Pilhas

Estrutura de Dados — QXD0010

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

Universidade Federal do Ceará
Campus Quixadá

1º semestre/2021

Introdução

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:

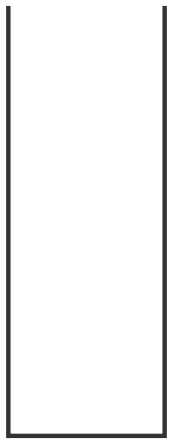
- push (*empilhar*): adiciona no topo da pilha
- pop (*desempilhar*): remove do topo da pilha

Pilha

Operações básicas:

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

Exemplo:

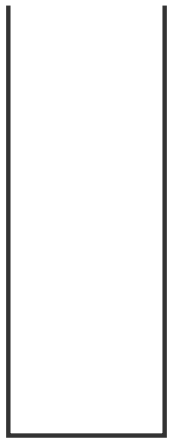


Pilha

Operações básicas:

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

Exemplo: `push(A)`

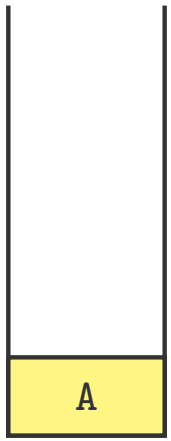


Pilha

Operações básicas:

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

Exemplo: `push(A)`



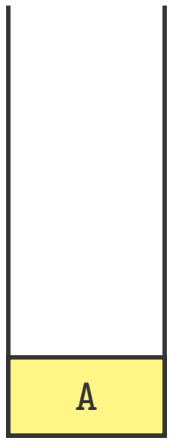
Pilha

Universidade Federal do Ceará
Campus Quixadá

Operações básicas:

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

Exemplo: `push(B)`

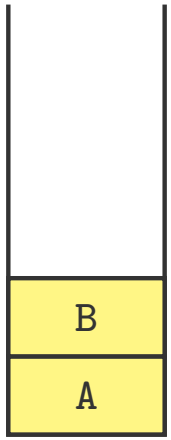


Pilha

Operações básicas:

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

Exemplo: `push(B)`



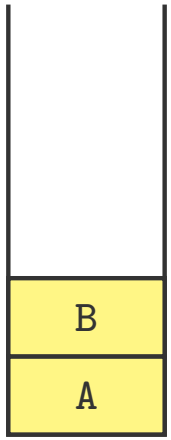
Pilha

Universidade Federal do Ceará
Campus Quixadá

Operações básicas:

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

Exemplo: `pop()`

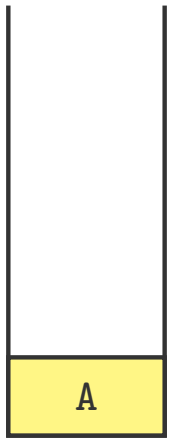


Pilha

Operações básicas:

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

Exemplo: `pop()`

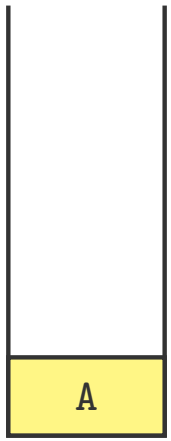


Pilha

Operações básicas:

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

Exemplo: `push(C)`



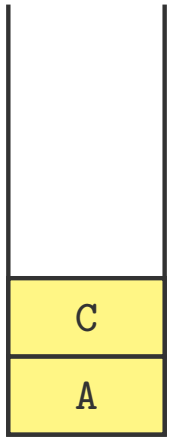
Pilha

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)`

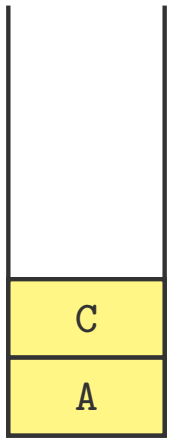


Pilha

Operações básicas:

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

Exemplo: `push(D)`

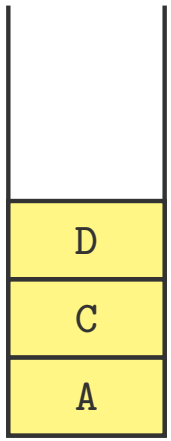


Pilha

Operações básicas:

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

Exemplo: `push(D)`

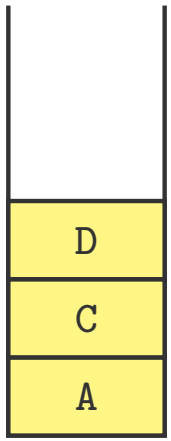


Pilha

Operações básicas:

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

Exemplo: `pop()`

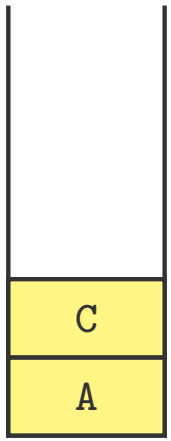


Pilha

Operações básicas:

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

Exemplo: `pop()`

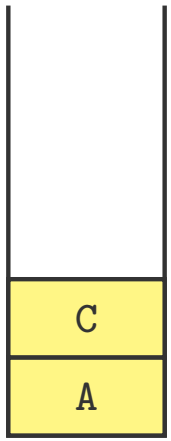


Pilha

Operações básicas:

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

Exemplo: `pop()`

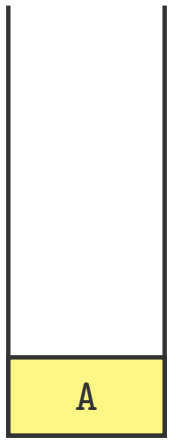


Pilha

Operações básicas:

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

Exemplo: `pop()`

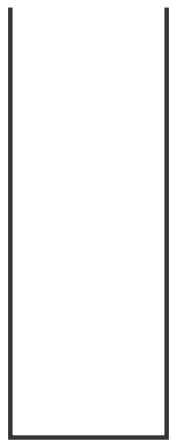


Pilha

Operações básicas:

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

Exemplo: `pop()`



Implementação de uma Pilha

Implementação de Pilha usando um Vetor:

- Em algumas aplicações computacionais que precisam de uma estrutura de dados pilha, é comum saber de antemão o tamanho da pilha.

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

Implementação de Pilha usando um Vetor:

- Em algumas aplicações computacionais que precisam de uma estrutura de dados pilha, é comum saber de antemão o tamanho da pilha.

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

- Vamos implementar as operações:
  - criar uma pilha vazia
  - inserir elemento no topo
  - remover elemento do topo
  - verificar se a pilha está vazia
  - verificar se a pilha está cheia
  - liberar a estrutura de pilha

Implementação de Pilha usando um Vetor:

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Em algumas aplicações computacionais que precisam de uma estrutura de dados pilha, é comum saber de antemão o tamanho da pilha.

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

- Vamos implementar as operações:
  - criar uma pilha vazia
  - inserir elemento no topo
  - remover elemento do topo
  - verificar se a pilha está vazia
  - verificar se a pilha está cheia
  - liberar a estrutura de pilha

- Implementaremos a pilha como uma Classe chamada Stack.

Item.h

- Nossa pilha armazenará elementos do tipo `Item`, que é definido no arquivo de cabeçalho `Item.h`.

- O conteúdo desse arquivo é o seguinte:

```
1 #ifndef ITEM_STACK_H
2 #define ITEM_STACK_H
3
4 typedef double Item;
5
6 #endif
```

Stack.h — Declarando a classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  #ifndef STACK_H
2  #define STACK_H
3  #include <iostream>
4  #include "Item.h"
5
6  class Stack {
7      private:
8          Item *_vec; // Ponteiro para um vetor de Item
9          int _top; // Posicao do proximo slot disponivel
10         int _capacity; // Tamanho total do vetor
11     public:
12         Stack(int n); // Construtor: recebe capacidade
13         ~Stack(); // Destrutor: libera memoria alocada
14         void push(Item v); // Inserir elemento no topo
15         void pop(); // Remover elemento do topo
16         Item top(); // Consulta elemento no topo
17         bool empty(); // Pilha esta vazia?
18         bool full(); // Pilha esta cheia?
19 };
20 #endif
```

Stack.cpp — Implementação da classe Stack

Universidade Federal do Ceará
Campus Quixadá

```
1 // Inclusao das bibliotecas
2 #include <iostream>
3 #include "Item.h"
4 #include "Stack.h"
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1 // Inclusao das bibliotecas
2 #include <iostream>
3 #include "Item.h"
4 #include "Stack.h"
5
6 // Construtor: recebe capacidade total da pilha
7 Stack::Stack(int n) {
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // Inclusao das bibliotecas
2  #include <iostream>
3  #include "Item.h"
4  #include "Stack.h"
5
6  // Construtor: recebe capacidade total da pilha
7  Stack::Stack(int n) {
8      _vec = new Item[n];
9      _capacity = n;
10     _top = 0;
11 }
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // Inclusao das bibliotecas
2  #include <iostream>
3  #include "Item.h"
4  #include "Stack.h"
5
6  // Construtor: recebe capacidade total da pilha
7  Stack::Stack(int n) {
8      _vec = new Item[n];
9      _capacity = n;
10     _top = 0;
11 }
12
13 // Destrutor: libera o vetor que foi alocado
14 Stack::~Stack() {
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // Inclusao das bibliotecas
2  #include <iostream>
3  #include "Item.h"
4  #include "Stack.h"
5
6  // Construtor: recebe capacidade total da pilha
7  Stack::Stack(int n) {
8      _vec = new Item[n];
9      _capacity = n;
10     _top = 0;
11 }
12
13 // Destrutor: libera o vetor que foi alocado
14 Stack::~Stack() {
15     if(_vec != nullptr) delete[] _vec;
16 }
```

Stack.cpp — Implementação da classe Stack


```
1 // isEmpty() -- Pilha esta vazia?
2 bool Stack::empty() {
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1 // isEmpty() -- Pilha esta vazia?
2 bool Stack::empty() {
3     return (_top == 0);
4 }
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1 // isEmpty() -- Pilha esta vazia?
2 bool Stack::empty() {
3     return (_top == 0);
4 }
5
6 // isFull() -- Pilha esta cheia?
7 bool Stack::full() {
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1 // isEmpty() -- Pilha esta vazia?
2 bool Stack::empty() {
3     return (_top == 0);
4 }
5
6 // isFull() -- Pilha esta cheia?
7 bool Stack::full() {
8     return (_top == _capacity);
9 }
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // isEmpty() -- Pilha esta vazia?
2  bool Stack::empty() {
3      return (_top == 0);
4  }
5
6  // isFull() -- Pilha esta cheia?
7  bool Stack::full() {
8      return (_top == _capacity);
9  }
10
11 // top() -- Consulta valor do elemento no topo
12 // Supoe que pilha nao esta vazia
13 Item Stack::top() {
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // isEmpty() -- Pilha esta vazia?
2  bool Stack::empty() {
3      return (_top == 0);
4  }
5
6  // isFull() -- Pilha esta cheia?
7  bool Stack::full() {
8      return (_top == _capacity);
9  }
10
11 // top() -- Consulta valor do elemento no topo
12 // Supoe que pilha nao esta vazia
13 Item Stack::top() {
14     return _vec[_top-1];
15 }
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1 // push() -- Insere elemento no topo
2 // Supoe que a pilha nao esta cheia
3 void Stack::push(Item v) {
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1 // push() -- Insere elemento no topo
2 // Supoe que a pilha nao esta cheia
3 void Stack::push(Item v) {
4     _vec[_top] = v;
5     _top++;
6 }
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // push() -- Insere elemento no topo
2  // Supoe que a pilha nao esta cheia
3  void Stack::push(Item v) {
4      _vec[_top] = v;
5      _top++;
6  }
7
8  // pop() -- Remove elemento do topo e retorna
       valor
9  // Supoe que a pilha nao esta vazia
10 void Stack::pop() {
```

Stack.cpp — Implementação da classe Stack

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // push() -- Insere elemento no topo
2  // Supoe que a pilha nao esta cheia
3  void Stack::push(Item v) {
4      _vec[_top] = v;
5      _top++;
6  }
7
8  // pop() -- Remove elemento do topo e retorna
       valor
9  // Supoe que a pilha nao esta vazia
10 void Stack::pop() {
11     _top--;
12 }
```

main.cpp — Exemplo de programa


```
1  #include <iostream>
2  #include "Item.h"
3  #include "Stack.h"
4  using namespace std;
5
6  int main() {
7      int n;
8      cout << "Digite tamanho da pilha: ";
9      cin >> n;
10     Stack *pilha = new Stack(n); // Cria uma pilha vazia
11
12     Item valor;
13     while ( !pilha->isFull() ) {
14         cout << "Digite um numero a ser empilhado: ";
15         cin >> valor;
16         pilha->push(valor);
17     }
18
19     while (!pilha->isEmpty()) cout << pilha->pop() << " ";
20
21     delete pilha; // Libera a pilha
22     return 0;
23 }
```

Exemplos de aplicações

Algumas aplicações de pilhas:

- Balanceamento de parênteses
  - expressões matemáticas
  - linguagens de programação
  - HTML...

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.)

Exemplos de aplicações

Algumas aplicações de pilhas:

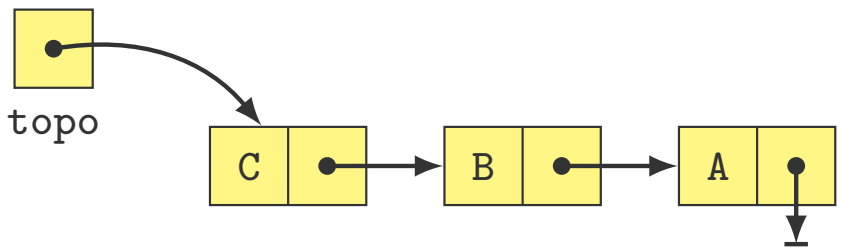
- Balanceamento de parênteses
  - expressões matemáticas
  - linguagens de programação
  - HTML...
- Cálculo e conversão de notações
  - pré-fixa
  - pós-fixa
  - infixa (com parênteses)
- Percurso de estruturas de dados complexas (árvores, grafos, etc.)
- Recursão

Exercícios

Exercício

Exercício: Implemente a estrutura de dados Pilha usando uma lista simplesmente encadeada sem nó cabeça. Implemente a pilha como uma classe.

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



Cada nó da lista é um struct definido do seguinte modo:

```
1 struct Node {
2     Item key;
3     Node *next;
4 };
```

O tipo `Item` está definido no arquivo `Item.h`, visto nesta aula.

Exercício

Defina sua classe do seguinte modo:

```
1  #ifndef STACK_H
2  #define STACK_H
3  #include <iostream>
4  #include "Item.h"
5
6  struct Node;
7
8  class Stack {
9      private:
10         Node *_top; // Ponteiro para o topo da pilha
11     public:
12         Stack(); // Construtor: Inicia a pilha
13         ~Stack(); // Destrutor: libera memoria alocada
14         void push(Item v); // Inserir elemento no topo
15         void pop(); // Remover elemento do topo
16         Item top(); // Consulta o elemento no topo
17         bool empty(); // Pilha esta vazia?
18 };
19 #endif
```

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

FIM