Algoritmos e Estruturas de Dados I

Profª Priscilla Abreu

priscilla.abreu@ime.uerj.br

2022.1

Algoritmos e Estruturas de Dados I

Roteiro da aula

- Listas lineares
  - Listas Sequenciais Particulares
    - Pilhas

Revisando...

Algoritmos e Estruturas de Dados I

Listas Linear

Listas lineares

Listas lineares gerais
SEM restrição de inserção e remoção de elementos

Listas particulares
COM restrição de inserção e remoção de elementos

Algoritmos e Estruturas de Dados I

Listas Lineares – Casos particulares:

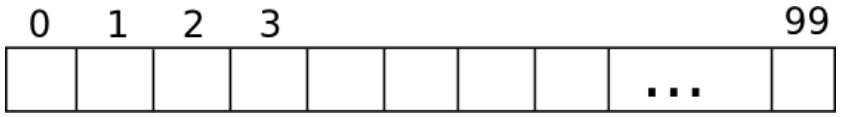
- Deque
  Inserção e remoção apenas nas extremidades;
- Pilha
  Inserção e remoção apenas em um extremo
- Fila
  Inserção em um extremo e remoção em outro extremo;

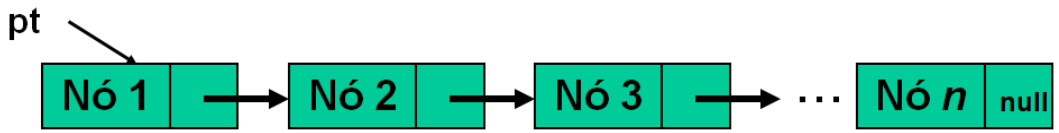
Algoritmos e Estruturas de Dados I

O tipo de armazenamento de uma lista linear pode ser classificado de acordo com a posição relativa na memória (contínua ou não) de cada dois nós consecutivos na lista.

Existem dois tipos de alocação:

- Alocação sequencial
- Alocação encadeada





Pilhas Sequenciais

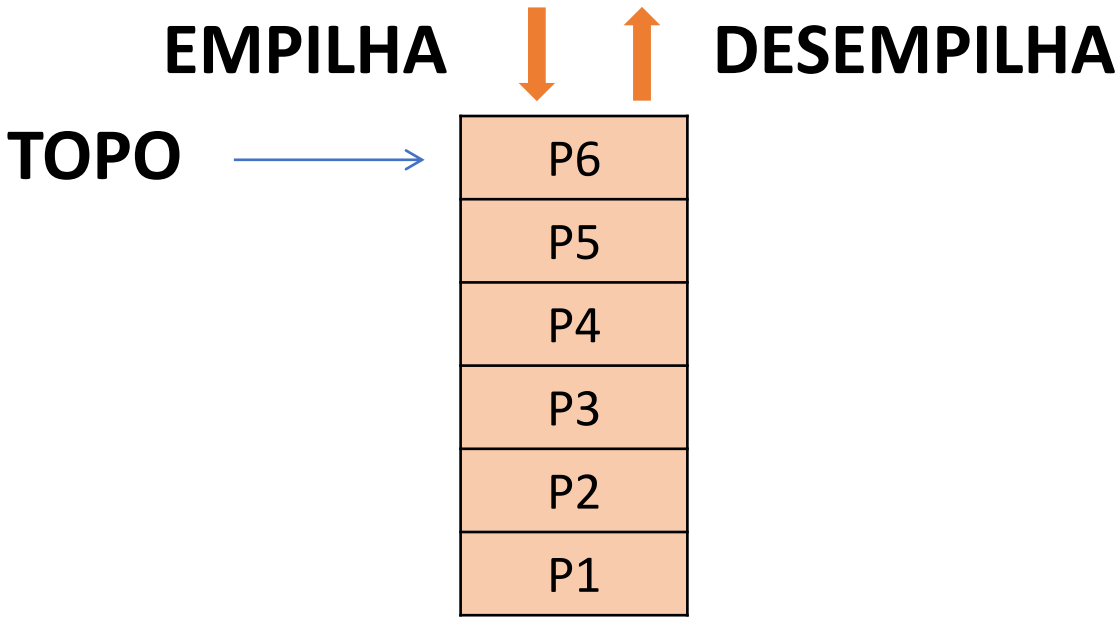
Algoritmos e Estruturas de Dados I

Pilha

- Estruturas de dados do tipo LIFO (last-in first-out): o último elemento a ser inserido, será o primeiro a ser removido.
- Manipulação dos elementos em apenas uma das extremidades: **topo**.
- Exemplos: pilha de pratos, pilha de livros, pilha de cartas de um baralho, etc.
    - Inserção: Empilha() ou Push()
    - Remoção: Desempilha() ou Pop()

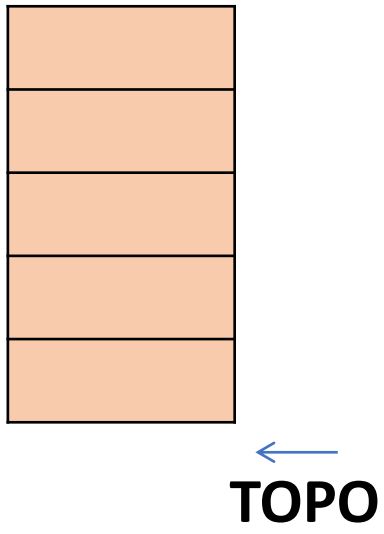
Algoritmos e Estruturas de Dados I

Pilha



Algoritmos e Estruturas de Dados I

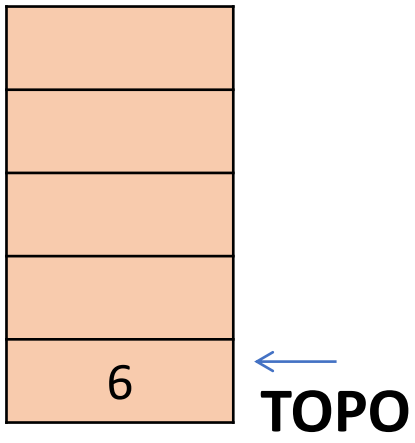
Pilha



Algoritmos e Estruturas de Dados I

Pilha

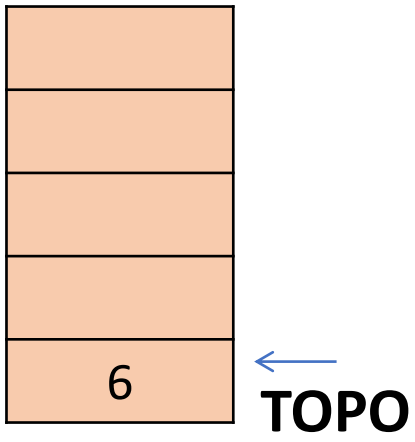
EMPILHA (6)



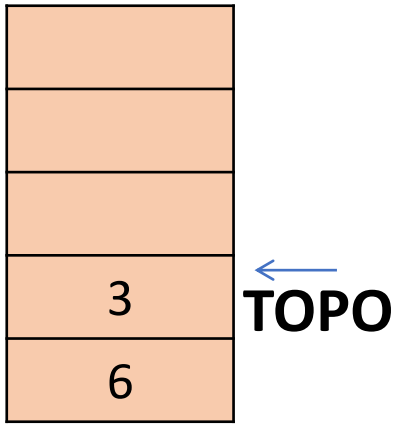
Algoritmos e Estruturas de Dados I

Pilha

EMPILHA (6)



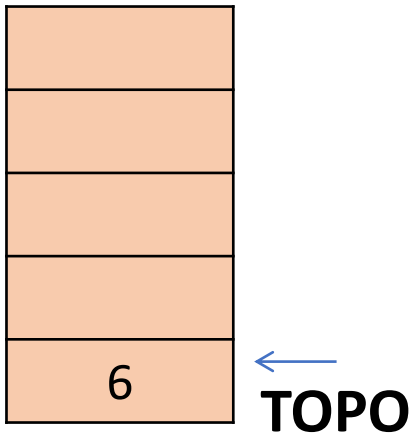
EMPILHA (3)



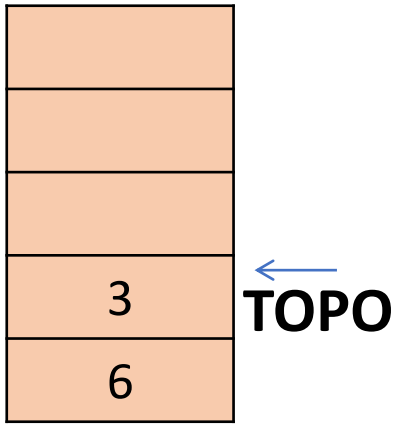
Algoritmos e Estruturas de Dados I

Pilha

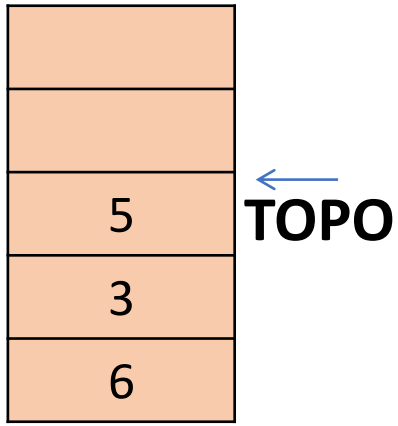
EMPILHA (6)



EMPILHA (3)



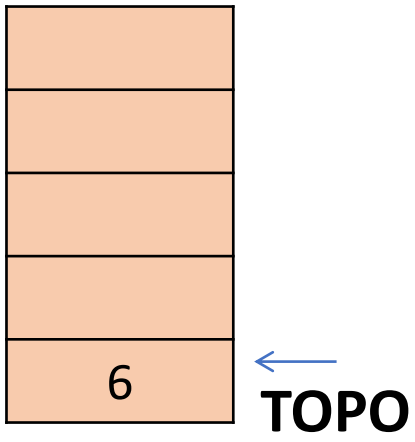
EMPILHA (5)



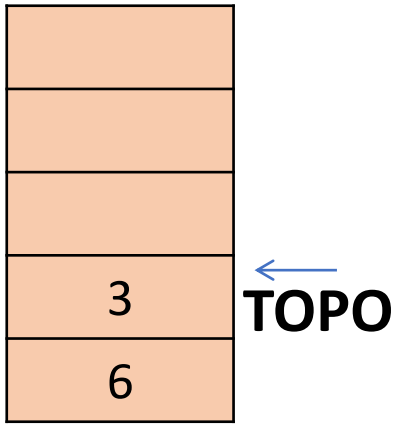
Algoritmos e Estruturas de Dados I

Pilha

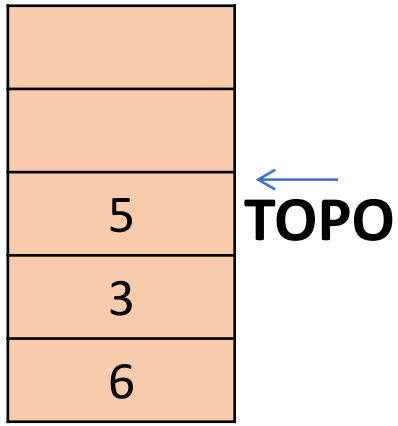
EMPILHA (6)



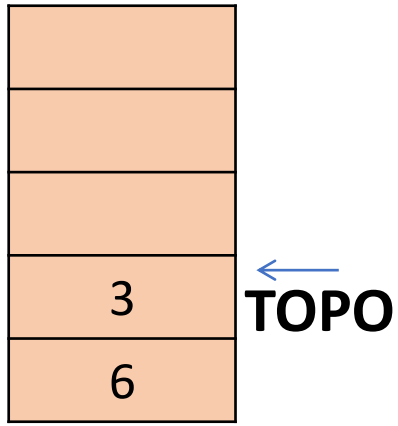
EMPILHA (3)



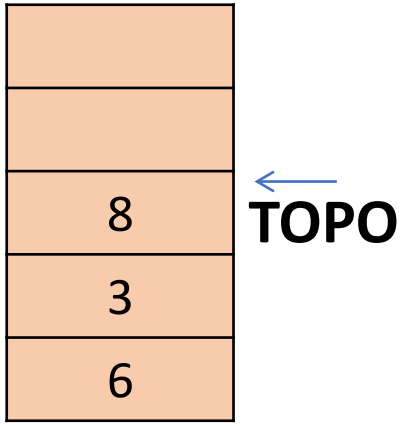
EMPILHA (5)



DESEMPILHA ()



EMPILHA (8)



Algoritmos e Estruturas de Dados I

Pilha

- Uso em aplicações onde os dados são obtidos na ordem inversa àquela em que foram fornecidos.
- Exemplos:
  - Calculadora para expressões matemáticas;
  - Conversão de número decimal para binário;
  - Mecanismo de fazer/desfazer do Word;
  - Mecanismo de navegação de páginas na Internet (avançar e retornar).

Algoritmos e Estruturas de Dados I

Pilha

- **Verificar se a pilha está cheia.**
- **Verificar se a pilha está vazia.**
- **Empilhamento**: consiste em inserir um valor no topo da pilha. É preciso verificar previamente se a pilha está cheia.
- **Desempilhamento**: consiste em retirar um valor do topo da pilha. É preciso verificar previamente se a pilha está vazia.
- **Mostrar o topo**

Algoritmos e Estruturas de Dados I

Pilha – implementação

estrutura no:
    chave: <inteiro>

estrutura pilha:
    valores [1..MAX]: no
    topo: inteiro

```
const int MAX = 10;
typedef struct no{
    int chave;
}no;

typedef struct pilhaSeq{
    no valores[MAX];
    int topo;
} pilhaSeq;
```

Algoritmos e Estruturas de Dados I

Pilha – implementação

```
void inicializa_pilha(int *topo){
   *topo = -1;
}

int pilha_cheia(int topo){
   return(topo==MAX-1);
}
```

Algoritmos e Estruturas de Dados I

Pilha – implementação

```
int pilha_vazia(int topo){
   return(topo==-1);
}
```

Algoritmos e Estruturas de Dados I

Pilha – implementação

```
int mostra_valor_topo(pilhaSeq *pilha){
   if(!pilha_vazia(pilha->topo)){
         printf("\nTopo da pilha: %d\n",pilha->valores[pilha->topo]);
   }
   else{
         printf("\nPilha vazia.\n");
   }
}
```

Algoritmos e Estruturas de Dados I

Pilha – implementação – EMPILHAR

- Consiste em inserir um valor no topo da pilha, caso a pilha não esteja cheia.
- Atualiza a posição do topo;
- Atribui o valor a ser inserido no novo topo.

Algoritmos e Estruturas de Dados I

Pilha – implementação – EMPILHAR

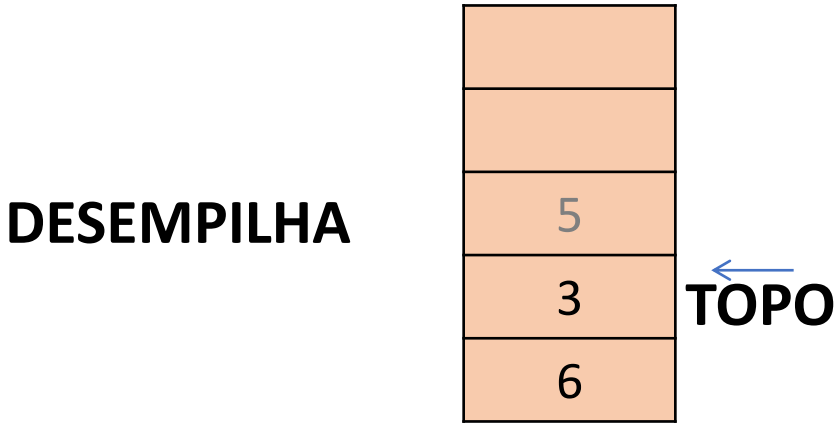
```
void empilha(pilhaSeq *pilha, int valor){
   if(!pilha_cheia(pilha->topo)){
            (pilha->topo)++;
            pilha->valores[pilha->topo].chave=valor;
            printf("\nValor empilhado!\n");
   }
   else{
            printf("\nPilha cheia!\n");
   }

}
```

Algoritmos e Estruturas de Dados I

Pilha – implementação – DESEMPILHAR

- Consiste em retirar um valor do topo da pilha e em seguida, ajustar o topo.
- Só é possível se a pilha não estiver vazia.
- A remoção de um elemento da pilha é realizada apenas alterando-se a informação da posição do topo.



Algoritmos e Estruturas de Dados I

Pilha – implementação – DESEMPILHAR

```
void desempilha(pilhaSeq *pilha){
   if(!pilha_vazia(pilha->topo)){
          printf("\nValor %d dempilhado!\n",pilha->valores[pilha->topo].chave);
          (pilha->topo)--;
   }
   else{
          printf("\nPilha vazia!\n");
   }

}
```

Algoritmos e Estruturas de Dados I

Pilha – implementação – DESEMPILHAR

Com base nas funções relacionadas à pilha, você deve implementar um programa que implemente uma pilha, disponibilizando as seguintes opções através de um menu ao usuário:

1- Empilhar

2- Desempilhar

3- Mostrar o topo

4- Sair

Algoritmos e Estruturas de Dados I

Pilha – implementação – DESEMPILHAR

```
#include <stdio.h>
#define M 10
//funções
...
int main(){
   pilhaSeq pilha[MAX];
   int valor, op;
   inicializa_pilha(&pilha->topo);
```

Algoritmos e Estruturas de Dados I

Pilha – implementação – DESEMPILHAR

```
do{
        printf("\n1- Empilhar");
        printf("\n2- Desempilhar");
        printf("\n3- Mostrar topo");
        printf("\n4- Sair");
        printf("\nInforme sua opção: ");
        scanf("%d",&op);
```

Algoritmos e Estruturas de Dados I

Pilha – implementação – DESEMPILHAR

```
switch(op){
        case 1:{
                printf("\nInforme o valor a empilhar: ");
                scanf("%d",&valor);
                empilha(pilha,valor);
                break;
        }
        case 2:{
                desempilha(pilha);
                break;
        }
```

Algoritmos e Estruturas de Dados I

Pilha – implementação – DESEMPILHAR

```
case 3:{
        mostra_valor_topo(pilha);
        break;
}
case 4:{
        printf("\nFinalizando...\n");
        break;
}
```

Algoritmos e Estruturas de Dados I

Pilha – implementação – DESEMPILHAR

```
                default:{
                        printf("\nOpção inválida!\n");
                        break;
                }
        }

  }while(op!=4);
}
```

Algoritmos e Estruturas de Dados I

Exercício

Dada uma pilha contendo números inteiros quaisquer, construir uma função que coloca os pares na base da pilha e os ímpares no topo da pilha. Você pode usar estruturas auxiliares.

FIM