***Tecnologia em Análise e Desenvolvimento de Sistemas - TADS***

# *Estrutura de Dados I*

*Atividades Pedagógicas Não Presenciais – APNP 2020*

*Prof.* ***Luciano Vargas Gonçalves***
*E-mail: luciano.goncalves@riogrande.ifrs.edu.br*

tads
Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

# Estrutura da Dados

- **Aula 5 – Estruturas Dinâmicas**
  - **Pilhas**

# Sumário

- **Estrutura de Dados**
  - **Dinâmicas:**
    - O tamanho se altera com a necessidade;
    - Cresce ou Decresce
      - Listas
        - Simplesmente Encadeadas
      - Pilhas

# Pilha

- **Características**
  - Estrutura de dados para armazenar informações;
  - Semelhante a uma lista mais com restrições;
  - Muito utilizada para controlar sistema de montagem e desmontagem;
  - Exemplos

Pilha de Livros

Pilha de Roupas

# Pilha

- **Características**
  - Quais Restrições??
  - Podemos pegar qualquer peças??
    - Pegamos o prato que esta no topo!!!!
    - Colocamos de volta o prato no topo!!!

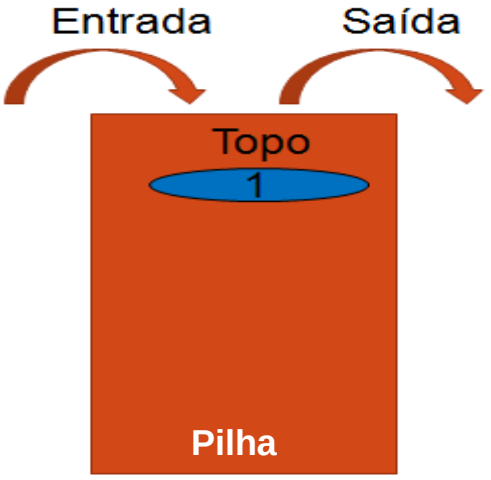
# Pilha

- **Características**
  - Quais Restrições??
  - Podemos pegar qualquer Carta no Jogo Cartas??
  - Regras!!!!
  - Pegar somente a que estiver no topo!!!

Jogo De Cartas

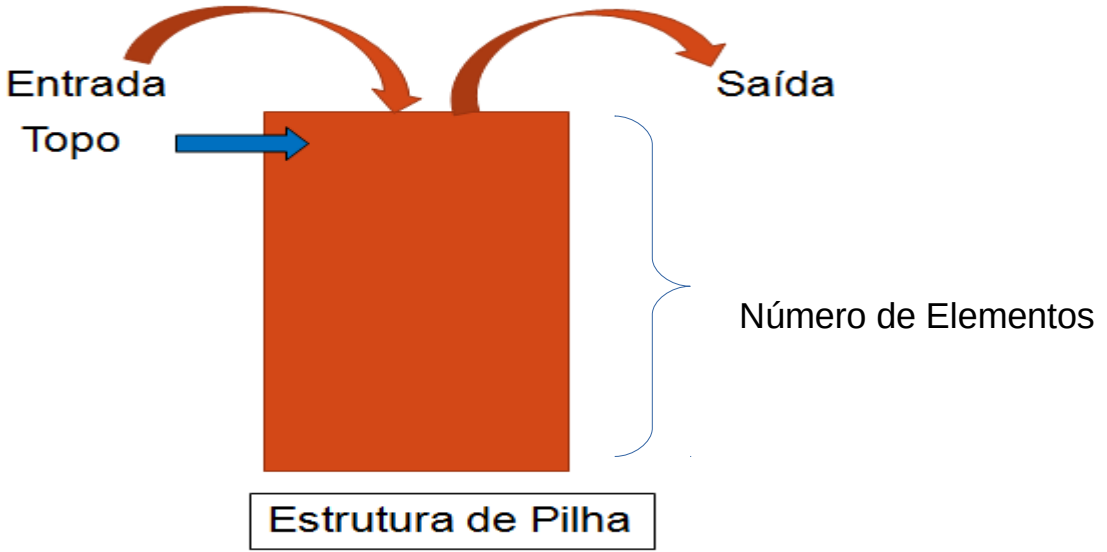
# Pilha

- Quais Restrições?? Estrutura de Pilha?
  - Só podemos retirar o elemento da pilha que está no topo. O elemento que está diretamente acessível;
  - Após a retirada do elemento que está no topo, o segundo elemento se torna acessível (Topo);
  - A inserção só pode ocorrer no topo da pilha;
  - Apresentar a situação pilha vazia;
  - Inserção e Remoção pela mesma extremidade.



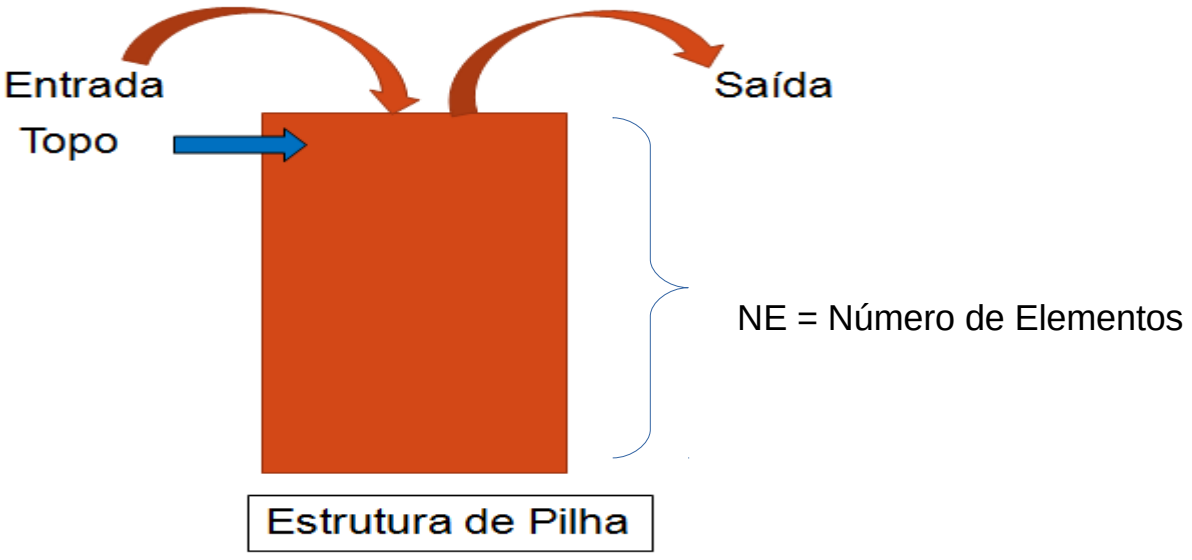
# Pilha

- Estrutura de Pilha:
  - Entrada e saída pelo mesma extremidade (Topo);
  - Topo é um apontador ou ponteiro para um elemento;



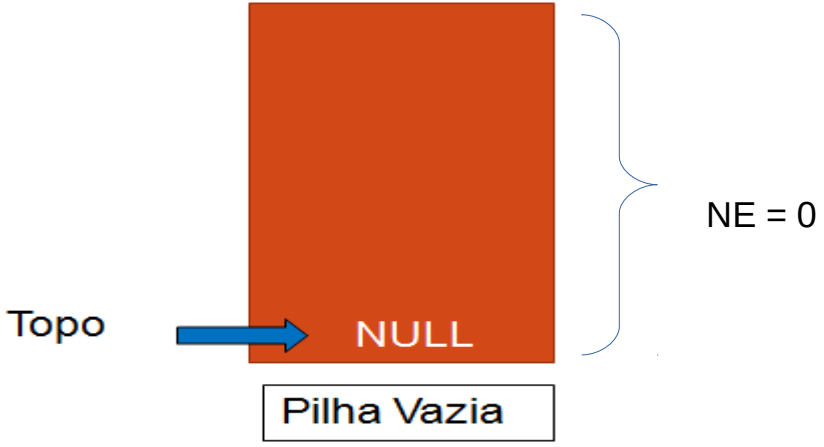
# Pilha

- Estrutura de Pilha:
  - LIFO – Last IN Firt Out  (o último a entrar é o primeiro a sair)



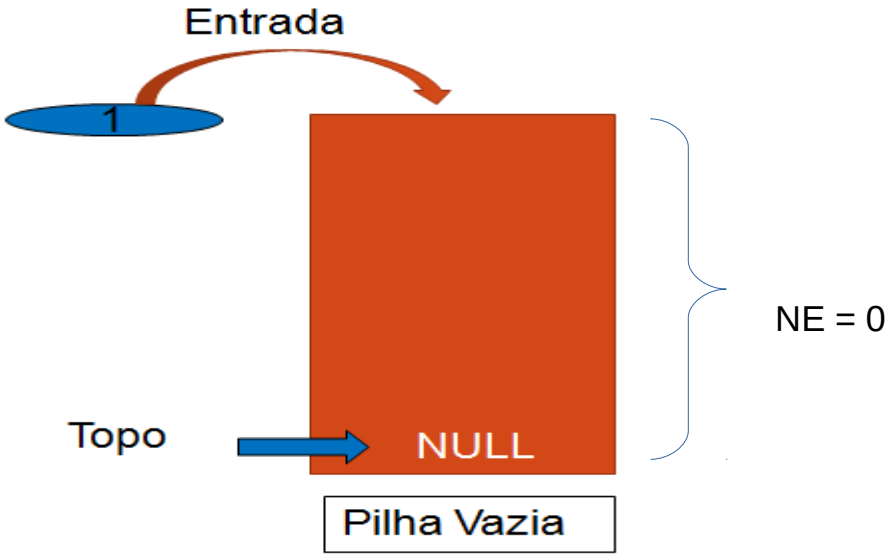
# Pilha

- Estrutura de Pilha:
  - LIFO – Last IN Firt Out  (o último a entrar é o primeiro a sair)
  - Pilha vazia – Topo aponta para NULL;
  - Número de Elementos é igual a zero;



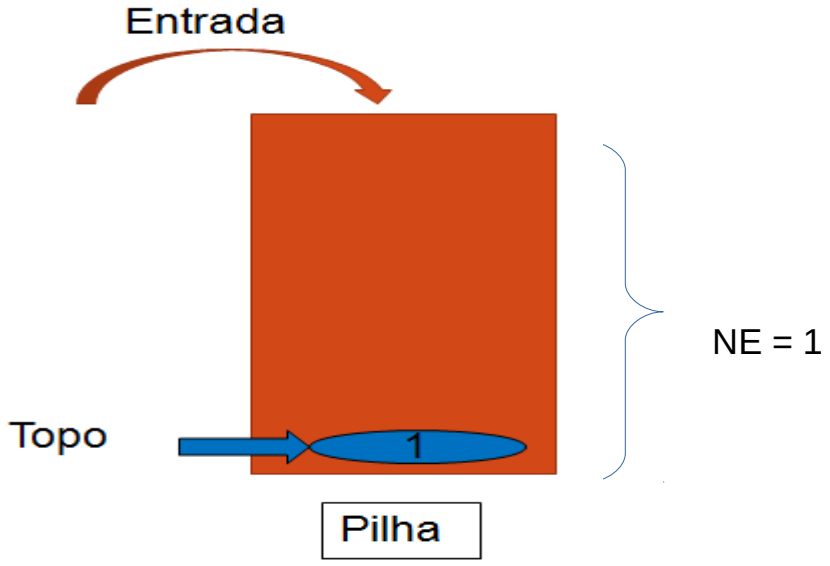
# Pilha

- Estrutura de Pilha:
  - Inserir Elemento - Comando PUSH;
  - Número de Elementos é igual a zero;



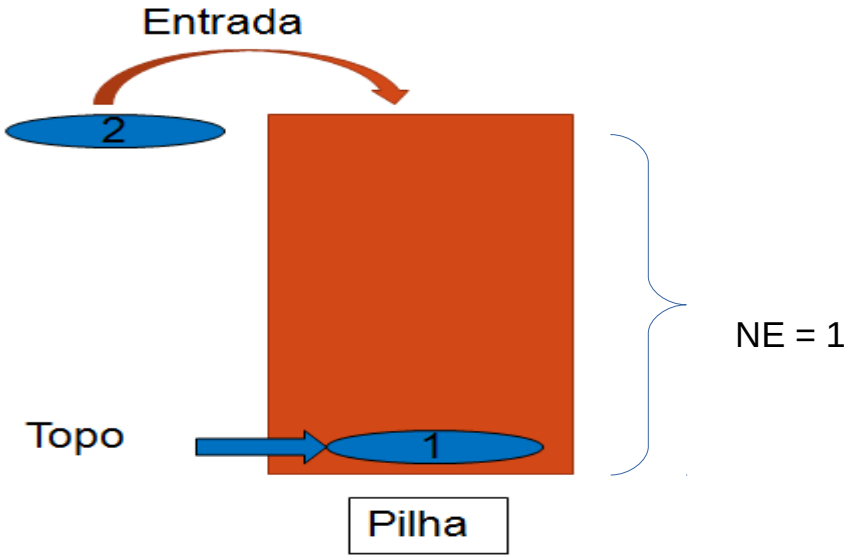
# Pilha

- Estrutura de Pilha:
  - Inserir Elemento - Comando PUSH(empurrar);
  - Número de Elementos é igual a 1;



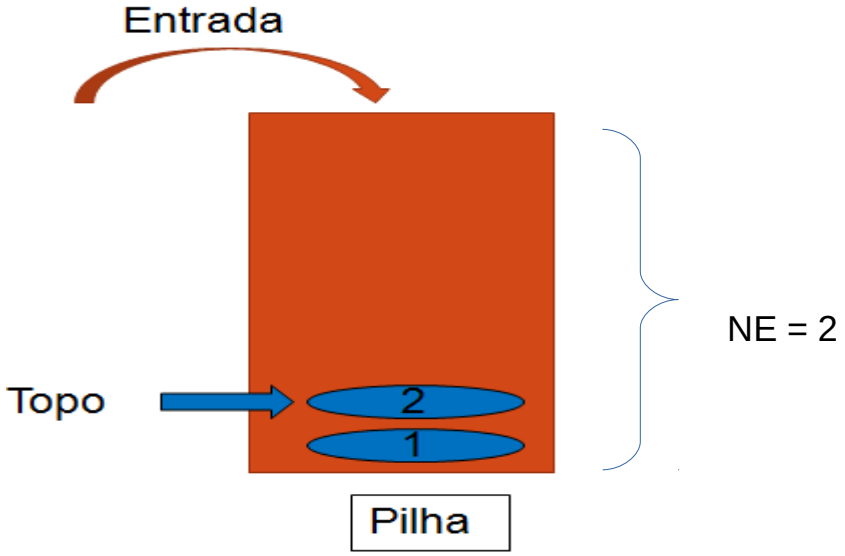
# Pilha

- Estrutura de Pilha:
  - Inserir Elemento 2 - Comando PUSH;
  - Número de Elementos é igual a 1;



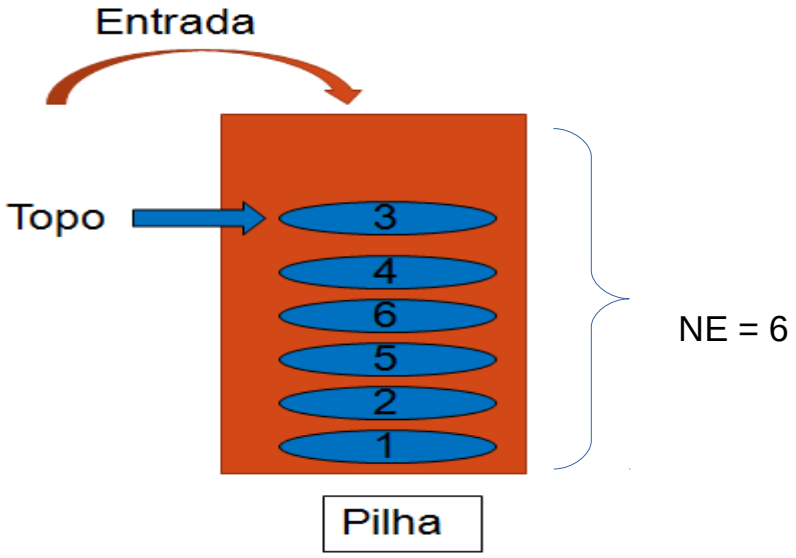
# Pilha

- Estrutura de Pilha:
  - Inserir Elemento 2 - Comando PUSH;
  - Número de Elementos é igual a 2;



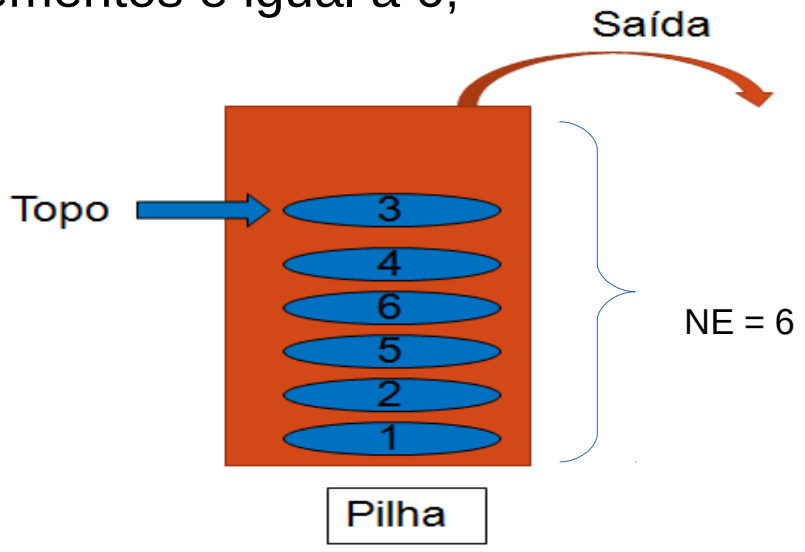
# Pilha

- Estrutura de Pilha:
  - Inserir Elemento 4 - Comando PUSH;
  - Número de Elementos é igual a 6;



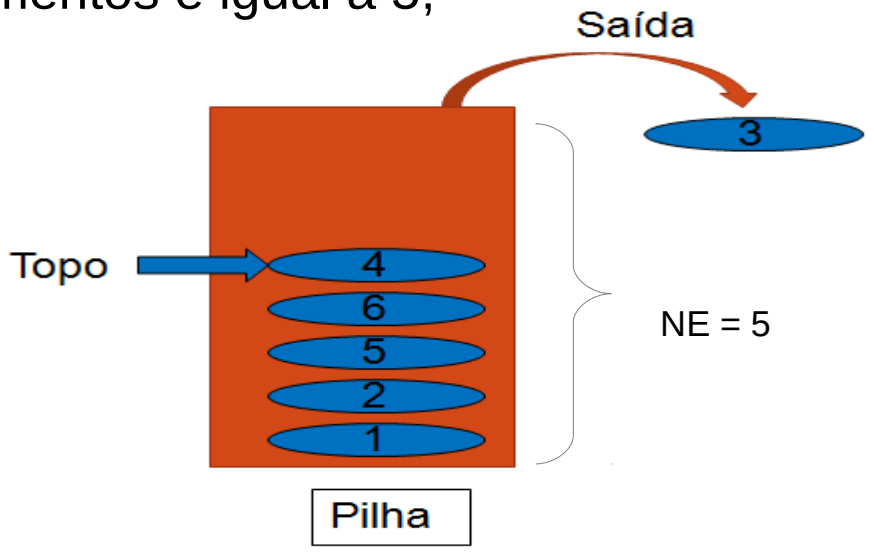
# Pilha

- Estrutura de Pilha (POP):
  - Remover um Elemento - Comando POP (remover par fora);
  - Número de Elementos é igual a 6;



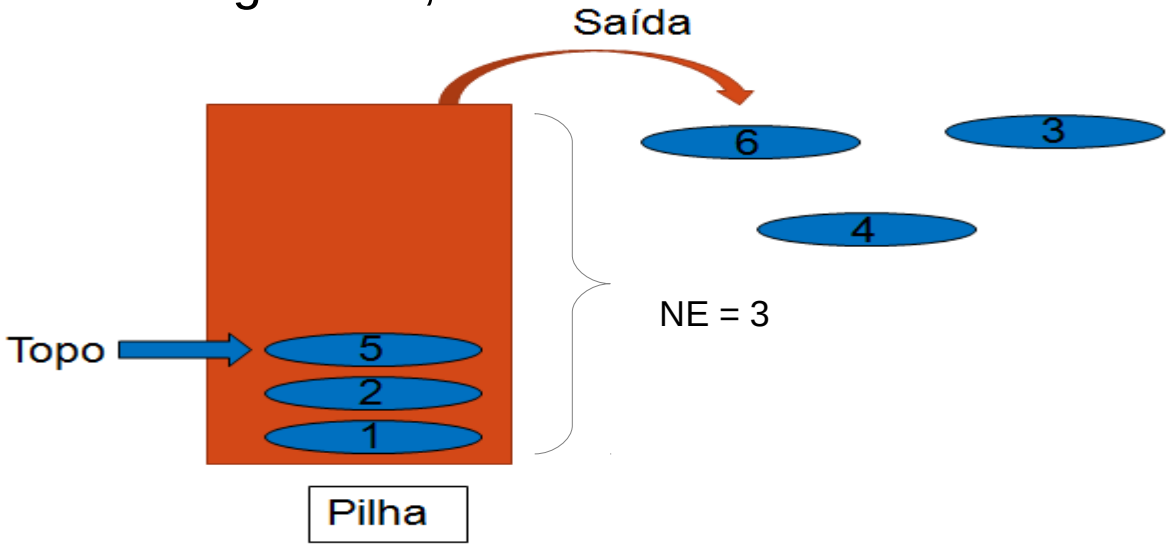
# Pilha

- Estrutura de Pilha (POP):
  - Remover um Elemento - Comando POP;
  - Número de Elementos é igual a 5;



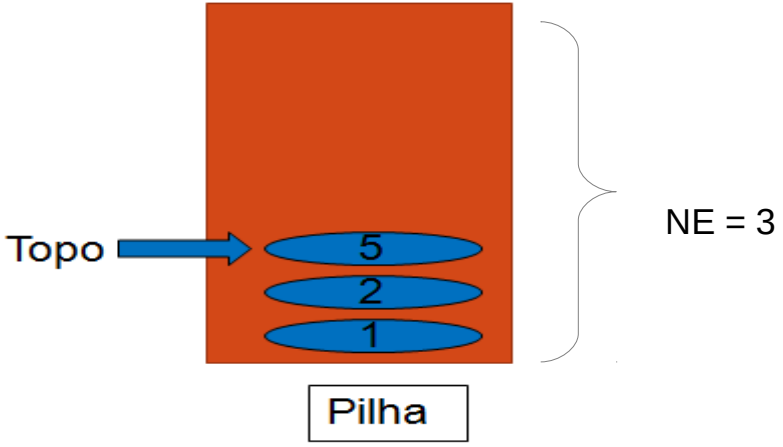
# Pilha

- Estrutura de Pilha (POP):
  - Remover 2 Elementos - Comando POP;
  - Número de Elementos é igual a 3;

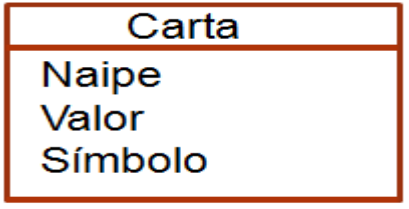


# Pilha

- Estrutura de Pilha (Consultar um Elemento):
  - Consultar um Elemento apenas o que está no TOPO (5);
  - Consultar os demais elementos, somente desempilhando os elementos até atingir o elemento desejado



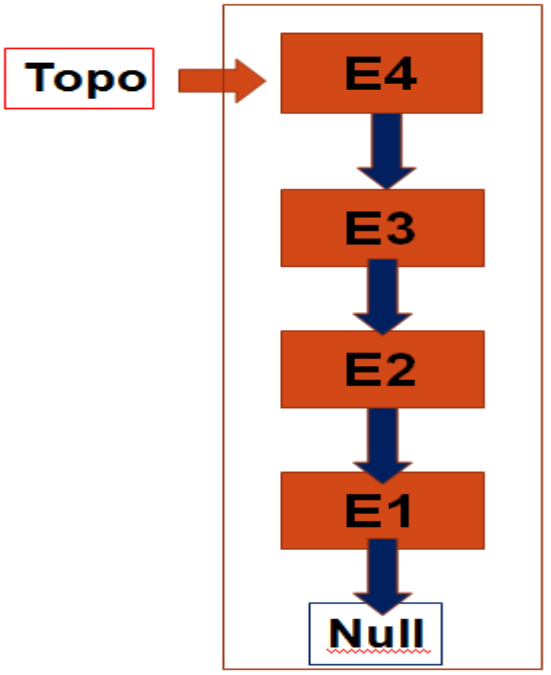
# Pilha

- **Elemento de uma Pilha**
  - Será uma estrutura para armazenar informações desse elemento em uma pilha;
    - Pilha de pratos o elemento é o Prato;
    - Pilha de livros o elemento é o livro;
    - Pilha de Cartas o elemento será uma carta.
  - 

Elemento Carta

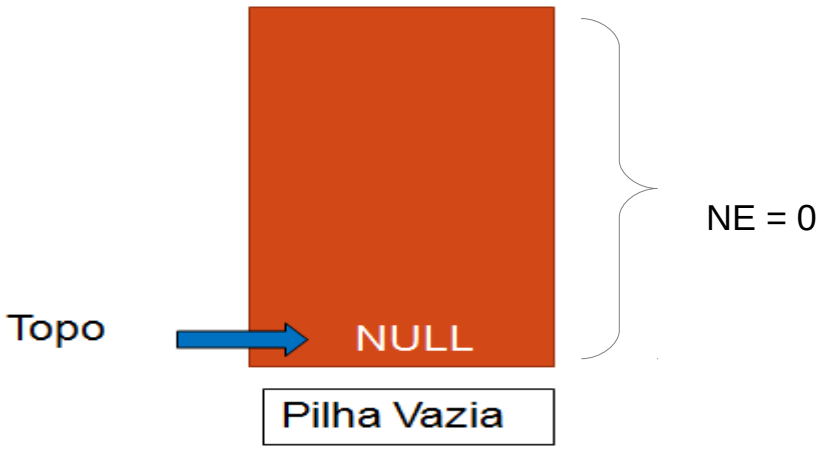
# Pilha

- **Pilha**
  - Pilhas nada mais são do que listas lineares Simplesmente encadeada com restrições:
    - Insere no Início (topo)
    - Remove no Início (topo)
    - Consulta primeiro elemento (topo)




Pilha

# Pilha

- **Estrutura de Pilha de Cartas:**
  - Estrutura da Pilha:
    - Precisamos de uma apontador Topo;
    - Contador de elementos;
  - 


Estrutura para um Pilha de Cartas

```
struct Pilha{
    Carta *topo;
    int n_elementos;
};
```

# Pilha

- **Estrutura de Pilha de Cartas:**
  - Para apontar próximo elemento iremos criar uma apontador que irá armazenar uma referência para o “próximo” elemento (carta);


Elemento Carta

Estrutura do Elemento CARTA em C

```
struct Elemento{
    char naipe [20];
    char simbolo;
    int valor;
    Carta *proximo;
};
```

# Pilha

- **Inicialização da Pilha de Cartas:**
  - Topo aponta para NULL
  - Número de Cartas igual a Zero
  - *p é o parâmetro(ponteiro) que recebe o endereço da pilha;

```
Pilha * criaPilha(){
    Pilha *p_nova = (Pilha *)malloc(sizeof(Pilha ));
    p_nova->n_elementos = 0;
    p_nova->topo = NULL;
}
```

# Pilha

- **Função Cria Carta!!**
  - Retorna um ponteiro para carta preenchida;

```
Carta * criaCarta(char naipe [], char simbolo, int valor){
    Carta *nova = (Carta *) malloc (sizeof(Carta));

    if(!nova){
        printf("Sem memoria disponivel!\n");
        exit(1);
    }

    strcpy(nova->naipe,naipe);
    nova->simbolo = simbolo;
    nova->valor = valor;
    nova->proximo = NULL;
    return nova;
}
```

# Pilha

- **Função Insere Carta na Pilha (PUSH)!!**
  - Insere uma carta na Pilha;

```
int pushPilha (Pilha *p, Carta *e){

    if(p->topo != NULL){
        e->proximo = p->topo;
    }else
        e->proximo = NULL;
     p->topo = e;
     p->n_elementos++;
     return 1;
}
```

# Pilha

- **Função Remove um Carta da Pilha (POP)!!**
  - Retorna um ponteiro para uma Carta

```
Carta * popPilha (Pilha *p){
    Carta *aux = NULL;
    if(p->topo != NULL){
        aux = p->topo;
        p->topo = p->topo->proximo;
        p->n_elementos--;
    }/*else{
        printf("\n    Pilha Vazia!! \n  ");
    }*/
    return aux;
}
```

# Pilha

- **Função Mostra Pilha !!**
  - Mostra os Elementos da Pilha
    - Precisa remover da pilha e armazenar em outra pilha
    - Mostrar o elemento (Carta);
    - Após mostrar todos elementos, retornar os elementos a pilha original;

# Tarefas

- Outros procedimentos e Funções para implementar

```
void embaralha(pilha *p){
    // Recebe uma pilha de cartas e embaralha aleatoriamente

}


int ordenaNaipe(pilha *p){
    // recebe uma pilha de cartas de um NAIPE e Ordena NAIPE
}

int invertePilha(pilha *p){
    //recebe um pilha de cartas e inverte a pilha
}
```