

# SIN 211 - Algoritmos e Estruturas de Dados

(Pilha Estática)

Prof<sup>o</sup>: Joelson Antônio dos Santos

Universidade Federal de Viçosa  
Instituto de Ciências Exatas e Tecnológicas  
Campus de Rio Paranaíba - MG

*joelsonn.santos@gmail.com*  
*Sala: BBT 233*

24 de abril de 2018

# Aula de Hoje

## 1 Pilha Estática

# Créditos

O material desta aula é composto por adaptações e extensões dos originais gentilmente cedidos pelos professores **Moacir Pereira Ponti** e **Rachel Reis**.

# Pilha



- Definição:
  - Uma estrutura linear de dados que pode ser acessada somente por uma de suas extremidades para armazenar e recuperar dados.
- Uma pilha é chamada de uma estrutura LIFO (do inglês *last in/first out*).

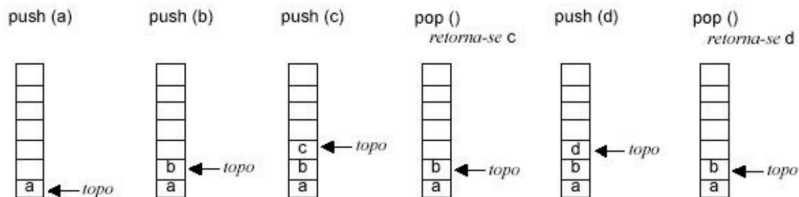
- Aplicações:
  - Pilhas de execução de um programa.
  - Casamento de delimitadores.
  - Notação posfixa.

# Pilha - Operações

- *clear()* - Limpa a pilha.
- *isEmpty()* - Verifica se está vazia.
- *push(el)* - Coloca (empilha) um elemento no topo.
- *pop()* - Retira (desempilha) o elemento do topo.
- *topEl()* - retorna o elemento do topo sem removê-lo.

# Pilha - Exemplo

- Sequência de operações de empilhar e desempilhar.





# Pilha - Operações

- Quais outras operações devemos adicionar a uma implementação de uma *TAD* pilha estática?

# Pilha - Operações

- Inicializar
- Verificar se está cheia.
- Quantidade de elementos na pilha.

# Pilha Estática

- Para uma implementação estática de pilha, usamos um vetor com tamanho fixo.
- Quais as vantagens dessa implementação? E as desvantagens?

# Pilha Estática

- Definição e declaração de uma pilha estática:

```
#include <stdio.h>
#define TAM 10

typedef struct stack{
    int elem[TAM];
    int topo;
}PILHA;


int main(){
    PILHA p;
    return 0;
}
```

# Pilha Estática - Inicializar

- Inicializamos a pilha como “vazia”. Assim como na lista estática, atribuímos um valor inválido para a variável **topo**.


```
void inicializarPilha(PILHA* pilha){  
    pilha->topo = -1;  
}
```

# Pilha Estática - Verificar Vazia



```
int pilhaVazia(PILHA* pilha){  
    if(pilha->topo == -1){  
        return 1;  
    }  
    return 0;  
}
```

# Pilha Estática - Verificar Cheia




```
int pilhaCheia(PILHA* pilha){  
    if(pilha->topo == (TAM - 1)){  
        return 1;  
    }  
    return 0;  
}
```

# Pilha Estática - *Push*

```
int push(PILHA* pilha, int elem){  
    if(pilhaCheia(pilha)){  
        printf("\n Erro: Pilha cheia!");  
        return 0;  
    }  
    pilha->elem[pilha->topo+1] = elem;  
    pilha->topo++;  
    return 1;  
}
```



# Pilha Estática - *Pop*


A vertical stack structure is shown on the left side of the slide. It consists of a vertical line with three horizontal bars. The top two bars have small squares to their left, representing elements in the stack. The bottom bar is open on the left side.

```
int pop(PILHA* pilha){
    int removido = -1; //elem inválido
    if(pilhaVazia(pilha)){
        printf("\n Pilha vazia");
        return removido;
    }
    removido = pilha->elem[pilha->topo];
    pilha->topo--;
    return removido;
}
```

- Por que nesta função estamos retornando um inteiro?

# Pilha Estática - *Topo*

- Retornar o elemento do topo sem removê-lo.

A vertical diagram on the left side of the code block represents a static stack. It consists of a vertical line with two square boxes at the top, each containing a minus sign (-). Below these boxes, the line continues down with a few horizontal tick marks, and ends with a curly brace (}).

```
int topoEl(PILHA* pilha){  
    int topo = -1; //elem inválido  
    if(pilhaVazia(pilha)){  
        printf("\n Pilha vazia");  
        return topo;  
    }  
    topo = pilha->elem[pilha->topo];  
    return topo;  
}
```

- Como podemos pesquisar um elemento que está no meio de uma pilha?
- Como eu posso imprimir todos os elementos de uma pilha?

# Pilha - Teste

```
int main(){
    PILHA p;
    inicializarPilha(&p);
    push(&p, 10);
    printf("\n Topo: %d", topoEl(&p));
    push(&p, 15);
    printf("\n Topo: %d", topoEl(&p));
    push(&p, 20);
    printf("\n Topo: %d", topoEl(&p));
    push(&p, 30);
    printf("\n Topo: %d", topoEl(&p));
    pop(&p);
    printf("\n Topo: %d", topoEl(&p));
    return 0;
}
```

# Bibliografia Básica

- DROZDEK, Adam. Estrutura de Dados e Algoritmos em C++. Editora Pioneira Thomson Learning, 2005. Capítulo 4.