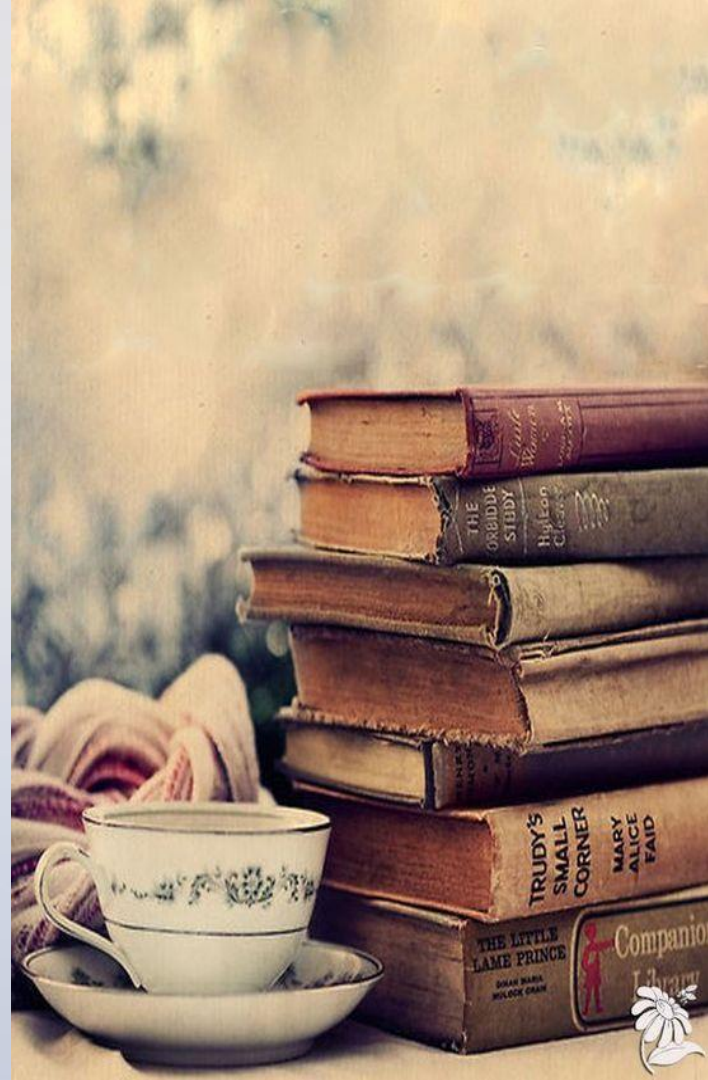


LABORATÓRIO DE PROGRAMAÇÃO II

Uma Introdução ao Conceito de Pilhas e Suas Aplicações






Agenda



Objetivos: Realizar operações de armazenamento e de recuperação de dados em estruturas lineares, com ordem de acesso pré-definida (LIFO)



Mecanismos de Acesso a Dados	
Pilha	
Atividade	



Mecanismos de Acesso à Dados

Há 4 mecanismos que controlam o modo como os dados podem ser acessados por um programa de computador

A escolha do mecanismo depende do problema abordado

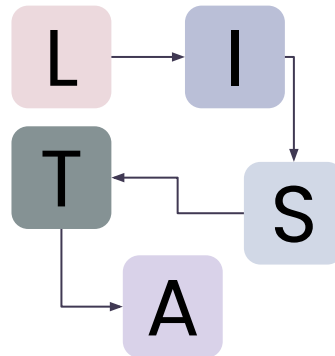
Pilha



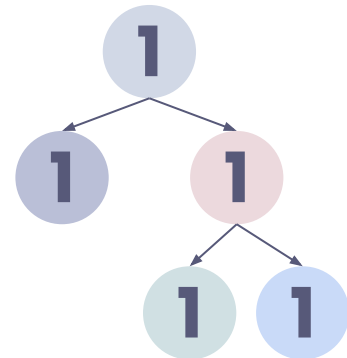
Fila



Lista Encadeada



Árvore Binária





...

01

Pilha

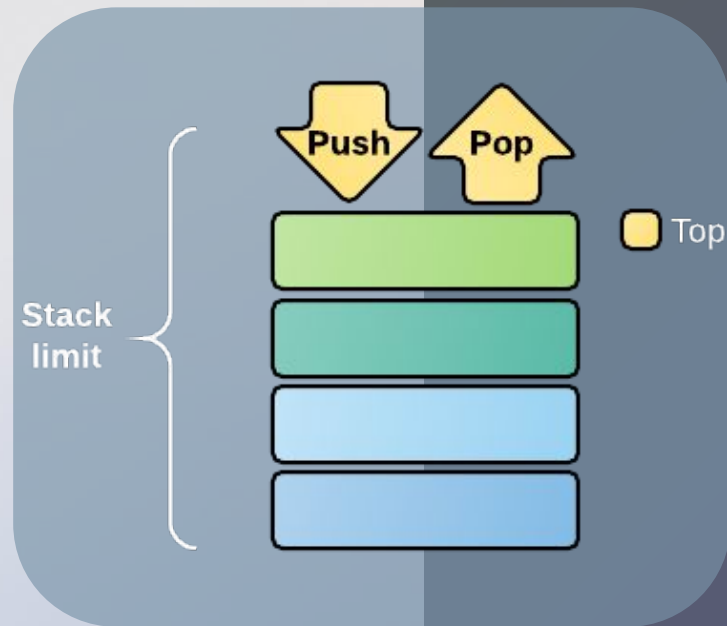
Pilhas

Uma **pilha (stack)** é uma estrutura de dados que admite **remoção** de elementos e **inserção** de novos objetos.

Estrutura sujeita à seguinte regra:

Sempre que houver uma remoção, o elemento removido é o que está na estrutura há menos tempo.

São estruturas de dados do tipo **LIFO**
(last-in first-out)



Pilhas

O exemplo clássico desse mecanismo é uma pilha de livros, de pratos ou produtos em uma prateleira no mercado.



Pilhas

O último a entrar é o primeiro a sair: **Ordem LIFO** (Last In, First Out)





Pilhas



São utilizadas duas funções:

push(): O item é empilhado

pop(): O item é desempilhado

O acesso a um item da pilha o destrói.

Esses são os únicos meios de armazenar e de recuperar em uma pilha; *não é permitido acesso aleatório* aos itens.



Pilhas

Exemplo de uma pilha em ação:

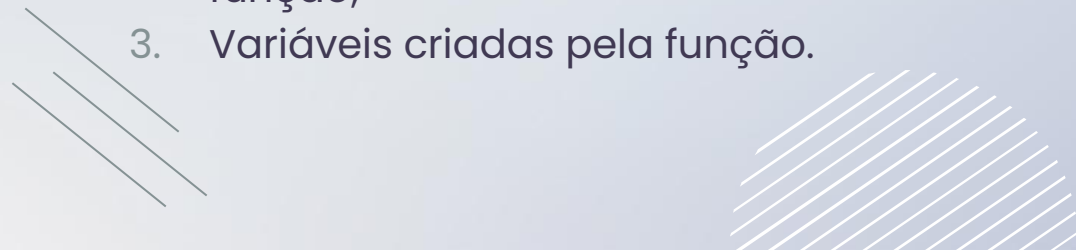
Ação	Conteúdo da pilha
push(A)	A
push(B)	B A
push(C)	C B A
pop() devolve C	B A
push(D)	D B A
pop() devolve D	B A
pop() devolve B	A



Pilhas

Exemplo de aplicação: argumentos passados para função.

Cada vez que uma função é chamada, ela coloca na pilha nesta ordem:

1. O endereço de retorno: ponto onde o programa deve retomar a execução após retornar da função chamada;
 2. Parâmetros usados na chamada da função;
 3. Variáveis criadas pela função.
- 

```
#include <stdio.h>
```

```
int funcao_B(int i)
{
    return(i-2);
}
```

```
int funcao_A(int n)
{
    int x;
    x=funcao_B(n-1);
    return(x);
}
```

```
void main()
{
    printf("resultado: %d\n",
           funcao_A(10));
}
```

Pilhas

Exemplo de aplicação: argumentos passados para função.

Execução da pilha

Pilha executando a linha 11:



Início da pilha

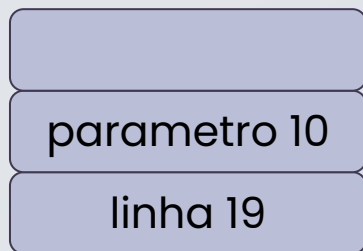
```
1  #include <stdio.h>
2
3  int funcao_B(int i)
4  {
5      return(i-2);
6  }
7
8  int funcao_A(int n)
9  {
10     int x;
11     x=funcao_B(n-1);
12     return(x);
13 }
14
15 void main()
16 {
17     printf("resultado: %d\n",
18           funcao_A(10));
19 }
20
```

Pilhas

Exemplo de aplicação: argumentos passados para função.

Execução da pilha

Pilha executando a linha 11:



Início da pilha

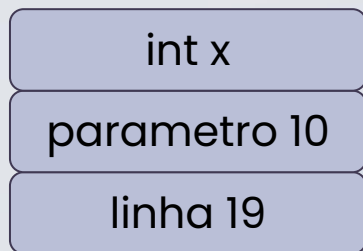
```
1  #include <stdio.h>
2
3  int funcao_B(int i)
4  {
5      return(i-2);
6  }
7
8  int funcao_A(int n)
9  {
10     int x;
11     x=funcao_B(n-1);
12     return(x);
13 }
14
15 void main()
16 {
17     printf("resultado: %d\n",
18           funcao_A(10));
19 }
20
```

Pilhas

Exemplo de aplicação: argumentos passados para função.

Execução da pilha

Pilha executando a linha 11:



Início da pilha

```
1  #include <stdio.h>
2
3  int funcao_B(int i)
4  {
5      return(i-2);
6  }
7
8  int funcao_A(int n)
9  {
10     int x;
11     x=funcao_B(n-1);
12     return(x);
13 }
14
15 void main()
16 {
17     printf("resultado: %d\n",
18           funcao_A(10));
19 }
20
```

Pilhas

Pilha executando a
linha 5:

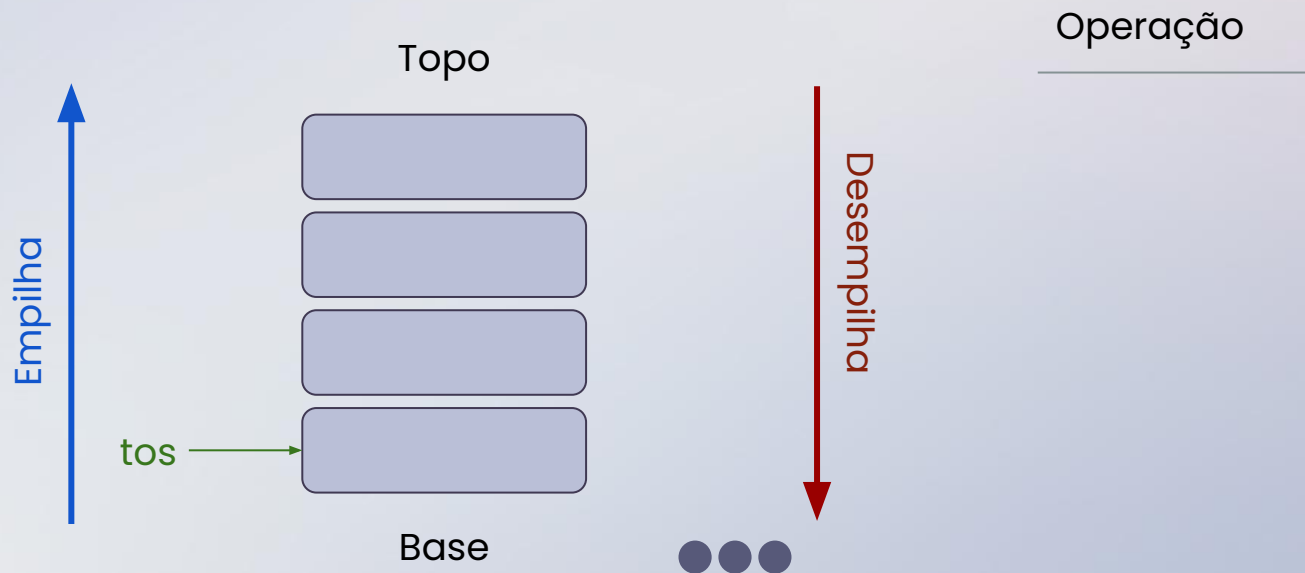


Início da pilha

```
1  #include <stdio.h>
2
3  int funcao_B(int i)
4  {
5      return(i-2);
6  }
7
8  int funcao_A(int n)
9  {
10     int x;
11     x=funcao_B(n-1);
12     return(x);
13 }
14
15 void main()
16 {
17     printf("resultado: %d\n",
18           funcao_A(10));
19 }
20
```

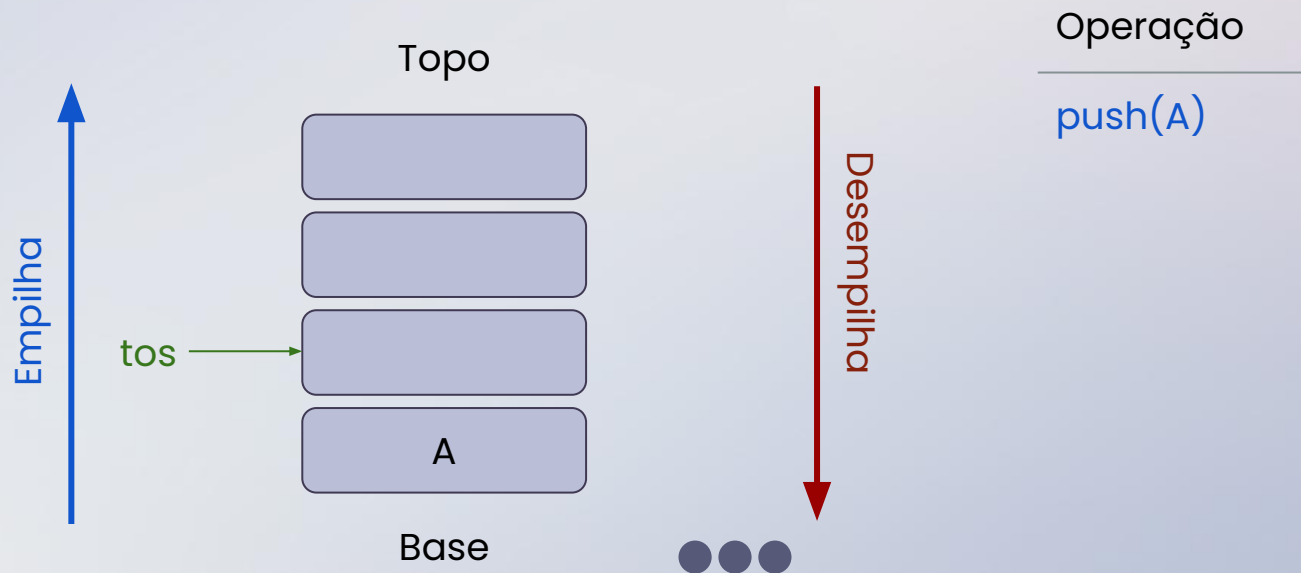
Pilhas

Usa um único índice para recuperação e armazenamento (tos); indica a próxima posição livre da pilha.



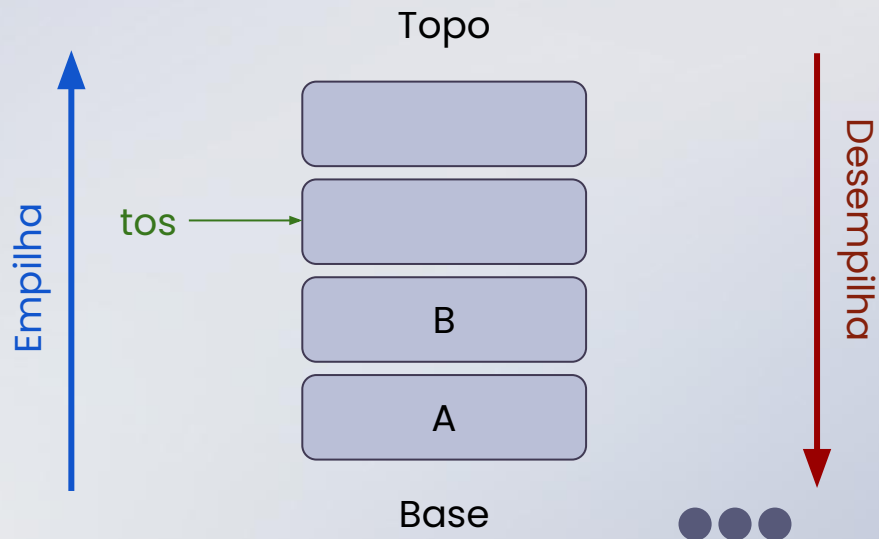
Pilhas

Usa um único índice para recuperação e armazenamento (tos); indica a próxima posição livre da pilha.



Pilhas

Usa um único índice para recuperação e armazenamento (tos); indica a próxima posição livre da pilha.

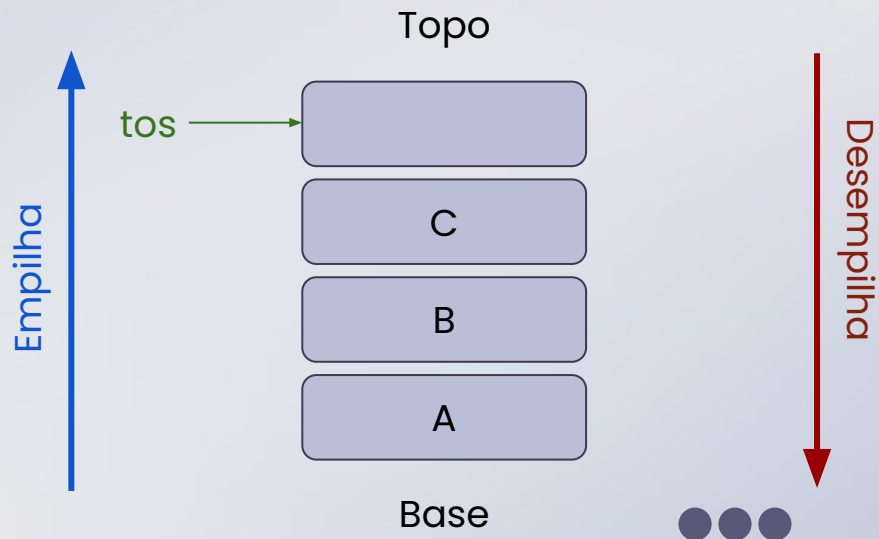


Operação

push(A)
push(B)

Pilhas

Usa um único índice para recuperação e armazenamento (tos); indica a próxima posição livre da pilha.

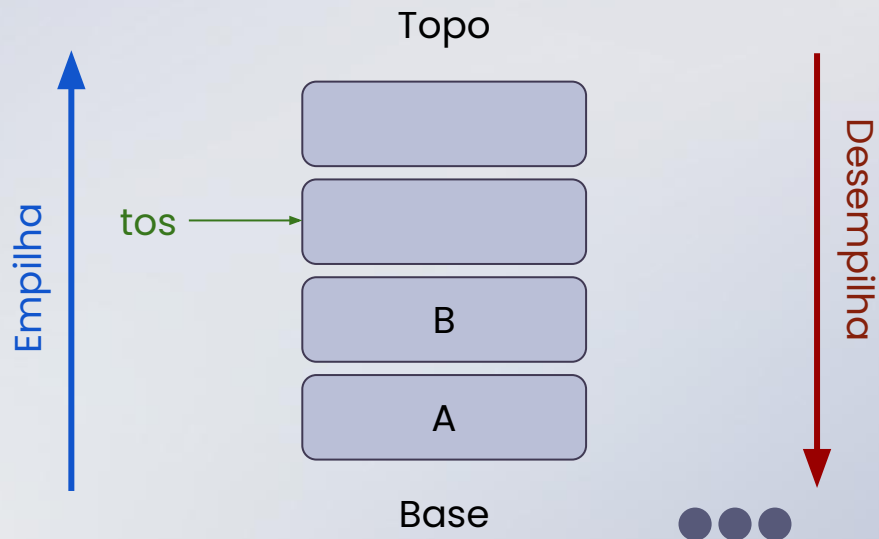


Operação

push(A)
push(B)
push(C)

Pilhas

Usa um único índice para recuperação e armazenamento (tos); indica a próxima posição livre da pilha.

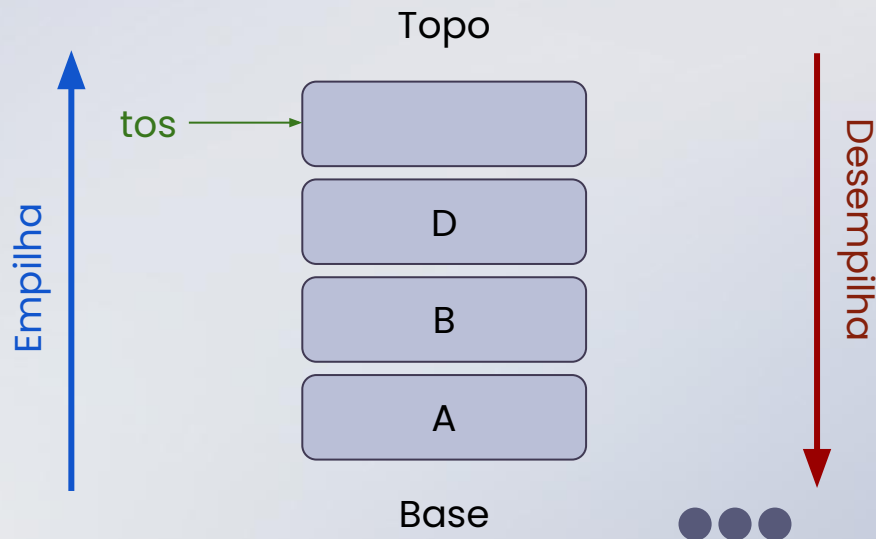


Operação

```
push(A)  
push(B)  
push(C)  
pop()
```

Pilhas

Usa um único índice para recuperação e armazenamento (tos); indica a próxima posição livre da pilha.

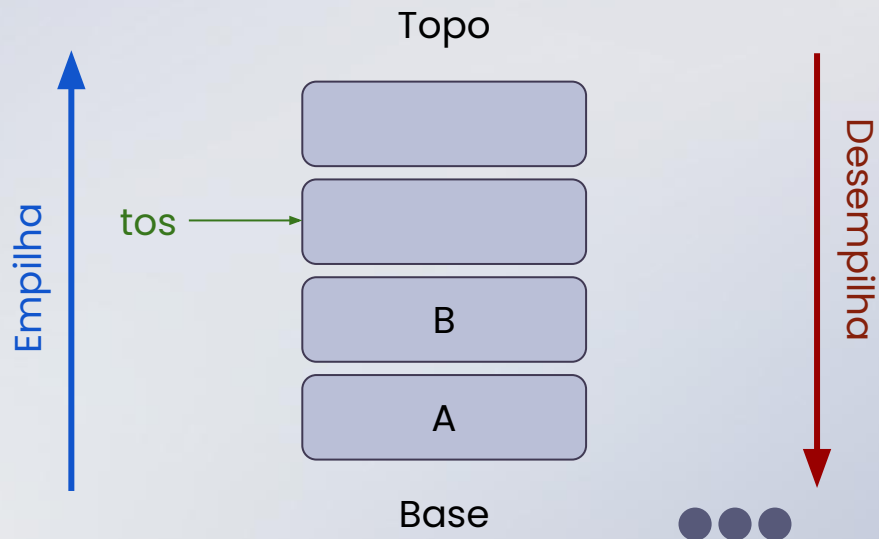


Operação

```
push(A)  
push(B)  
push(C)  
pop()  
push(D)
```

Pilhas

Usa um único índice para recuperação e armazenamento (tos); indica a próxima posição livre da pilha.

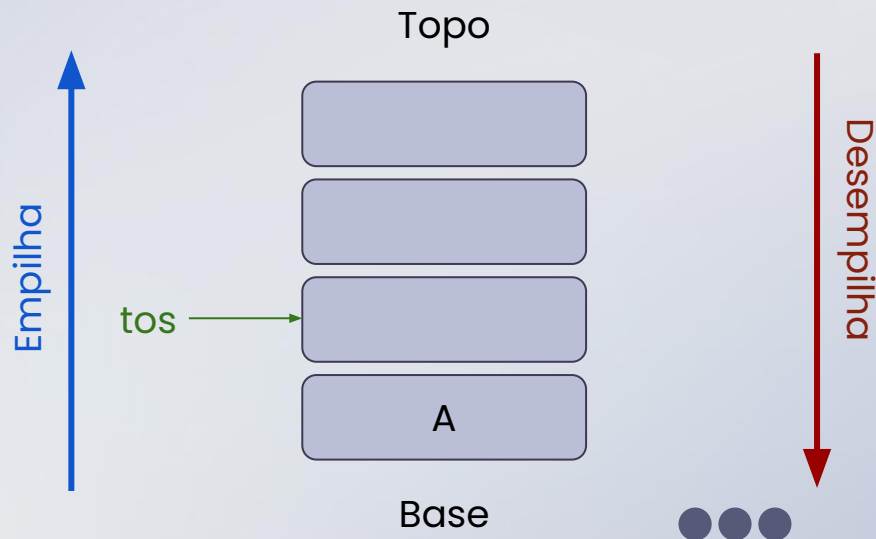


Operação

```
push(A)
push(B)
push(C)
pop()
push(D)
pop()
```

Pilhas

Usa um único índice para recuperação e armazenamento (tos); indica a próxima posição livre da pilha.

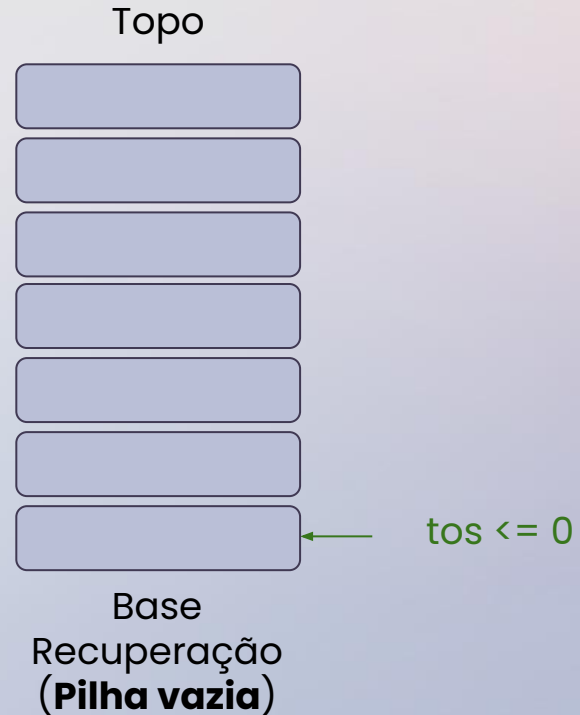
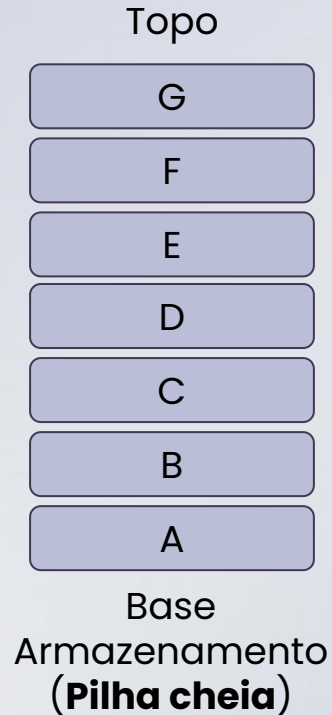


Operação

```
push(A)
push(B)
push(C)
pop()
push(D)
pop()
pop()
```

Pilhas


Restrições para armazenamento e recuperação





Pilhas

Abra o arquivo pilha.c e veja como podem ser implementadas as operações `push()` e `pop()` sobre uma estrutura vetorial alocada estaticamente.



A partir do arquivo acima, faça uma solução completa para o problema proposto.





Pilhas

ATIVIDADE

Faça uma pesquisa e aponte algumas das aplicações práticas de pilhas.

