Calculator

Sergi Martínez

April 2019

1 Introduction

In this document can be found explained my implementation of the calculator. This includes algorithms, data structures, classes used and other relevant information. I split the code in three main sections, which will be explained below.

2 token.h

Token is a class I had to implement in order to make things easier. Since I split the input string into tokens, I found convenient to create this class. The object Token stores a string which can be an operand or an operator,(for example:"4", "*", "54","/"...), plus an integer that tells if the token is an operand or an operator. If the token is an operand, the class also tells the level of precedence and associativity. These two attributes are very relevant for the algorithm used in the computation of the arithmetic expression.

3 String to tokens

Functions used:***stringToTokens, getType***
In this section of the algorithm the program gets the input string, and turns it into a vector of tokens. I decided to turn the string into tokens because I believe it makes things really simpler, as I have an easier access to the elements.

The algorithm in ***stringToTokens*** basically iterates over the input string character by character, and turns each operand/operator into a token, which is pushed into a regular vector of tokens later on. When a number is found, it stores the character in a buffer and keeps iterating while saving in the buffer all the continuous numbers it founds. When an operator is found, the buffer (which contains the full operand), is pushed back into the vector of tokens. The operator is pushed back too.

This is a pretty primitive way of parsing a string into tokens, but since the input string is not supposed to be really big, I believe it does the trick correctly.
GetType is an auxiliary function I use to check if the "piece" of string I'm trying to turn into a token, it is an operand or an operator.

Implementing this function didn't give a lot of troubles, but there were several special cases which I had to keep in mind. For example, when an operator is found, and the buffer it's empty, it means that the arithmetic expression is incorrect so the execution must be terminated.

4 Infix notation into postfix notation

Functions used:***shuntingYardAlgorithm***
Maybe the most important characteristic of my implementation is that I used a postfix notation instead of infix notation, which means that each operator is placed behind the operands. Thanks to this kind of notation, it's really easy to handle the operator precedence. This postfix notations is also known by the name of "Polish reverse notation".

ShuntingYardAlgorithm basically receives a vector of tokens, and sorts it so it follows a Polish reverse notation. For example, if we have a vector of tokens like this: "654 + 3422 + 55 - 32 / 45 * 56", the function turns it into the following one: "654 3422 + 55 + 32 45 / 56 * - ". Here is where the Token class gets into

action. The algorithm needs check the level of precedence and associativity of the operator, so the class is really convenient here.
Since the algorithm is perfectly explained in the Wikipedia[2] article, I didn't have a lot of troubles implementing this.
If I had more time to make this assignment, I could have implemented the use of parentheses. The algorithm provided in the Wikipedia article explains how to handle parentheses, but it supposes adding at least three or four more conditional statements, making the code much more difficult to read and comprehend.

5 Solving the Reverse Polish notation expression

Functions used:***solveReversePolish, evaluate***
Once the vector of tokens is sorted into a postfix notation, it's time to calculate the final result. Wikipedia[1] also offers a really good way for implementing this. ***SolveReversePolish*** basically receives a vector of tokens, and computes the final result. The auxiliary function ***evaluate*** just computes the last two popped elements of the stack with the current operator (this is part of the Wikipedia's algorithm).

References

[1] Reverse Polish Notation wikipedia. `https://en.wikipedia.org/wiki/Reverse_Polish_notation`.

[2] Shunting-yard algorithm wikipedia. `https://en.wikipedia.org/wiki/Shunting-yard_algorithm`.