Algoritmos Computacionais e Estruturas de Dados

Prof. Me. Felipe Borges

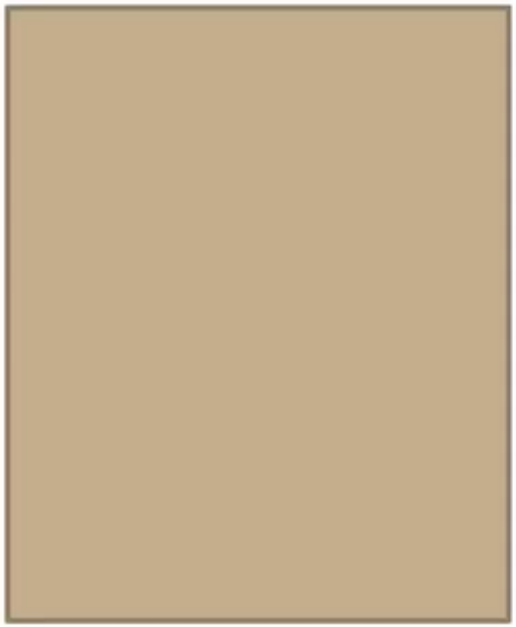
Prof. Felipe Borges

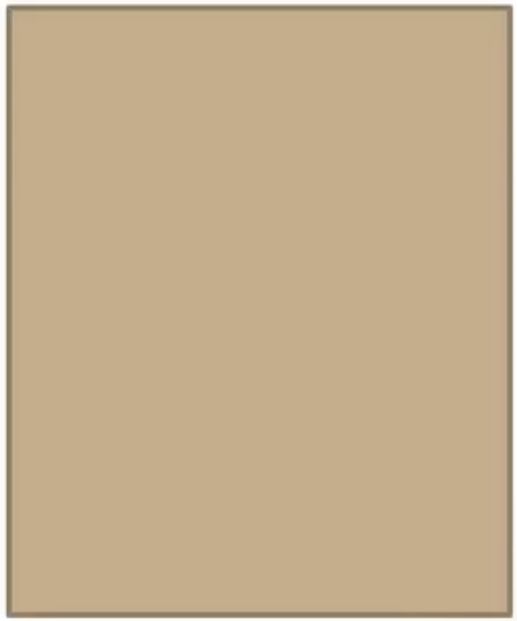
Doutorando em Sistemas de Potência – UFMA – Brasil
Mestre em Sistemas de Potência – UFMA – Brasil
MBA em Qualidade e Produtividade – FAENE – Brasil
Graduado em Engenharia Elétrica – IFMA – Brasil
Graduado em Engenharia Elétrica – Fontys – Holanda
Técnico em Eletrotécnica – IFMA – Brasil

Projetos e Instalações Elétricas – Engenharia – Banco do Brasil
Desenvolvimento e Gestão de Projetos – Frencken Engineering BV

PILHA (stack)

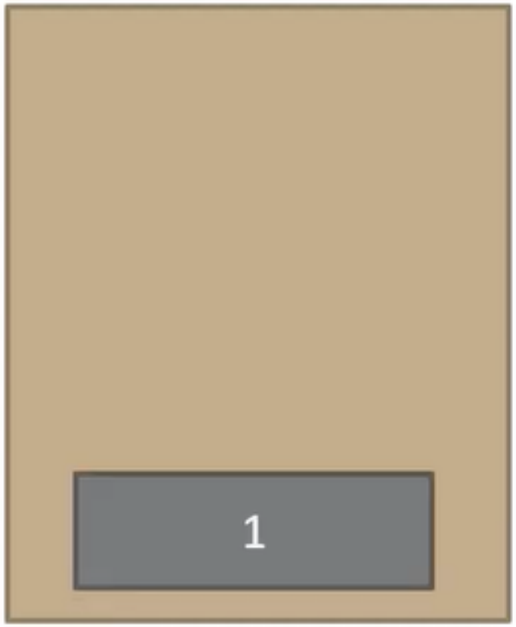
- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
  - Similar a uma pilha de livros
  - LIFO – Last In, First Out
    - O último a entrar é o primeiro a sair

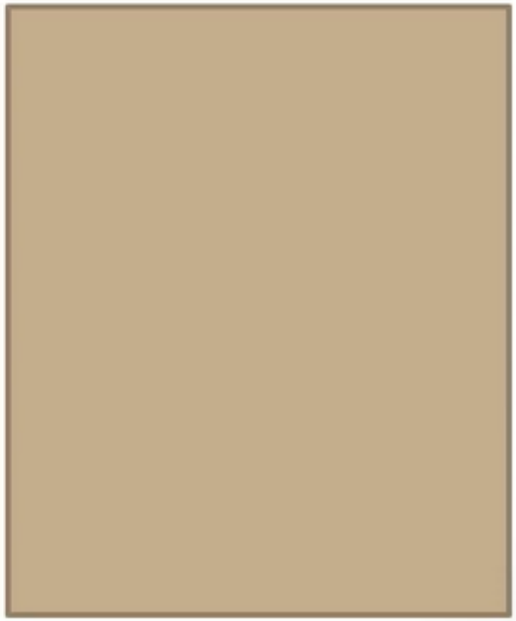




PILHA (stack)

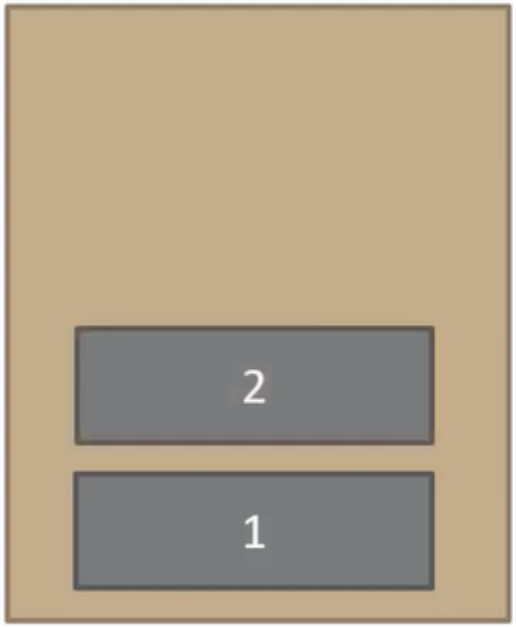
- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
  - Similar a uma pilha de livros
  - LIFO – Last In, First Out
    - O último a entrar é o primeiro a sair

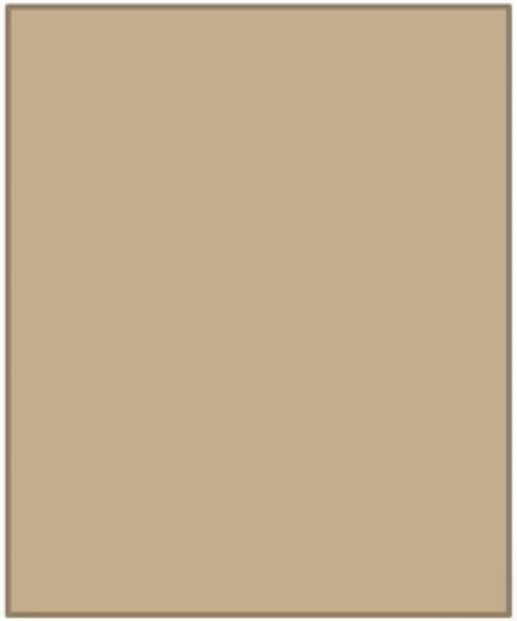




PILHA (stack)

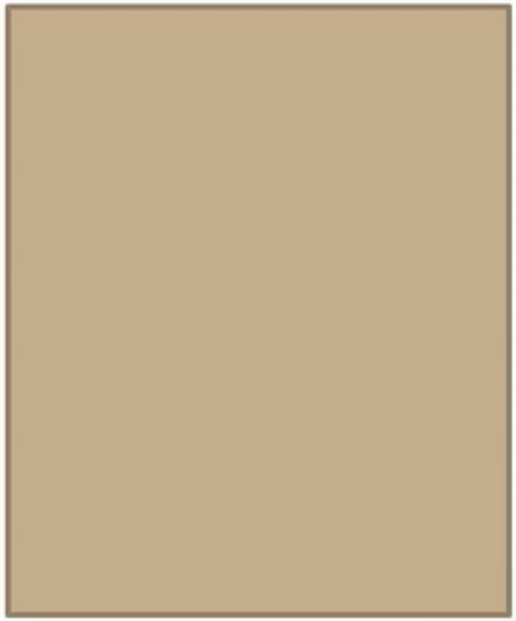
- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
  - Similar a uma pilha de livros
  - LIFO – Last In, First Out
    - O último a entrar é o primeiro a sair





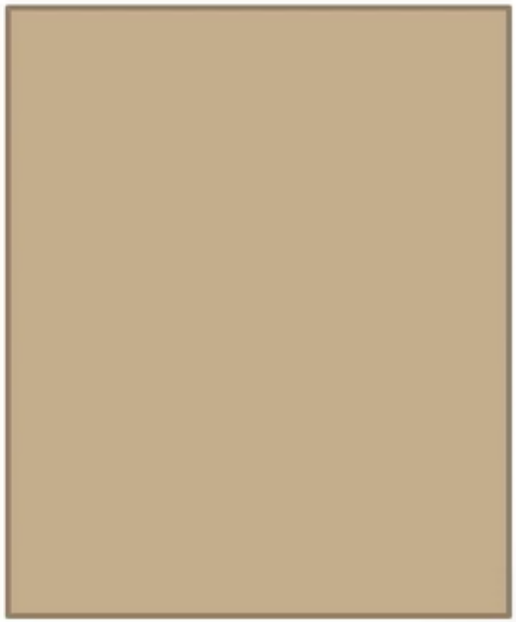
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
  - Similar a uma pilha de livros
  - LIFO – Last In, First Out
    - O último a entrar é o primeiro a sair



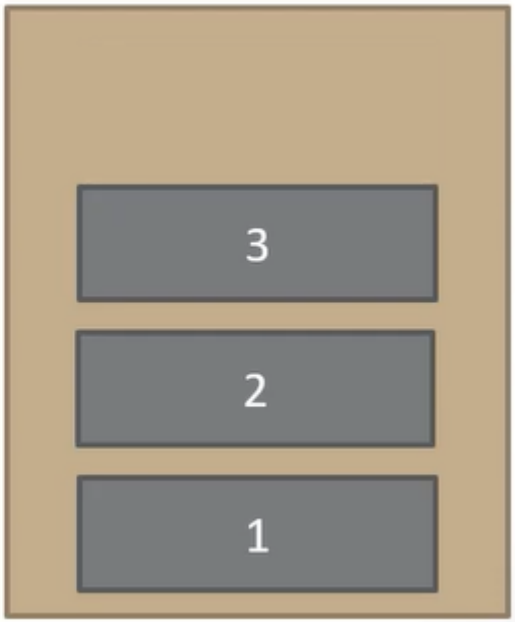
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
  - Similar a uma pilha de livros
  - LIFO – Last In, First Out
    - O último a entrar é o primeiro a sair



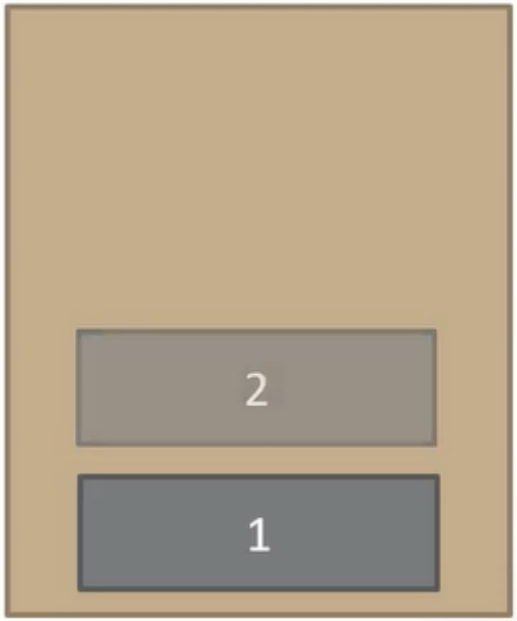
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
  - Similar a uma pilha de livros
  - LIFO – Last In, First Out
    - O último a entrar é o primeiro a sair



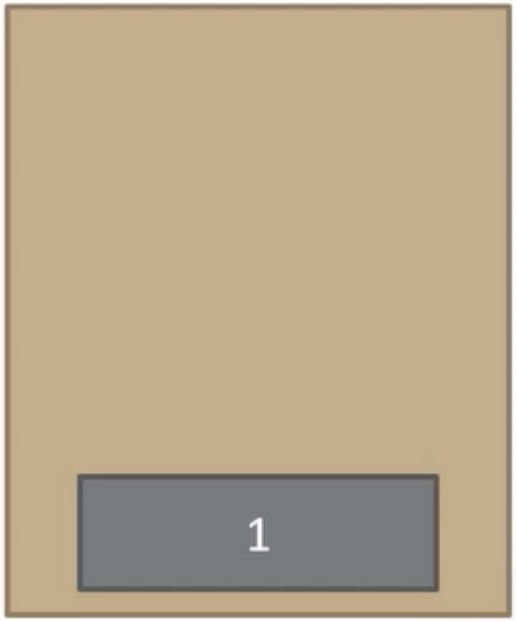
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
  - Similar a uma pilha de livros
  - LIFO – Last In, First Out
    - O último a entrar é o primeiro a sair



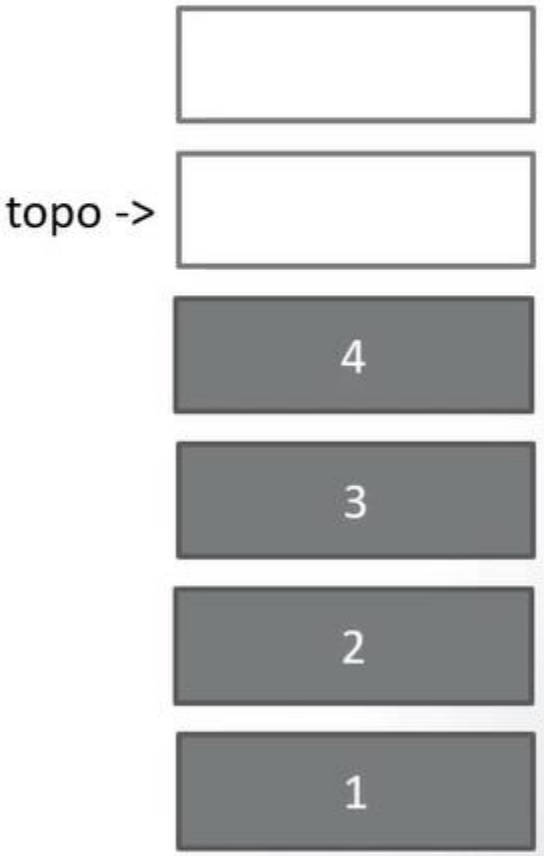
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
  - Similar a uma pilha de livros
  - LIFO – Last In, First Out
    - O último a entrar é o primeiro a sair



PILHA (stack)

- Características
  - Número de elementos - *n*
    - Máximo número de elementos que cabe na pilha
  - Topo – *topo*
    - Posição onde um novo elemento entrará
- Operações
  - Empilhar
    - Adiciona um novo elemento ao topo
    - Incrementa o topo
    - Stack overflow (pilha cheia) se *topo = n*
  - Desempilhar
    - Remover um elemento
    - Decrementa o topo
    - Stack empty (pilha vazia) se *topo = 0*



PILHA (stack)

- Pode ser implementada com vetor

- Variáveis de controle da pilha
  - Número de elementos - *n*
    - Máximo número de elementos que cabe na pilha

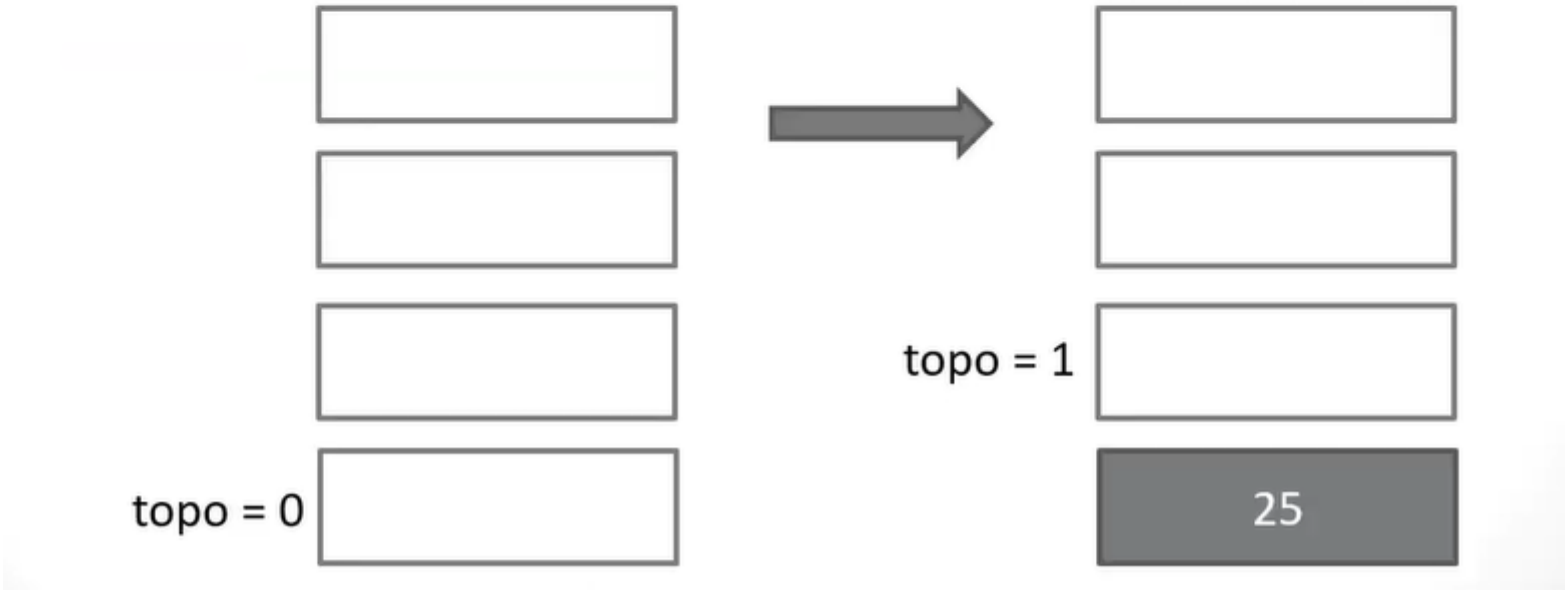
  - Topo – *topo*
    - Posição onde um novo elemento entrará

PILHA (stack)

- Aplicações
  - Controle de fluxo de programas
  - Quando o usuário chama uma função em um programa
    1. O sistema operacional adiciona a uma pilha (stack) a posição da memória de código onde o programa estava no momento da chamada da função
    2. O controle de programa salta para uma posição na memória de código onde inicia o código da função
    3. Após a função terminar (return), o sistema operacional verifica na pilha a posição da memória de código onde o programa deve continuar
  - Stack overflow
    - Ocorre quando o usuário chama muitas vezes uma função dentro de outra, a ponto da pilha encher e o programa não mais conseguir retornar
    - Pode ocorrer em microcontroladores e outros sistemas que possuem pouca memória, onde a pilha é pequena – limite de chamadas

PILHA (stack)

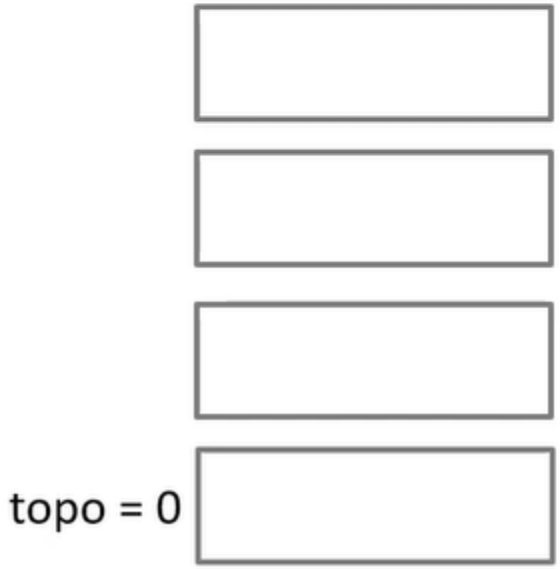
```
int pilha[20];          // cria uma vetor de 20 elementos
int topo = 0;           // define onde esta o topo da pilha

// empilha um elemento
pilha[topo] = 25;
topo++;
```



PILHA (stack)

Um exemplo de struct para uma pilha

```
typedef struct {
      int pilha[4];    // tamanho da pilha
      int topo;
} Pilha;
```



Exemplo - PILHA

- Faça um programa que implemente uma pilha de 20 elementos do tipo inteiro utilizando struct
  - Implemente um método que empilhe um novo inteiro
    - O método deve enviar uma mensagem se a pilha estiver cheia
    - void empilha(int valor, Pilha *pilha);
  - Implemente um método que desempilhe
    - O método deve exibir uma mensagem se a pilha estiver vazia
    - void desempilha(Pilha *pilha);
  - Implemente um método que retorne 1 se a pilha está cheia e 0 se não
    - int isCheia(Pilha *pilha);
  - Implemente um método que retorne 1 se a pilha está vazia e 0 se não
    - int isVazia(Pilha *pilha);
- Porque nos métodos deve ser passado como parâmetro um ponteiro para a pilha, e não a pilha diretamente?

Exemplo - PILHA

- Faça um programa que implemente uma pilha de 20 elementos do tipo inteiro utilizando struct

Exemplo - PILHA

```
#include <stdio.h>
#include <stdlib.h>
#define TAMANHO_PILHA 20

typedef struct{
        int vetor[TAMANHO_PILHA]; //VETOR É A PILHA com tamanho = a tamanho da pilha
        int topo;
}Pilha;

int main(int argc, char *argv[]) {
        //DECLARA UMA PILHA
        Pilha p;
        p.topo=0;
        //empilhar numero 45: Faz-se dessa forma pq p foi declarado de forma local
        p.vetor[p.topo]=45;
        p.topo++;
        //desempilhar o ultimo numero
        p.topo--;
        printf("Elemento retirado da pilha: %d \n",p.vetor[p.topo]);
        return 0;
}
```

Exemplo - PILHA

- Faça um programa que implemente uma pilha de 20 elementos do tipo inteiro utilizando struct
  - Implemente um método que empilhe um novo inteiro
    - O método deve enviar uma mensagem se a pilha estiver cheia
    - void empilha(int valor, Pilha *pilha);

Exemplo - PILHA

```
#include <stdio.h>
#include <stdlib.h>
#define TAMANHO_PILHA 20
typedef struct{
        int vetor[TAMANHO_PILHA]; //tamanho da pilha
        int topo;
}Pilha;

//prototipo da função empilha
void empilha(int valor,Pilha *pilha){
        //pilha->topo significa: ponteiro "pilha" apontando para CONTEÚDO de um item de
uma struct
        if(pilha->topo < TAMANHO_PILHA){ //verificando se a pilha não está cheia
        //daí pode empilhar
        pilha->vetor[pilha->topo]=valor;
        pilha->topo++;
        }else{
                printf("Nao ha mais espaco na pilha, \n");
        }
}
```

Exemplo - PILHA

- Porque nos métodos deve ser passado como parâmetro um ponteiro para a pilha, e não a pilha diretamente?

Devido as variáveis locais e globais.

Ao não passar como ponteiro, as funções vão manipular a cópia da pilha, mas não vão manipular a pilha original (da função *main*)

Exemplo - PILHA

- Faça um programa que implemente uma pilha de 20 elementos do tipo inteiro utilizando struct
  - Implemente um método que empilhe um novo inteiro
    - O método deve enviar uma mensagem se a pilha estiver cheia
    - void empilha(int valor, Pilha *pilha);
  - Implemente um método que desempilhe
    - O método deve exibir uma mensagem se a pilha estiver vazia
    - void desempilha(Pilha *pilha);

Exemplo - PILHA

```
void desempilha(Pilha *pilha){
        if(pilha->topo > 0){
                pilha->topo--; //desempilha
                printf("Elemento retirado: %d. \n",pilha->vetor[pilha->topo]);
        }else{
                printf("A pilha está vazia. \n"); //pilha vazia
        }
}
```

Exemplo - PILHA

- Faça um programa que implemente uma pilha de 20 elementos do tipo inteiro utilizando struct
  - Implemente um método que empilhe um novo inteiro
    - O método deve enviar uma mensagem se a pilha estiver cheia
    - void empilha(int valor, Pilha *pilha);
  - Implemente um método que desempilhe
    - O método deve exibir uma mensagem se a pilha estiver vazia
    - void desempilha(Pilha *pilha);
  - Implemente um método que retorne 1 se a pilha está cheia e 0 se não
    - int isCheia(Pilha *pilha);
  - Implemente um método que retorne 1 se a pilha está vazia e 0 se não
    - int isVazia(Pilha *pilha);

Exemplo - PILHA

```
int isCheia(Pilha *pilha){
        if(pilha->topo >= TAMANHO_PILHA){
                return 1;               //identifica se a pilha está cheia
        }else{
                return 0;
        }
}
```

Exemplo - PILHA

```
int isVazia(Pilha *pilha){
        if(pilha->topo == 0){
                return 1;               //identifica se a pilha está vazia
        }else{
                return 0;
        }
}
```

Exemplo - PILHA

INSERINDO ITENS NA PILHA E VERIFICANDO A POSIÇÃO QUE ESTÁ NO TOPO.

```
int main(int argc, char *argv[]) {
        //DECLARA UMA PILHA
        Pilha p;
        p.topo=0; // o topo da pilha deve começar em zero

        empilha(12,&p);    //elemento 1
        empilha(2,&p);     //elemento 2
        empilha(6,&p);     //elemento 3
        empilha(9,&p);     //elemento 4
        empilha(50,&p);    //elemento 5

        printf("Topo da pilha: %d. \n",p.topo);
        return 0;
}
```

Exemplo - PILHA

IMPLEMENTAÇÃO FUNÇÃO IMPRIMIR PILHA:

```
void imprimePilha(Pilha *pilha){
        int i;
        for(i=(pilha->topo);i--;i>=0){ //valor inicial de i é a ultima posição da pilha e daí
decrementa
                printf("%02d \n",pilha->vetor[i]);
        }
}
```

Exemplo - PILHA

TESTANDO A FUNÇÃO imprimePilha:

```
int main(int argc, char *argv[]) {
        //DECLARA UMA PILHA
        Pilha p;
        p.topo=0; // o topo da pilha deve começar em zero

        empilha(12,&p);     //elemento 1
        empilha(2,&p);      //elemento 2
        empilha(6,&p);      //elemento 3
        empilha(9,&p);      //elemento 4
        empilha(50,&p);     //elemento 5

        printf("Topo da pilha: %d. \n",p.topo);

        imprimePilha(&p);

        return 0;
}
```

Exemplo - PILHA

TESTANDO A FUNÇÃO desempilha:

```
int main(int argc, char *argv[]) {
        //DECLARA UMA PILHA
        Pilha p;
        p.topo=0; // o topo da pilha deve começar em zero

        empilha(12,&p);     //elemento 1
        empilha(2,&p);      //elemento 2
        empilha(6,&p);      //elemento 3
        empilha(9,&p);      //elemento 4
        empilha(50,&p);     //elemento 5

        desempilha(&p);

        printf("Topo da pilha: %d. \n",p.topo);

        imprimePilha(&p);
        return 0;
}
```

Exercício - PILHA

- Com base no exercício 01, adicione um menu para que o usuário possa escolher se quer adicionar um elemento à pilha, ou remover um elemento
  - Se pressionar 'a', o programa solicita um número a ser adicionado à pilha
  - Se pressionar 'r', o programa remove um elemento da pilha
  - Se pressionar 's', o programa exibe os elementos já presentes na pilha
  - Se pressionar 't', o programa exibe o número de elementos
  - Se pressionar 'x', encerra o programa

```
Escolha uma opção:
a: adiciona elemento
r: remove elemento
s: exibe elementos
...
```

Menu em C

```
char c = 'a';

while(c!='x')      // se o usuario pressionar 'x', encerra o laço do menu
{
        printf("Digite um caractere:\n");
        scanf("%c",&c);                    // captura caractere pressionado

        switch(c)
        {
                case 'a':
                        ...
                break;

                case 'x':
                        ...
                break;

                default:
                        ...
                break;

        }
}
```

Exemplo - PILHA

```c
int main(int argc, char *argv[]) {

    //
    char c = 'a';   // guarda o caractere (opcao) do usuario

// declara uma pilha
Pilha p;
p.topo = 0;

// menu infinito
while(c!='x')   // se o usuario pressionar 'x', encerra o laço do menu
{
    printf("Digite um caractere:\n");
    //c = (char)getchar();  // captura caractere pressionado
    scanf("%c",&c);

    switch(c)
    {
        case 'a':
            printf("Empilhar\n");
            // TODO
        break;

        case 'x':
            printf("Sair\n");
            // TODO
        break;

        case 's':
            // TODO
            break;

        default:
            // TODO
        break;

    }
}
```