ADDS Prac 9 Design

Hideki Yoshinaga | a1658945

UML Diagram

PolishPrefix
-vector<string> numbers - vector<string> operators -int computeResult -string standard_expression
- infix() - binary_operation(string, string, string) -binary_operation(int, string, string) +string getStd() +int getResult() +PolishPrefix(vector<string>)

Description

- PolishPrefix
  - vector<string> numbers：store operants.
  - vector<std::string> operators：store operators.
  - int computeResult：store the result after computing.
  - string standard_expression：store the result after converting.
  - infix()：this function converts Polish prefix to standard expression and compute the result. Using a while loop to to read element from both numbers_vector and operators_vector then using if statements to check the order of operators, if multiplication comes after addition, the first operation needs to be in brackets. The out come standard expression will be stored in standard_expression. While the prefix is converted to standard expression, numbers and operators are passed to binary_operation() to compute the result. Result will be stored in computeResult.

- binary_operation() : This function takes three arguments (number, number and operator) however numbers can be in either int type or string type as it will determine the type and convert it to int type anyway. This function will then apply the binary operation according to the operator and return the result as an int.

- getStd() : return the standard expression.

- getResult() : return the result after computing

- PolishPrefix(vector<string>) : This is the constructor of the PolishPrefix class. The constructor takes a vector as its argument, separates the elements to two different vectors(number, number and operator) and use infix fuction to infix the input and also compute the result.

- Main
  - Collect user input and store them in a string vector called L, each element will be the token. Next, the input will be validate in a while loop. In the while loop, tokens will be read one by one and using if statement to check token is a number or operator otherwise print Error. If token is a number then it will be cheked if it is within 0~99 otherwise print Error. If so, check next token, token after number token must be a number otherwise print Error. Number tokens and operator tokens are counted in the wile loop and after the while loop. The counter value are compared(the number of number tokens minus the number of operator tokens must be 1). After validation, PlishPrefix object will be created and finally the result will be printed.

Testing

1.
   i. Input: * - 3 2 1
   ii. Output: (3 - 2) * 1 = 1

This will check if the program works with correct inputs

2.

i. Input: 9 9 8 * -
ii. Output : Error

This will check if the validation check the order of the input correctly.

3.

i. Input : + 9 6 5 4
ii. Output : Error

the number of number tokens must be one more than the number of operator tokens

4.

i. Input : + - * 6 6 5 -4
ii. Output : Error

Number must be 0~99

5.

i. Input : hfaiuhai hia fooi o oiabib o
ii. Output : Error

Elements other than numbers and operators will retorn Error

6.