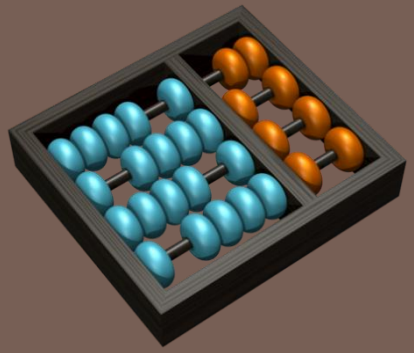


UNICAMP

PILHAS

Prof.: Julio Cesar dos Reis

Roteiro

- Conceito de pilha
- Implementação com arranjos
- Implementação com listas ligadas

Motivação

- Listas lineares não restringem a disciplina de acesso aos elementos
- Inadequadas quando aplicações precisam preservar a ordem de entrada e saída
- Necessidade de estruturas lineares com políticas de operações mais específicas

Pilhas (*Stack*)

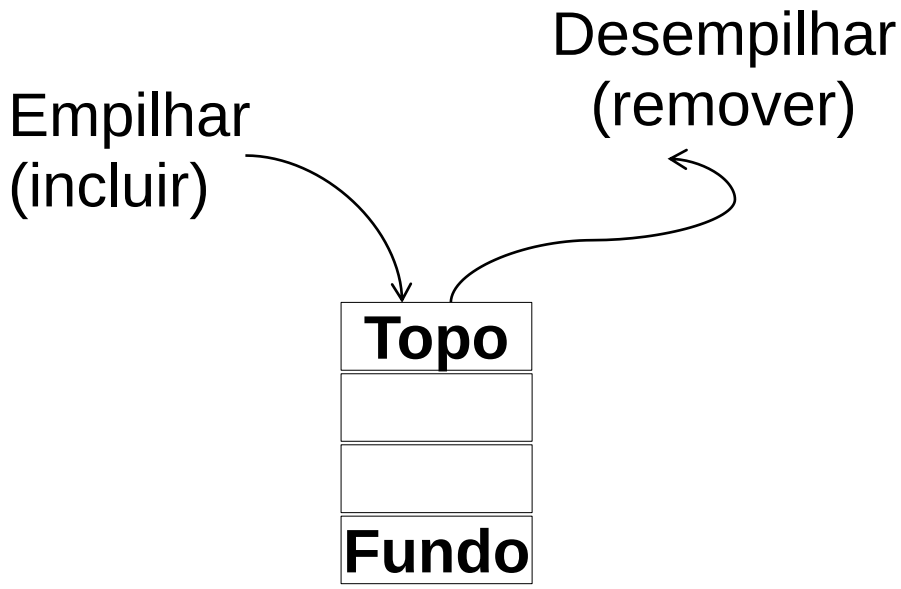
- Primeiro a entrar é o último a sair
- Remove primeiro elementos inseridos há menos tempo

Pilhas (*Stack*)

- Primeiro a entrar é o último a sair
- Remove primeiro elementos inseridos há menos tempo
- Inserção e remoção se faz pela mesma extremidade (topo da pilha)
  - Utiliza-se a mesma lógica de uma pilha de papéis

TAD Pilha

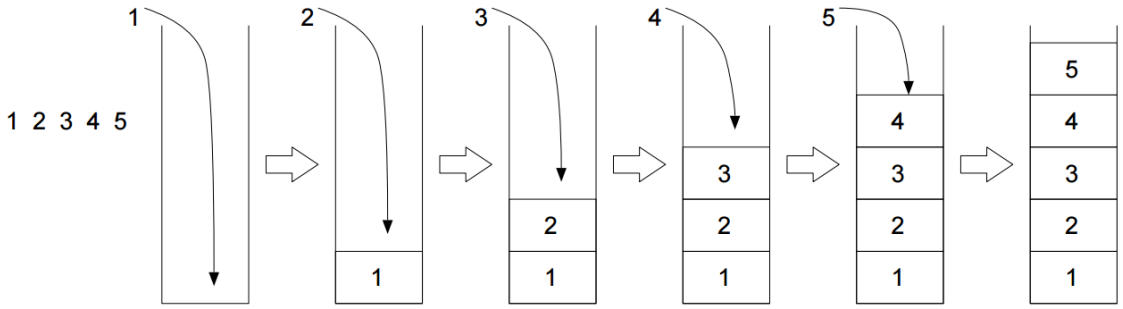
- Operações



- empilhar
- desempilhar
- pilhaVazia
- pilhaCheia
- topoPilha
- tamanho

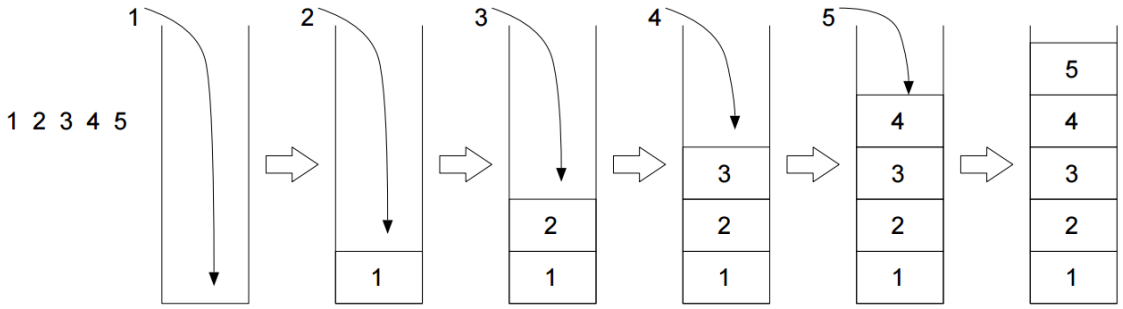
Exemplo de operações em pilha

- Sequência de empilhamento

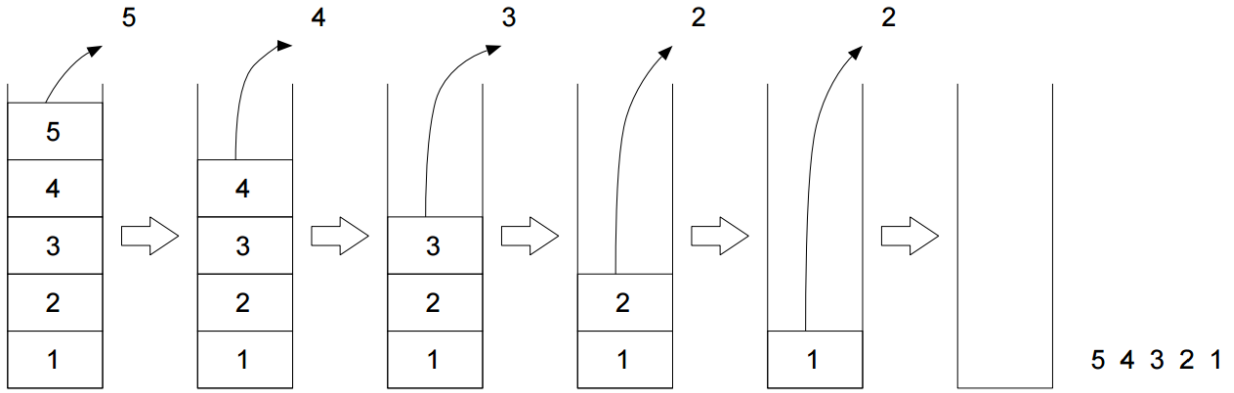


Exemplo de operações em pilha

- Sequência de empilhamento



- Sequência de desempilhamento

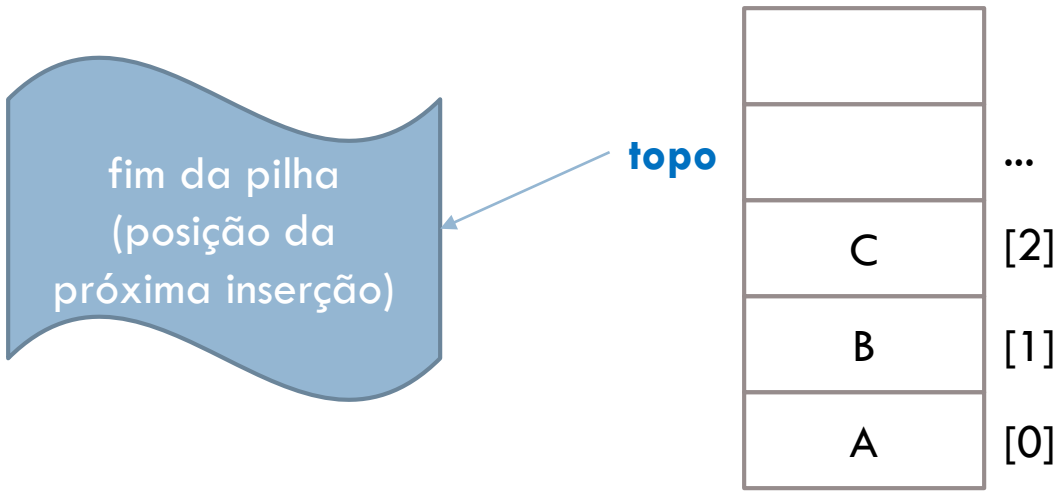


Abordagens de implementação

- Sequencial com uso de arranjos
- Encadeada usando uma estrutura ligada

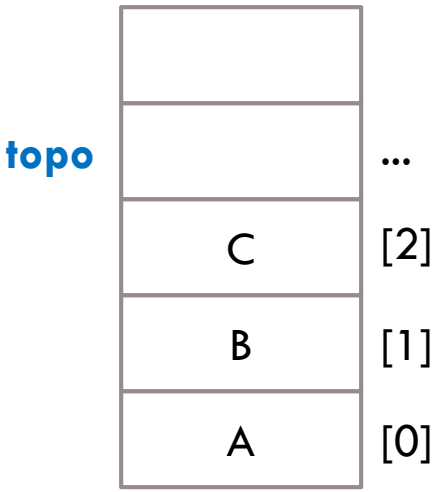
Implementação com arranjos

- Uso de vetor
  - Tamanho máximo do vetor (*n* elementos)
  - Índice de **topo** da pilha



Implementação com arranjos

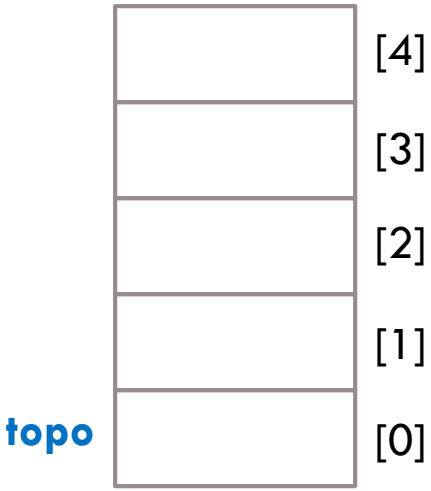
- Uso de vetor
  - Tamanho máximo do vetor (*n* elementos)
  - Índice de **topo** da pilha
    - Pilha está cheia se topo == *n*
    - Pilha está vazia se topo == 0



Exemplo com Pilha (vetor)

- Pilha de tamanho máximo igual a 5

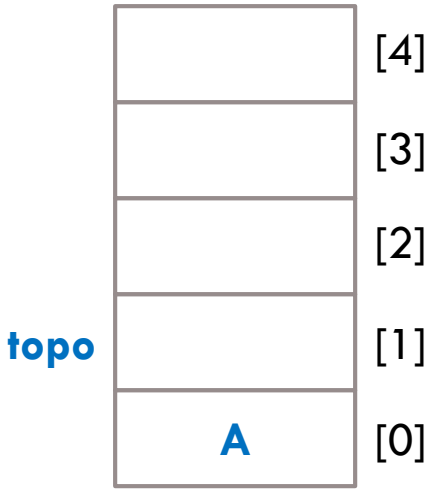
- empilhar(A)
- empilhar(B)
- empilhar(C)
- desempilhar()
- empilhar(D)
- empilhar(E)
- desempilhar()
- empilhar(F)
- empilhar(G)



Exemplo com Pilha (vetor)

- Pilha de tamanho máximo igual a 5

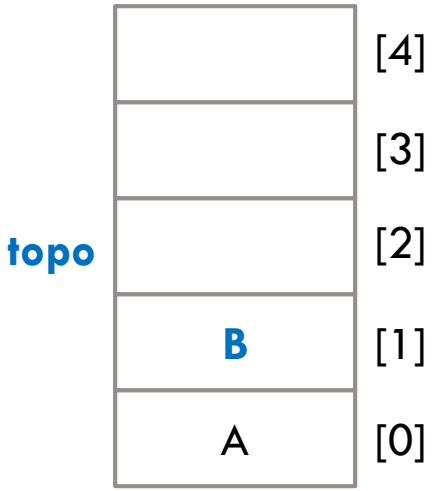
- **empilhar(A)**
- empilhar(B)
- empilhar(C)
- desempilhar()
- empilhar(D)
- empilhar(E)
- desempilhar()
- empilhar(F)
- empilhar(G)



Exemplo com Pilha (vetor)

- Pilha de tamanho máximo igual a 5

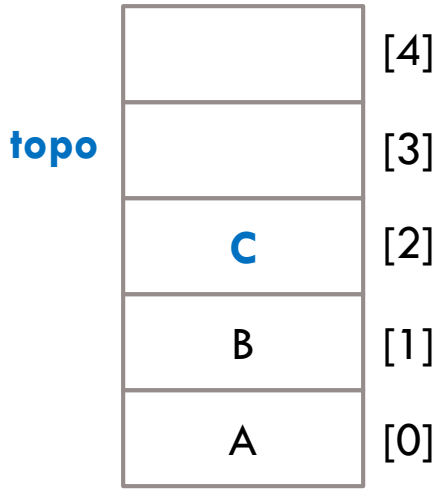
- empilhar(A)
- **empilhar(B)**
- empilhar(C)
- desempilhar()
- empilhar(D)
- empilhar(E)
- desempilhar()
- empilhar(F)
- empilhar(G)



Exemplo com Pilha (vetor)

- Pilha de tamanho máximo igual a 5

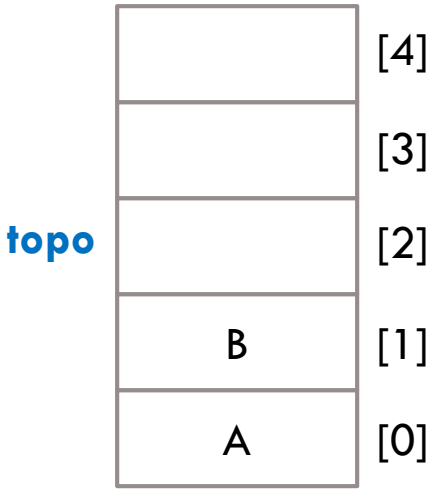
- empilhar(A)
- empilhar(B)
- **empilhar(C)**
- desempilhar()
- empilhar(D)
- empilhar(E)
- desempilhar()
- empilhar(F)
- empilhar(G)



Exemplo com Pilha (vetor)

- Pilha de tamanho máximo igual a 5

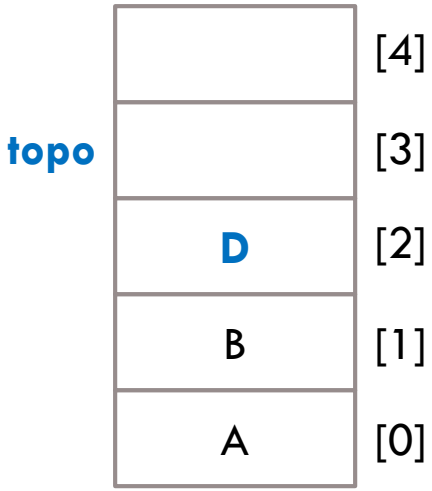
- empilhar(A)
- empilhar(B)
- empilhar(C)
- **desempilhar()**
- empilhar(D)
- empilhar(E)
- desempilhar()
- empilhar(F)
- empilhar(G)



Exemplo com Pilha (vetor)

- Pilha de tamanho máximo igual a 5

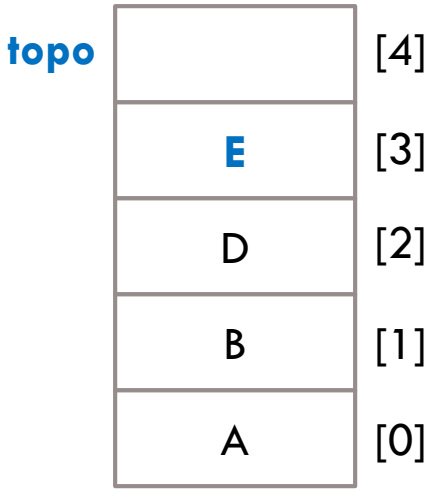
- empilhar(A)
- empilhar(B)
- empilhar(C)
- desempilhar()
- **empilhar(D)**
- empilhar(E)
- desempilhar()
- empilhar(F)
- empilhar(G)



Exemplo com Pilha (vetor)

- Pilha de tamanho máximo igual a 5

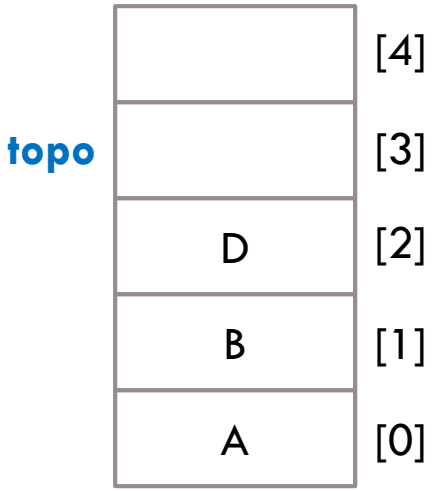
- empilhar(A)
- empilhar(B)
- empilhar(C)
- desempilhar()
- empilhar(D)
- **empilhar(E)**
- desempilhar()
- empilhar(F)
- empilhar(G)



Exemplo com Pilha (vetor)

- Pilha de tamanho máximo igual a 5

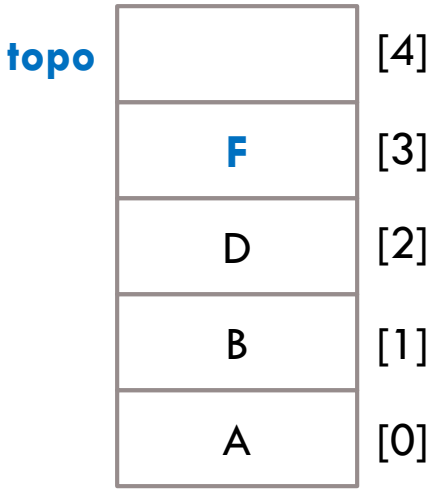
- empilhar(A)
- empilhar(B)
- empilhar(C)
- desempilhar()
- empilhar(D)
- empilhar(E)
- **desempilhar()**
- empilhar(F)
- empilhar(G)



Exemplo com Pilha (vetor)

- Pilha de tamanho máximo igual a 5

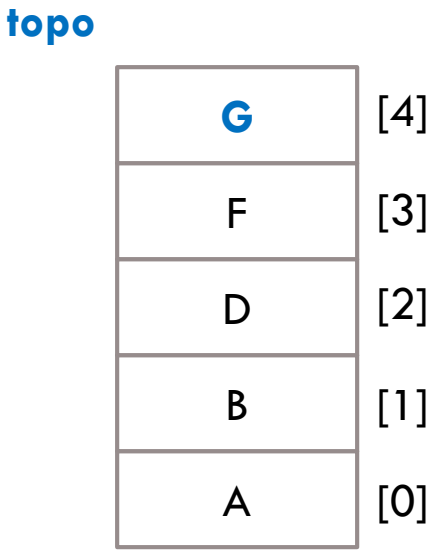
- empilhar(A)
- empilhar(B)
- empilhar(C)
- desempilhar()
- empilhar(D)
- empilhar(E)
- desempilhar()
- **empilhar(F)**
- empilhar(G)



Exemplo com Pilha (vetor)

- Pilha de tamanho máximo igual a 5

- empilhar(A)
- empilhar(B)
- empilhar(C)
- desempilhar()
- empilhar(D)
- empilhar(E)
- desempilhar()
- empilhar(F)
- **empilhar(G)**



Estrutura da pilha com arranjo


```
typedef struct Pilha {
        int *elems; //alocação de vetor para armazenar os dados
        //int elems[T_MAX]  [#define T_MAX 50;]
        int topo; //índice de fim da pilha
} Pilha;
```

Operações da Pilha (com vetor)

- Inicialização

```
Pilha* inicializarPilha(int tam){
    Pilha pilha, *p; p=&pilha;
    p->elems=malloc(sizeof(int*tam));
    p->topo=0;
    return p;
}
```

Operações da Pilha (com vetor)

- Inserção

```
bool empilhar(Pilha *p, int valor){
  p->elems[p->topo] = valor;  //verificar se pilha cheia?
  (p->topo)++;
  return true; }
```

Operações da Pilha (com vetor)

- Inserção

```
bool empilhar(Pilha *p, int valor){
  p->elems[p->topo] = valor;
  (p->topo)++;
  return true; }
```

```
bool empilhar(Pilha *p, int valor){
  if (p->topo!=p->qtd_max){
     p->elems[p->topo] = valor;
     (p->topo)++;
     return true; }
}
```

Operações da Pilha (com vetor)

- Inserção

```
bool empilhar(Pilha *p, int valor){
  p->elems[p->topo] = valor;
  (p->topo)++;
  return true; }
```

```
bool empilhar(Pilha *p, int valor){
  if (p->topo!=p->qtd_max){
    p->elems[p->topo] = valor;
    (p->topo)++;
    return true; }
}
```

```
typedef struct Pilha {
        int *elems;
        int topo;
        int qtd_max;
} Pilha;
```

inicialização

```
p->qtd_max=tam;
```

Operações da Pilha (com vetor)

- Remoção

```
int desempilhar(Pilha *p){
   (p->topo)--; //verificar se pilha vazia?
    return p->elems[p->topo];
}
```

Operações da Pilha (com vetor)

- Remoção

```
int desempilhar(Pilha *p){
   (p->topo)--;
    return p->elems[p->topo];
}
```

```
int desempilhar(Pilha *p){
  if (p->topo!=0){ //podemos ter uma função pilhaVazia
     (p->topo)--;
     return p->elems[p->topo];
  }
}
```

Operações da Pilha (com vetor)

- Elemento do topo

```
int itemTopo(Pilha *p){  return p->elems[p->topo-1]; }
```

- Pilha Cheia

```
bool pilhaCheia(Pilha *p, int T_MAX){
    if (p->topo==p->qtd_max) return true; else return false;
}
```

- Pilha Vazia

```
bool pilhaVazia(Pilha *p){
   if (p->topo==0) return true; else return false;
}
```

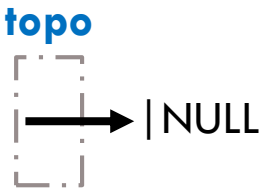
Implementação com estruturas ligadas

- Uso de nós em uma lista ligada
- Não necessitam de uma porção de memória contígua alocada a priori
- Alocação de elementos feita sob demanda

Exemplo de Pilha (com lista)

- Pilha vazia então *topo* aponta para *NULL*

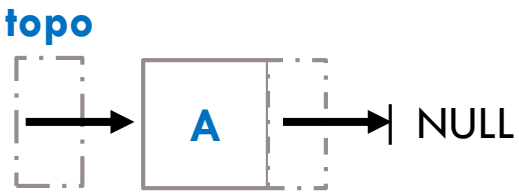
- empilhar(A)
- empilhar(B)
- empilhar(C)
- desempilhar()



Exemplo de Pilha (com lista)

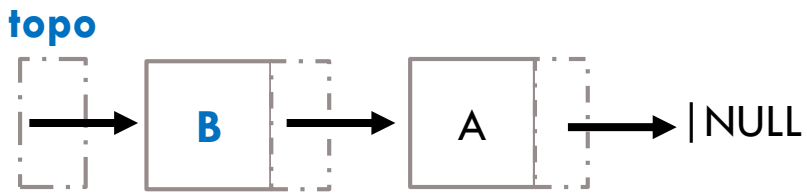
- Empilhamento deve inserir elemento no início da lista

- **empilhar(A)**
- empilhar(B)
- empilhar(C)
- desempilhar()



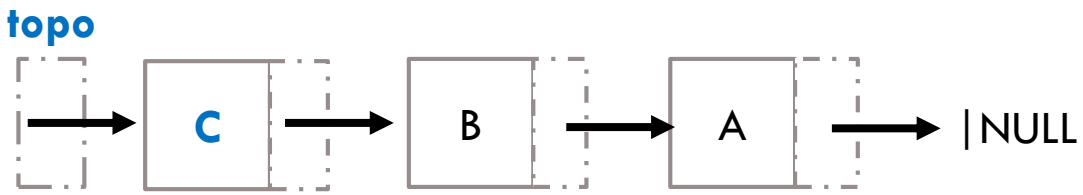
Exemplo de Pilha (com lista)

- empilhar(A)
- **empilhar(B)**
- empilhar(C)
- desempilhar()



Exemplo de Pilha (com lista)

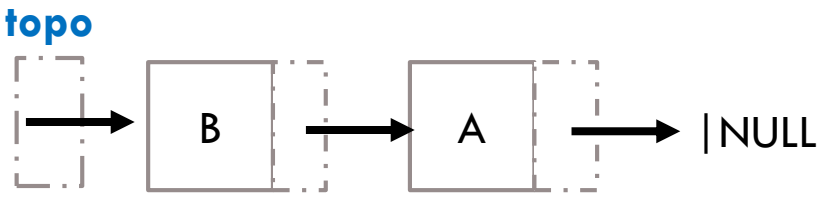
- empilhar(A)
- empilhar(B)
- **empilhar(C)**
- desempilhar()



Exemplo de Pilha (com lista)

- Desempilhamento remove elemento do inicio da lista

- empilhar(A)
- empilhar(B)
- empilhar(C)
- **desempilhar()**



Estrutura da Pilha (com lista)

Elemento nó

```
typedef struct NoPilha{
        int chave;
        struct NoPilha *prox;
} NoPilha;
```

Pilha

```
typedef struct Pilha {
    NoPilha *topo;
} Pilha;
```

Operações da Pilha (com lista)

- Inserção (empilhamento)

```
bool empilhar(Pilha *p, int valor){
    NoPilha* novo=malloc(sizeof(NoPilha));
    if (novo!=NULL){
       novo->chave=valor;
       novo->prox=p->topo; //inserção no inicio
       p->topo=novo;
       return true;
    } return false;
}
```

Operações da Pilha (com lista)

- Remoção (desempilhamento)

```
int desempilhar(Pilha *p){
    NoPilha *aux = p->topo;
    if (aux==NULL) return -1; //pilha vazia
    int chave = aux->chave;
    p->topo=p->topo->prox; //caminha na lista
    free(aux);
    return chave;
}
```

Operações da Pilha (com lista)

- Elemento do topo

```
int itemTopo(Pilha *p){
    if (p!=NULL) //poderia verificar se pilha não vazia
            return p->topo->chave;
}
```

- Pilha Vazia

```
bool pilhaVazia(Pilha *p){
    if (p!=NULL && p->topo==NULL)
            return true;
    else
            return false;
}
```

Uso de duas pilhas

- Um professor gostaria de organizar a correção de provas
  - Necessita cadastrar um conjunto de provas à corrigir
  - A cada prova corrigida, um subconjunto de provas corrigidas é criado

Uso de duas pilhas

- Um professor gostaria de organizar a correção de provas
  - Necessita cadastrar um conjunto de provas à corrigir
  - A cada prova corrigida, um subconjunto de provas corrigidas é criado

- Desenvolva um programa que auxilie o professor a gerenciar a correção das provas
  - Use pilhas

Uso de duas pilhas


```
int main(){
   Pilha pr_n_corrigidas, pr_corrigidas;
   Pilha *p_n, *p_c;
   p_n=&pr_n_corrigidas; p_c=&pr_corrigidas;
   int prova;
   //cadastrado de provas não corrigidas
   do{
      println("Digite o código da prova: "); scanf(%d, &prova);
      if (prova!=-1)
         empilhar(p_n, prova);
   } while(prova!=-1)
   (....)
```

Uso de duas pilhas


```
    //corrigindo provas
    while(!pilhaVazia(p_n)){
        empilhar(p_c, desempilhar(p_n));
    }
    //apresentando provas corrigidas
    while(!pilhaVazia(p_c)){
        printf("Prova corrigida %d\n", desempilhar(p_c));
    }
  return(0);
}
```

Exercício

- Resolva o problema de correção de provas utilizando uma pilha implementada com um arranjo de tamanho predefinido
  - Apenas um único vetor pode ser utilizado

Exercício

- Resolva o problema de correção de provas utilizando uma pilha implementada com um arranjo de tamanho predefinido
  - Apenas um único vetor pode ser utilizado
  - Dicas
    - Organize uma pilha em cada extremidade do arranjo
    - Defina índices para controlar a posição do elemento que está no topo das pilhas

Exemplos de aplicação de pilhas

- Cálculo de expressões matemáticas
- Meio de implementar recursão
- Percurso em estruturas de dados avançadas (e.g., grafos)

Síntese

- Pilha é uma estrutura de dados linear com disciplina de acesso
  - Primeiro a entrar é o último a sair
- Estudamos a definição e operações para implementação com vetores e com listas ligadas
- Pilhas possuem diversas aplicações relevantes em computação