

Exercício: Pilhas

Disciplina: FGA0147 - Estruturas de Dados e Algoritmos

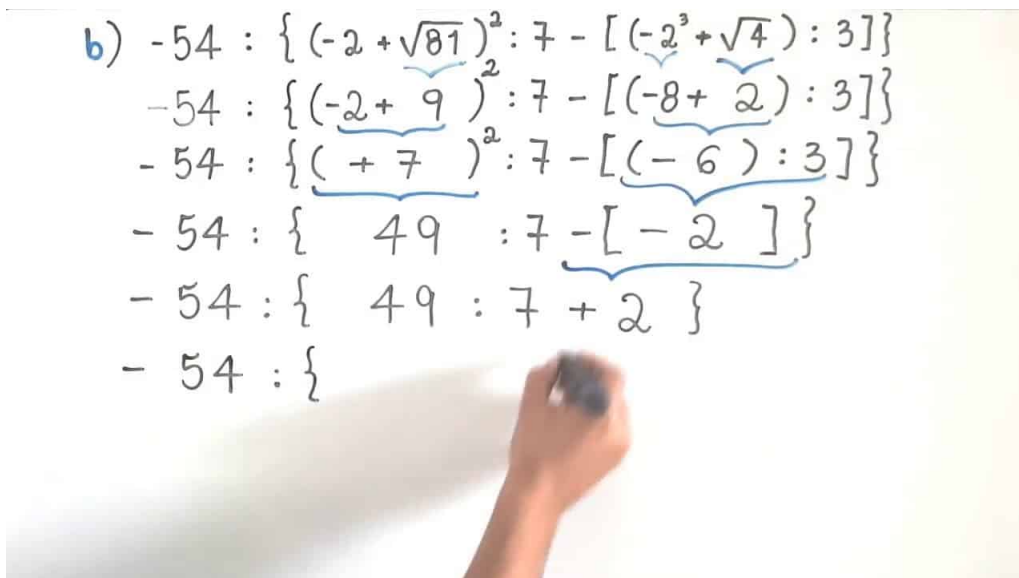
Prof. Dr. Nilton Correia da Silva

Faculdade UnB Gama - FGA

Universidade de Brasília - UnB

## Aplicação de Pilha para Verificar Escrita de Expressões Algébricas.

---



b)  $-54 : \{ (-2 + \sqrt{81})^2 : 7 - [(-2^3 + \sqrt{4}) : 3] \}$   
 $-54 : \{ (-2 + 9)^2 : 7 - [(-8 + 2) : 3] \}$   
 $-54 : \{ (+7)^2 : 7 - [(-6) : 3] \}$   
 $-54 : \{ 49 : 7 - [-2] \}$   
 $-54 : \{ 49 : 7 + 2 \}$   
 $-54 : \{$

# Sumário

<b>1</b>	<b>Introdução</b>	<b>3</b>
<b>2</b>	<b>Objetivo</b>	<b>4</b>
<b>3</b>	<b>Algoritmo</b>	<b>5</b>
<b>4</b>	<b>Estrutura Básica de Código</b>	<b>6</b>
4.1	Tipo Pilha . . . . .	6
4.2	Main . . . . .	6
4.3	Varredura da Expressão: Verificação dos abre e fecha parênteses e colchetes . . . . .	6
4.4	Função Auxiliar: Verifica abertura e fechamento . . . . .	7



### **Orientações**

Exercício prático para sedimentação dos conhecimentos de estrutura de dados do tipo PILHA.

A solução deverá ser em linguagem C ou C++. No caso do aluno adotar C++, poderá se valer de uma solução orientada a objetos, contudo sua solução não deve usar classes e contêineres já prontos da linguagem C++ - por exemplo, algoritmos de ordenação, busca, objetos vector, list, matrix, etc.



# 1 Introdução

A Pilha é uma estrutura de dados com regras de acesso: O último elemento armazenado é o primeiro a ser removido: LIFO (Last In, First Out).

O conceito de pilha é usado em muitos softwares de sistemas incluindo compiladores e interpretadores. (A maioria dos compiladores C usa pilha quando passa argumentos para funções).

Neste exercício teremos a oportunidade de exercitar Pilha para fazer a verificação do uso parênteses e conchetes em expressões aritméticas.



## 2 Objetivo

A solução deste exercício consiste em verificar se os parênteses e colchetes de uma expressão aritmética estão sendo usados de maneira correta. Ou seja, sua solução deve verificar um texto contendo uma expressão aritmética conforme abaixo:

**Todos os parênteses abertos devem ser fechados.**

**Todos os colchetes abertos devem estar fechados.**



### 3 Algoritmo

Lógica para resolução:

1. Iniciar uma pilha vazia.
2. Obtenha o próximo caractere C da esquerda para direita.
3. Se C for um abre parênteses ou abre colchete, empilhar.
4. Se C for um fecha parênteses ou fecha colchete:
  - (a) Se C não for o fechamento do topo da pilha, reporte erro.
  - (b) Se C for o fechamento do topo da pilha, desempilhe.
5. Volte ao item 2. .

Ao final, se a pilha não estiver vazia significa que a expressão está mal formada



## 4 Estrutura Básica de Código

Elementos de código em C Ansi para uso na solução.

### 4.1 Tipo Pilha

```
1 // Tipo a ser usado para sua pilha
2
3 typedef struct no{
4     char character;
5     struct no *proximo;
6 }No;
7
```

### 4.2 Main

```
1
2 // Função Main
3 int main(){
4     char exp[50];
5
6     printf("\tDigite um expressao: ");
7     scanf("%49[^\n]", exp);
8     printf("\nExpressao: %s\nRetorno: %d\n", exp, identifica_formacao(exp));
9 }
```

### 4.3 Varredura da Expressão: Verificação dos abre e fecha parênteses e colchetes

```
1
2 // Função identifica_formacao
3 // Assinatura: int identifica_formacao(char x[]);
4 // Retorno:
5 //.    0 - Caso a expressão de entrada está bem formada
6 //.    1 - Caso a expressão de entrada não esteja bem formada
7 //
8
9 int identifica_formacao(char x[]){
10
11     /* #####
12         VOCÊ DEVE IMPLEMENTAR ESTE CÓDIGO.
13     */ #####
14
15 }
```



#### 4.4 Função Auxiliar: Verifica abertura e fechamento

```
1
2 // Função identifica_formacao
3 // Assinatura: int forma_par(char f, char d);
4 // Retorno:
5 //.    0 - Caso f não seja o fechamento de d
6 //.    1 - Caso f seja o fechamento de d
7 //
8 int forma_par(char f, char d){
9     switch(f){
10     case ')':
11         if(d == '(')
12             return 1; // bem formada
13         else
14             return 0; // mal formada
15         break;
16     case ']':
17         if(d == '[')
18             return 1; // bem formada
19         else
20             return 0; // mal formada
21         break;
22     case '}':
23         if(d == '{')
24             return 1; // bem formada
25         else
26             return 0; // mal formada
27         break;
28     }
29 }
```

