# Pilhas

## Estrutura de Dados — QXD0010

Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará
Campus Quixadá

2º semestre/2025

# Pilha



- São `listas lineares` que adotam a política LIFO para a manipulação de elementos.

# Pilha



- São `listas lineares` que adotam a política LIFO para a manipulação de elementos.
- LIFO (*last-in first-out*): último a entrar é primeiro a sair. Remove primeiro objetos inseridos há menos tempo

# Pilha



- São `listas lineares` que adotam a política LIFO para a manipulação de elementos.
- LIFO (*last-in first-out*): último a entrar é primeiro a sair. Remove primeiro objetos inseridos há menos tempo

É como uma pilha de pratos:

# Pilha



- São `listas lineares` que adotam a política LIFO para a manipulação de elementos.
- LIFO (*last-in first-out*): último a entrar é primeiro a sair. Remove primeiro objetos inseridos há menos tempo

É como uma pilha de pratos:

- Empilha os pratos limpos sobre os que já estão na pilha

# Pilha



- São `listas lineares` que adotam a política LIFO para a manipulação de elementos.
- LIFO (*last-in first-out*): último a entrar é primeiro a sair. Remove primeiro objetos inseridos há menos tempo

É como uma pilha de pratos:

- Empilha os pratos limpos sobre os que já estão na pilha
- Desempilha o prato de cima para usar

# Pilha

Operações básicas:

# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha

# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

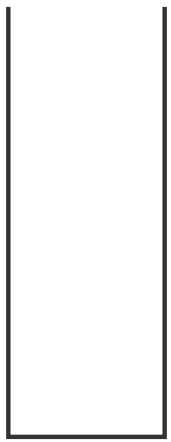
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo:



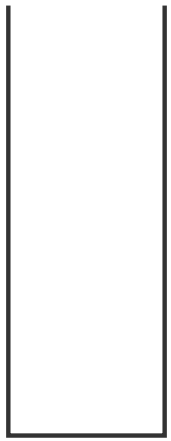
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `push(A)`



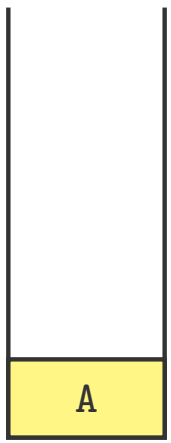
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `push(A)`



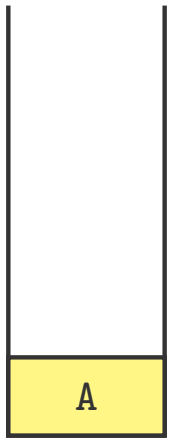
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `push(B)`



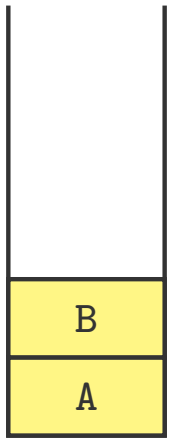
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `push(B)`



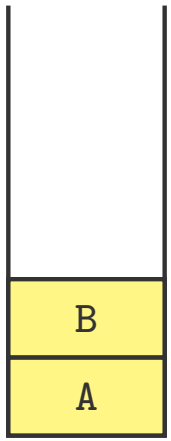
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `pop()`



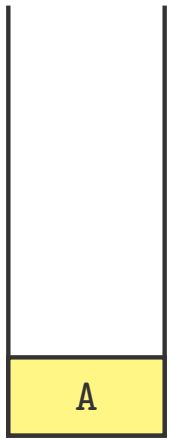
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `pop()`



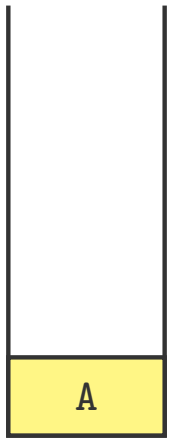
# Pilha

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `push(C)`



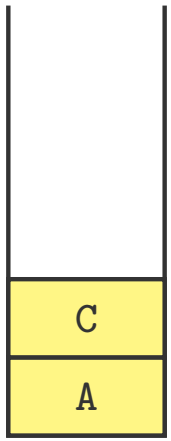
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `push(C)`



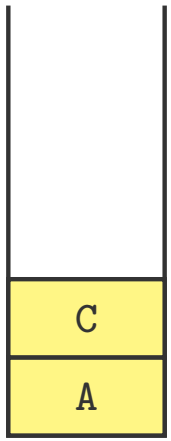
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `push(D)`



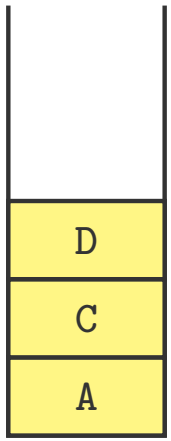
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `push(D)`



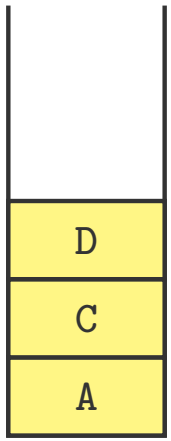
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `pop()`



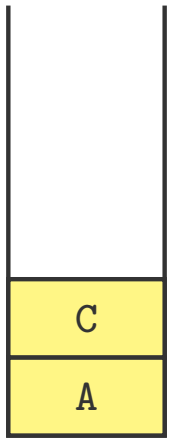
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `pop()`



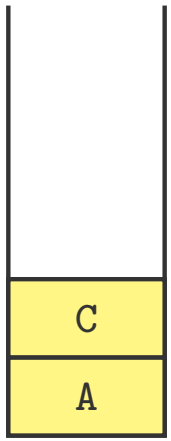
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `pop()`



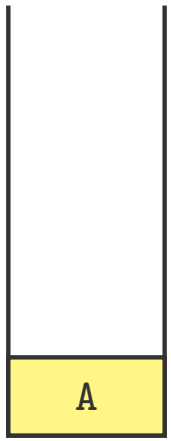
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `pop()`



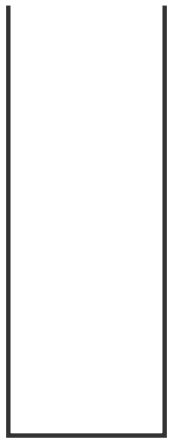
# Pilha



Operações básicas:

- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Exemplo: `pop()`



# Exemplos de aplicações



Algumas aplicações de pilhas:

- Balanceamento de parênteses
  - expressões matemáticas
  - linguagens de programação
  - HTML...
- Cálculo e conversão de notações
  - pré-fixa
  - pós-fixa
  - infixa (com parênteses)
- Percurso de estruturas de dados complexas (árvores, grafos, etc.)
- Recursão

# Implementação de uma Pilha

# Pilha usando Lista Encadeada



- Em algumas aplicações computacionais precisamos usar a estrutura de dados pilha, e não sabemos de antemão o tamanho da pilha.

- Nesses casos, a implementação da pilha pode ser feita de forma simples usando alocação encadeada.

# Pilha usando Lista Encadeada



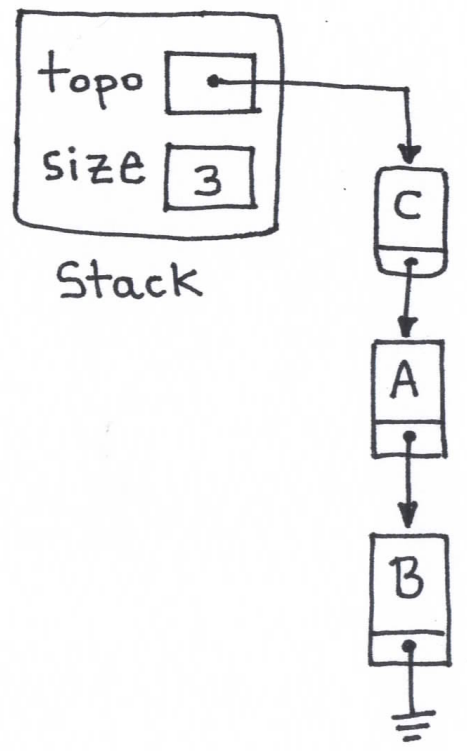
- Em algumas aplicações computacionais precisamos usar a estrutura de dados pilha, e não sabemos de antemão o tamanho da pilha.

- Nesses casos, a implementação da pilha pode ser feita de forma simples usando alocação encadeada.

# Implementação

Implementaremos a estrutura de dados Pilha como um struct chamado `Stack`, usando como base uma lista simplesmente encadeada sem nó sentinela.

Como exemplo ilustrativo, após empilhar `B`, `A` e `C`, a lista encadeada deve ter a seguinte configuração:



# struct node & struct stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // Definição do tipo struct node
2  struct node {
3      int value;
4      struct node *next;
5  };
6
7  // Definição do tipo struct stack
8  struct stack {
9      Node *top;
10     size_t size;
11 };
12
13 // declaração dos tipos opacos
14 typedef struct node Node;
15 typedef struct stack Stack;
```

# Operações de Pilha



- Vamos implementar as operações:
  - criar uma pilha vazia
  - verificar se a pilha está vazia
  - retornar o tamanho atual da pilha
  - consultar elemento no topo
  - inserir elemento no topo
  - remover elemento do topo
  - liberar a memória da pilha
  - trocar duas pilhas

# Arquivo de cabeçalho — Stack.h



```
1  #ifndef STACK_H
2  #define STACK_H
3  #include <stdio.h>
4  #include <stdbool.h>
5
6  typedef struct node Node;
7  typedef struct stack Stack;
8
9  Stack* stack_create(void);
10 size_t stack_size(Stack *p);
11 bool stack_empty(Stack *p);
12 void stack_push(Stack *p, int value);
13 void stack_pop(Stack *p);
14 int stack_top(Stack *p);
15 void stack_free(Stack *p);
16 void stack_swap(Stack *p1, Stack *p2);
17
18 #endif
```

# Arquivo fonte — `Stack.c`

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

**Exercício:** implemente as funcões da pilha no arquivo de código fonte Stack.c

# Exercícios

# Exercícios



(1) Inversão de palavras. Escreva uma função em C++ que inverta a ordem das letras de cada palavra de uma sentença, preservando a ordem das palavras. Suponha que as palavras da sentença são separadas por espaços. Por exemplo, a aplicação da sua operação à sentença AMU MEGASNEM ATERCES deve produzir a sentença UMA MENSAGEM SECRETA.

(2) Escreva uma função que verifique se uma string de entrada é da forma $str_1 C str_2$ tal que $str_1$ é uma string composta apenas por caracteres $A$ e $B$ e $str_2$ é a string reversa de $str_1$. Por exemplo, a cadeia $ABABBACABBABA$ é do formato especificado, enquanto as cadeias $ABABBACABB$, $ABA$, $BBBBCAA$ e $ABBACBAABBBBBAB$ não seguem o formato. Sua função deve obedecer o seguinte protótipo:

`bool` $str_1 C str_2$`(string` $str$`);`

# Exercícios



(3) Implemente uma pilha **usando exatamente DUAS filas**. A pilha deve armazenar números inteiros. Não é necessário implementar todas as operações do TAD Pilha, as QUATRO únicas operações que são obrigatórias para esta questão são as operações `empilhar (push)`, `desempilhar (pop)`, `construir pilha (Construtor)` e `destruir Pilha (Destrutor)`.

**Observação:** Crie um arquivo **main.cpp** a fim de testar a estrutura de dados pilha que você criou.

**Atenção:** Não é para contruir a pilha suando listas encadeadas ou vetores. Pois nós já fizemos isso em aula. A única estrutura de dados permitida e que pode ser usada para resolver esse exercícios são as filas. Use EXATAMENTE duas filas. Nem mais nem menos que isso.

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

# FIM