
Universidade Federal da Bahia
Instituto de Matemática
Departamento de Ciência da Computação

# Relatório Trabalho 1

André Lima
Bernardo Flores
Carlos Frederico Mendes

06/07/2018
Salvador-Bahia

## 1. Introdução:

Encontrar o resultado de uma expressão infixa é uma tarefa que humanos realizam com facilidade, uma vez acostumados com a precedência dos operadores. Porém, as notações prefixa e posfixa são mais fáceis de avaliar, em termos computacionais, pois não necessitam de ordem de precedência entre os operadores, tampouco parênteses. O "shunting-yard algorithm" de Edsger Dijkstra é um procedimento para converter expressões infixas em posfixas. Adaptando-o, é possível fazê-lo apenas exprimir o resultado da expressão infixa, como descrito em [1].

## 2. Descrição do Algoritmo:

O algoritmo em [1] assume que a expressão está bem formatada. Ele utiliza duas pilhas: uma para operadores e uma para operandos. Sua descrição segue aqui traduzida:

1. Enquanto há caracteres para serem lidos:
  1.1 Leia o próximo caractere.
  1.2 Se o caractere é:
    1.2.1 Um dígito: ponha-o na pilha de operandos.
    1.2.2 Uma variável: obtenha seu valor e ponha-o na pilha de operandos.
    1.2.3 Um parêntese esquerdo: ponha-o na pilha de operadores.
    1.2.4 Um parêntese direito:
      1.2.4 .1 Enquanto o operador no topo da pilha de operadores não for um parêntese esquerdo:
        1.2.4 .1.1 Retire o operador da pilha de operadores.
        1.2.4 .1.2 Retire dois operandos da pilha de operandos.
        1.2.4 .1.3 Aplique o operador aos operandos na ordem correta.
        1.2.4 .1.4 Ponha o resultado na pilha de operandos.
      1.2.4 .2 Retire o parêntese esquerdo da pilha de operadores e descarte-o.
    1.2.5 Um operador (chamemo-lo Op):
      1.2.5 .1 Enquanto a pilha de operadores não estiver vazia, e o operador no topo da pilha de operadores tiver a mesma precedência ou uma precedência maior que Op:
        1.2.5 .1.1 Retire o operador da pilha de operadores.
        1.2.5 .1.2 Retire dois operandos da pilha de operandos.
        1.2.5 .1.3 Aplique o operador aos operandos na ordem correta.
        1.2.5 .1.4 Ponha o resultado na pilha de operandos.
      1.2.5 .2 Ponha Op na pilha de operadores.
2. Enquanto a pilha de operadores não estiver vazia:
  2.1 Retire o operador da pilha de operadores.
  2.2 Retire dois operandos da pilha de operandos.
  2.3 Aplique o operador aos operandos na ordem correta.
  2.4 Ponha o resultado na pilha de operandos.
3. Nesse ponto a pilha de operadores deve estar vazia e a pilha de operandos deve ter apenas um valor, que é o resultado final da expressão.

Adaptamos tal algoritmo para que possa avaliar também se a expressão está bem formatada no que diz respeito aos parênteses. Se no laço em 1.2.4.1, a pilha ficar vazia, sabemos que há um parêntese direito sem o correspondente esquerdo. Se no laço em 2. surgir um parêntese esquerdo, sabemos que ele não possui o correspondente direito. Nesses casos, podemos imprimir "Erro de formatação".

## 3. Decisões de Implementação:

Na nossa implementação, a precedência das operações + e – é dita 1. A das operações * e / é dita 2. E a precedência do parêntese esquerdo é dita 0. Optamos por tratar os casos em que há espaços entre os operandos e os operadores, porém decidimos não dar suporte a operações em ponto flutuante. Vale observar que os operandos devem se limitar aos números de 0 a 9 (números que podem ser representados por um único dígito decimal). E deve-se atentar para não confundir o sinal da operação de subtração '-' (hífen) com a meia risca ou o travessão. As nossas pilhas de operandos e de operadores suportam no máximo 500 itens cada, sem contar parênteses direitos, que não fazem diferença nesta contagem. É preferível que após a compilação com o NASM, seja feita a linkagem com o gcc, visto que a função principal chama-se "main" e não "_start", como esperado pelo ld.

## 4. Conclusão:

Conseguimos obter o resultado esperado, após *muitas* horas de edição e depuração do código.

## 5. Bibliografia:

[1] https://www.geeksforgeeks.org/expression-evaluation/ Acessado pela última vez em 06/07/2018.