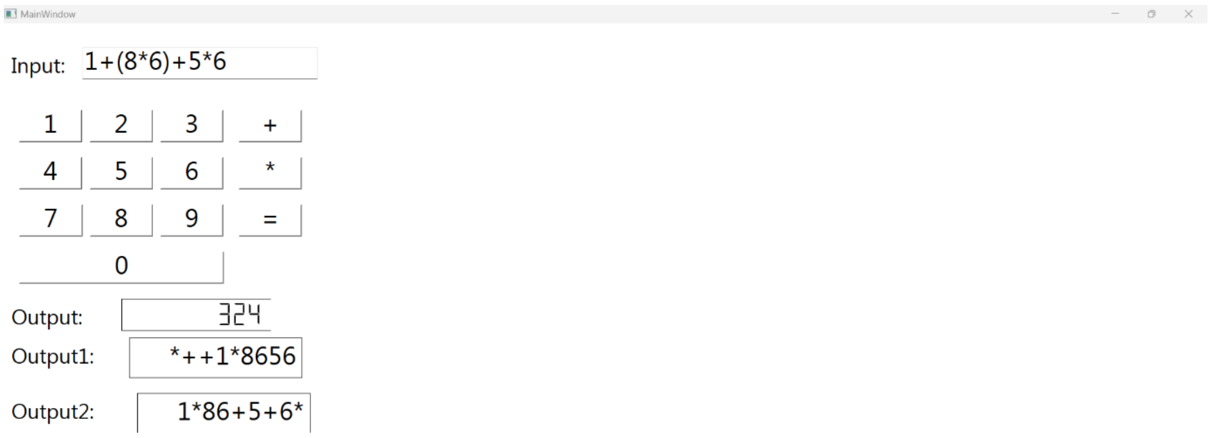
Test

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



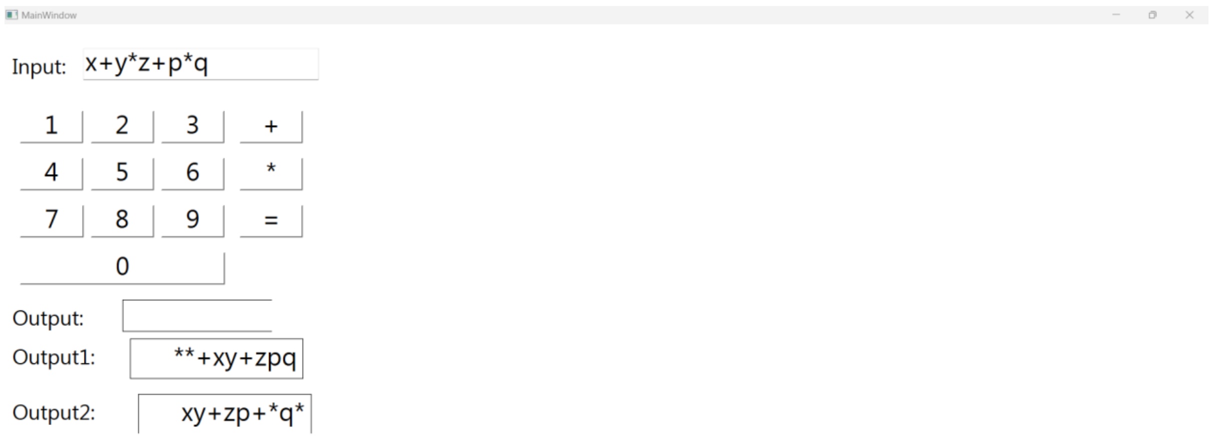
```
Input:   2+1*3
Output:  9
Output1: *+213
Output2: 21+3*
```

Figure 1 : calculation that + has higher precedence than *



```
Input:   1+(8*6)+5*6
Output:  324
Output1: *++1*8656
Output2: 1*86+5+6*
```

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



Input: x+y*z+p*q

Output:

Output1: **+xy+zpq

Output2: xy+zp+*q*

Figure 2 : calculator can take A-Z and a-z as input

Source code

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