Pilhas

Prof. Dr. Marcelo Fernando Rauber

São estruturas de dados do tipo **L.I.F.O.** (Las In, First Out), o que significa que o último elemento inserido é o primeiro a ser removido.

Seu funcionamento computacional pode ser comparado com as pilhas não computacionais, como, por exemplo, uma pilha de pratos ou de livros. Ao adicionarmos um prato a pilha, o natural é que seja colocado sobre o topo da pilha. Da mesma forma, ao removermos um prato, o faremos do topo.

Ao implementar este tipo de estrutura, temos acesso apenas ao último elemento da pilha (chamado de topo), sendo que para termos acesso ao penúltimo elemento, devemos remover o último.

São exemplos de aplicações de pilhas computacionais:

- Gerenciamento de Chamadas Recursivas
- Avaliação de expressões numéricas;
- Os mecanismos de desfazer/refazer de editores de texto;
- Navegação entre páginas web;
- Algoritmos para notação polonesa;
- etc.

Temos 3 operações básicas com pilhas:

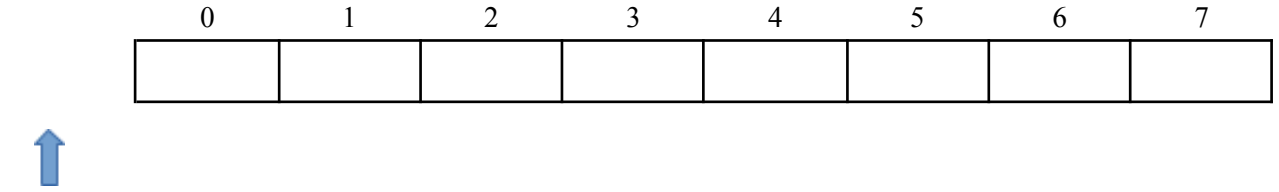
- push: Empilhar ou inserir um novo elemento no topo;
- pop: Retirar ou remover o elemento do topo;
- top: Observar ou listar o elemento do topo;

A implementação de pilhas pode ser feita de forma estática ou dinâmica. Na forma estática usamos como base um vetor. Já na implementação dinâmica, usamos alocação dinâmica de memória. A natureza dos elementos a serem armazenados pode ser distinta: int, String ou um Tipo Abstrato de Dados, etc. O Importante é o mecanismo de funcionamento (último a entrar é o primeiro a sair). Ao ser implementado utilizando vetores (que tem tamanho máximo pré-definido), devemos cuidar para não ultrapassarmos a capacidade máxima de armazenamento, evento este conhecido como estouro de pilha (Quem lembra da tela azul do Windows?).

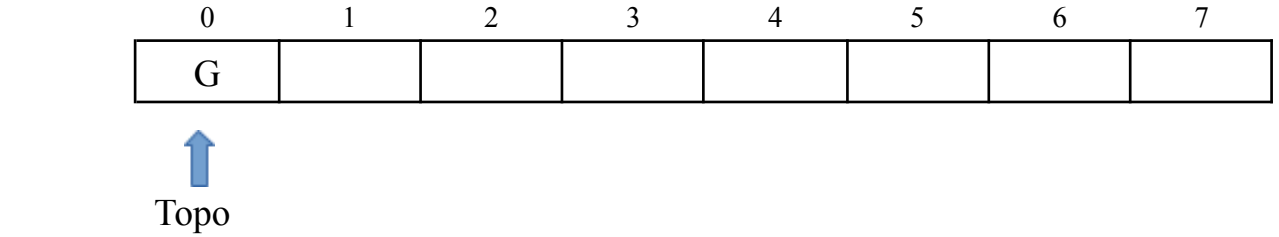
Um exemplo de Implementação de PILHA Estática

Imaginemos uma pilha implementada através de vetor destinada a armazenar **letras**. Precisamos de um vetor e uma variável auxiliar (int) para armazenar o índice do topo. De início, com a pilha vazia, o topo tem valor -1.

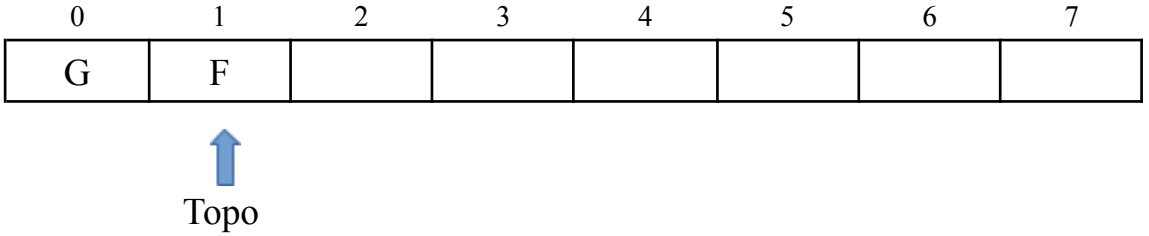
No início, a pilha **vazia** poderia ser assim representada:



Ao Empilhar a letra G, incrementamos o valor da variável auxiliar Topo e armazenamos o valor no vetor no índice correspondente. A pilha poderia ser assim representada:



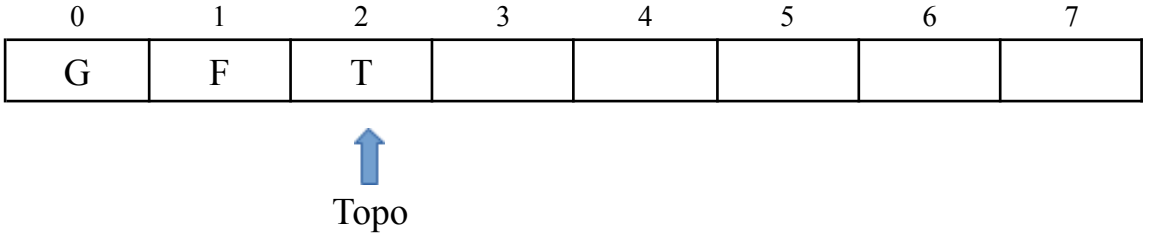
Numa seguinte operação poderíamos empilhar a letra F, e a pilha poderia ser assim representada:



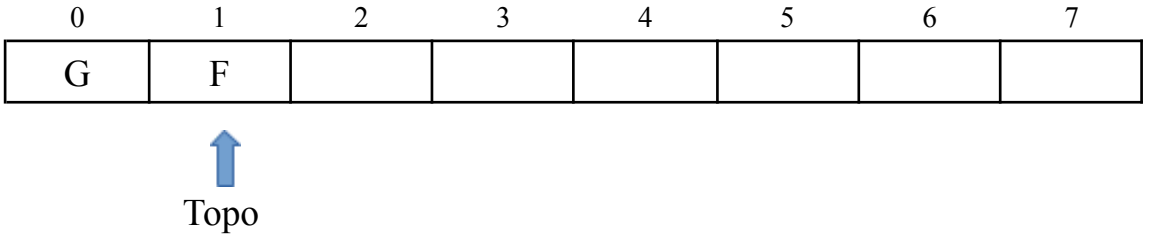
O trecho de código em Java para realizar o empilhamento da letra F, poderia ser assim implementado:

```
if (Topo < vetor_pilha.length-1) {
    Topo = Topo + 1;
    vetor_pilha[Topo] = 'F';
} else {
    System.out.println ("Estouro de pilha");
}
```

Numa seguinte operação, continuando a empilhar, desta vez a letra T, a pilha poderia ser assim representada:



De forma análoga, ao REMOVER um elemento (pop), o fazemos do topo. O importante é decrementar o valor do topo. Assim, ao remover um elemento, a pilha poderia ser assim representada:



Exercício (Não precisa Entregar)

Elabore um programa completo em Java no NetBeans, que implementa uma pilha estática para armazenar de nomes de pessoas com um vetor de Strings. Apresente ao usuário e torne funcional o seguinte menu:

Menu:
1 – Empilhar nome (push);
2 – Mostrar elemento no topo (top);
3 – Desempilhar nome (pop);
4 – Sair.

Exemplo - Um algoritmo que utiliza pilha para analisar a adequada utilização de parênteses em uma expressão.

```
package teste.pilhaparenteses;

import java.util.Scanner;

/**
 *
 * @author Prof_Marcelo_Rauber
 */
public class PilhaParenteses {

    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.println("Digite a expressão: ");
        //String expressao = teclado.nextLine();
        String expressao = "(2+4/(8*5)<(3))";
        //String expressao = "(2+4))";

        if (verificarParenteses(expressao)) {
            System.out.println("Os parênteses estão balanceados.");
        } else {
            System.out.println("Os parênteses NÃO estão balanceados.");
        }
    }

    static boolean verificarParenteses(String expressao) {
        int tamanho = expressao.length();
        char pilha[] = new char[tamanho];
        int topo = -1;

        for (int i = 0; i < tamanho; i++) {
            char c = expressao.charAt(i);
            if (c == '(') {
                topo = empilharPush(pilha, c, topo);
                if (topo == -1) {
                    //System.out.println("Estouro de pilha, tela azul kkk");
                    return false; // Pilha cheia, erro
                }
            } else if (c == ')') {
                topo = desempilharPop(pilha, topo);
                if (topo < -1) {
                    return false; // Pilha vazia, erro
                }
            }
        }
        return topo == -1; // Se a pilha estiver vazia, está balanceado
    }

    static int empilharPush(char pilha[], char elemento, int topo) {
        int tamanho = pilha.length;
        if (topo == tamanho - 1) {
            return -1; // Pilha cheia
        }
        topo += 1;
        pilha[topo] = elemento;
        return topo;
    }

    static int desempilharPop(char pilha[], int topo) {
        //System.out.println("Removendo da pilha o seguinte elemento: " +
pilha[topo]);
```

```
        topo = topo - 1;
        return topo;
    }
}
```