

# Lista 8 - MAC0122 Princípios de Desenvolvimento de Algoritmos - POLI

Prof. Ronaldo Fumio Hashimoto

## 1 Exercícios

Os exercícios de 9 a 17 correspondem, respectivamente, aos exercícios de 4.9 a 4.17 do livro de Robert Sedgewick [1].

9. Converta a expressão abaixo para a notação posfixa.

$( 5 * ( ( 9 * 8 ) + ( 7 * ( 4 + 6 ) ) ) )$

10. Abaixo são apresentados uma figura (Figura 4.2 do livro) e um programa (Programa 4.2 do livro). Desenhe uma figura semelhante a essa, com o conteúdo da pilha conforme a expressão abaixo é processada pelo programa.

5 9 \* 8 7 4 6 + \* 2 1 3 \* + \* + \*

5	5			
9	5	9		
8	5	9	8	
+	5	17		
4	5	17	4	
6	5	17	4	6
*	5	17	24	
*	5	408		
7	5	408	7	
+	5	415		
*	2075			

Figura 1: Figura adaptada de [2].

### Cálculo de uma expressão posfixa

Essa sequência exhibe o uso de uma pilha para calcular a expressão posfixa 5 9 8 + \* \* 7 + \*. Calculando a expressão da esquerda para a direita, se encontrarmos um número, ele é empilhado; se encontramos um operador, nós empilhamos o resultado do operador aplicado aos dois últimos números do topo da pilha.

#### Programa 4.2

```
#include <stdio.h>
#include <string.h>
#include "Item.h"
#include "STACK.h"
main(int argc, char *argv[])
{ char *a = argv[1]; int i, N = strlen(a);
  STACKinit(N);
  for (i = 0; i < N; i++)
  {
    if (a[i] == '+')
      STACKpush(STACKpop() + STACKpop());
    if (a[i] == '*')
      STACKpush(STACKpop() * STACKpop());
    if ((a[i] >= '0') && (a[i] <= '9'))
      STACKpush(0);
    while ((a[i] >= '0') && (a[i] <= '9'))
      STACKpush(10*STACKpop() + (a[i++] - '0'));
  }
  printf("%d \n", STACKpop());
}
```

11. Estenda o Programa do exercício 10 (Programa 4.2 do livro) e o programa abaixo (Programa 4.3 do livro) para incluir as operações de subtração (-) e divisão (/).

#### Programa 4.3

```
#include <stdio.h>
#include <string.h>
#include "Item.h"
#include "STACK.h"
main(int argc, char *argv[])
{ char *a = argv[1]; int i, N = strlen(a);
  STACKinit(N);
  for (i = 0; i < N; i++)
  {
    if (a[i] == ')')
      printf("%c", STACKpop());
    if ((a[i] == '+') || (a[i] <= '*'))
      STACKpush(a[i]);
    if ((a[i] >= '0' && (a[i] <= '9'))
      printf("%c", a[i]);
  }
  printf("\n");
}
```

12. Estenda sua solução do Exercício 11 para incluir os operadores unários  $-$  (negação) e  $\$$  (raiz quadrada). Modifique também a pilha abstrata do programa do exercício 10 (programa 4.2 do livro) para a utilização de números em ponto flutuante. Por exemplo, dada a expressão

$$(-(-1) + \$((-1) * (-1) - (4 * (-1))))/2$$

seu programa deve imprimir o valor 1.618034

13. Escreva um programa PostScript que desenha a figura abaixo:



14. Prove por indução que o programa do exercício 10 (Programa 4.2 do livro) calcula corretamente qualquer expressão posfixa.
15. Escreva um programa que converta uma expressão posfixa para infixa, utilizando uma pilha (*pushdown stack*).
16. Combine os programas dos exercícios 4.2 e 4.3 (Programa 4.2 e 4.3 do livro) em um único módulo que use duas ADTs (*Abstract Data Type*) de pilha diferentes: uma pilha de inteiros e uma pilha de operadores.
17. Implemente um compilador e interpretador para uma linguagem de programação onde cada programa consiste de uma única expressão aritmética precedida por uma sequência de atribuições envolvendo inteiros e variáveis nomeadas com um único caractere minúsculo. Por exemplo, dada a entrada

```
(x = 1)
(y = (x + 1))
(((x + y) * 3) + (4 * x))
```

seu programa deve imprimir o valor 13.

## Referências

- [1] R. Sedgewick, “Algorithms in c—third edition,” p. 144, 1998.
- [2] R. Sedgewick, “Algorithms in c—third edition,” p. 139, 1998.