

Acesse este conteúdo pelo smartphone

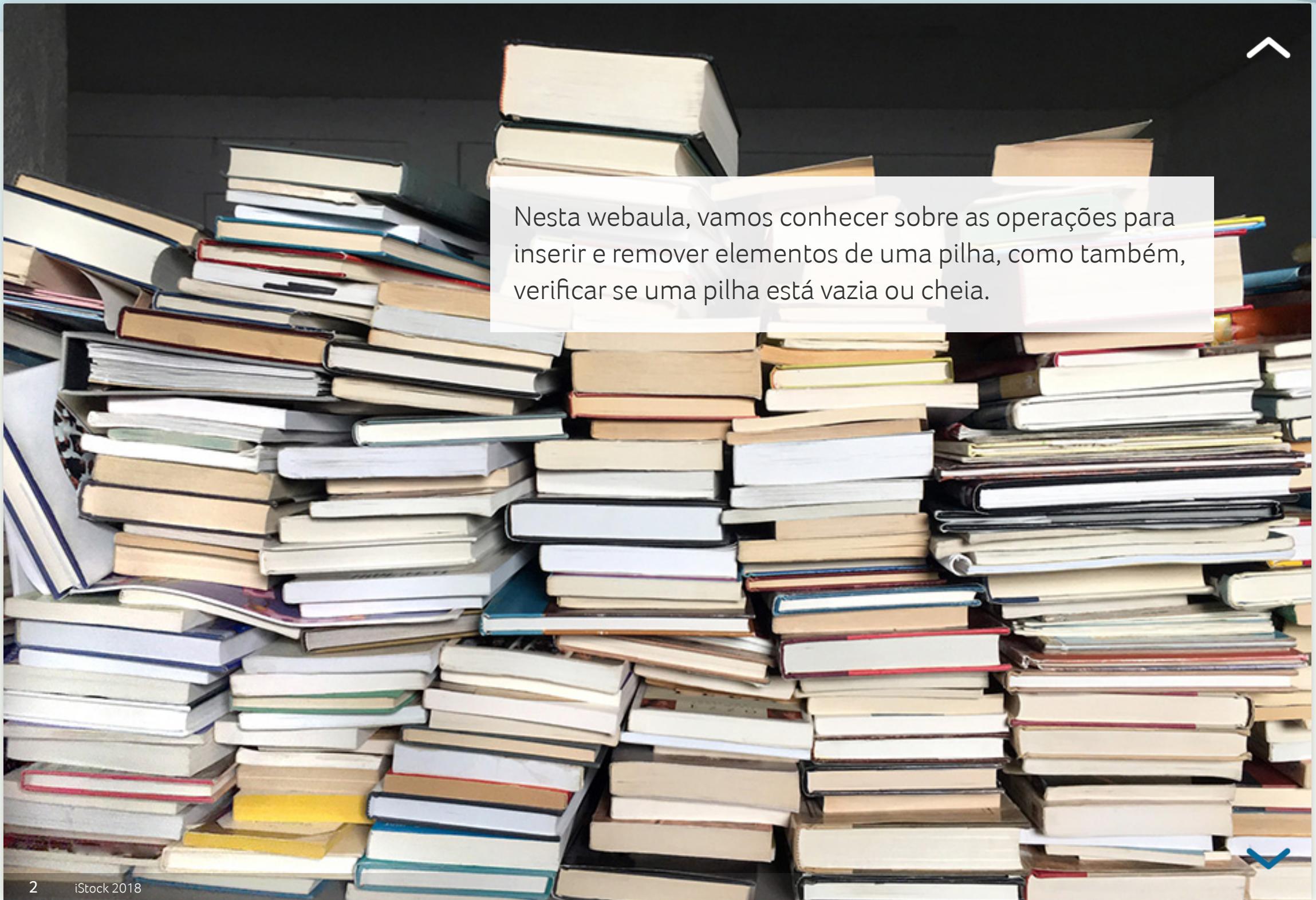


O que é isso?  
Clique no código e saiba mais.

# Algoritmos e Estrutura de Dados

# **Webaula 2**

## **Operações e Problemas com Pilhas**



Nesta webaula, vamos conhecer sobre as operações para inserir e remover elementos de uma pilha, como também, verificar se uma pilha está vazia ou cheia.



Em uma estrutura da pilha, conforme Celes, Cerqueira, Range (2004), devem ser implementadas duas operações básicas:

- Empilhar um novo elemento.
- Desempilhar um elemento.

A operação de empilhar um novo elemento, segundo Lorenzi, Mattos, Carvalho (2015), tem a função de inserir um novo elemento na pilha, e definida na programação em C++ como *push()*. Já a operação de desempilhar tem a função de remover um elemento do topo da pilha utilizada na programação em C++ como *pop()*.

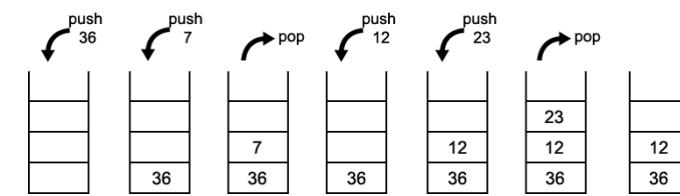




Conforme Drozdek (2016), outras operações que podem ser implementadas na pilha são:

- Limpar a pilha, utilizando a função *clear()*.
- Verificar se a pilha está vazia, com base na função *isEmpty()*.

Na imagem podemos observar uma sequência de operações para inserir um novo elemento na pilha com a função *push()* e remoção do elemento com a função *pop()*.



Fonte: elaborado pelo autor.





## Inserir elementos na pilha

Conforme Tenenbaum (2007), uma pilha possui uma estrutura que pode ser declarada contendo dois objetos:

Um ponteiro, que irá armazenar o endereçamento inicial da pilha.

Um valor inteiro, que irá indicar a posição do topo da pilha.



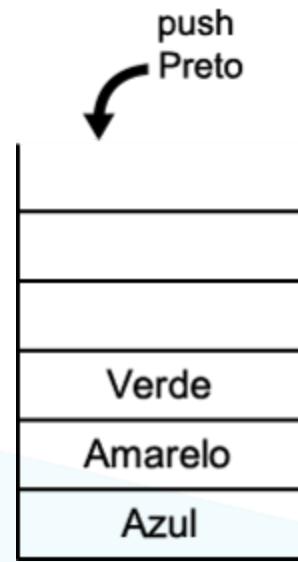
Para utilização de uma pilha, primeiramente é necessário:

Criar a declaração da estrutura da pilha.

Criar a pilha com a alocação dinâmica.

Criar as funções para inserir e remover da pilha.

Com a função para criar a pilha realizada, ela estará vazia, ou seja, não terá nenhum elemento na pilha em sua criação. Assim, você pode criar a função que vai permitir ser inserido um novo elemento na pilha, conforme a imagem.

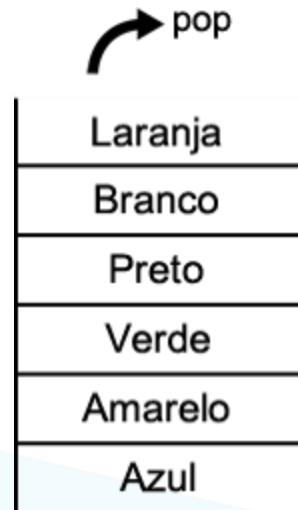


Fonte: elaborado pelo autor.



## Remover elementos da pilha

Lembre-se de que, a remoção de um novo elemento é realizada somente pelo topo da pilha, como na imagem, onde o elemento laranja está no topo e será removido com a função *pop()*.



Fonte: elaborado pelo autor.

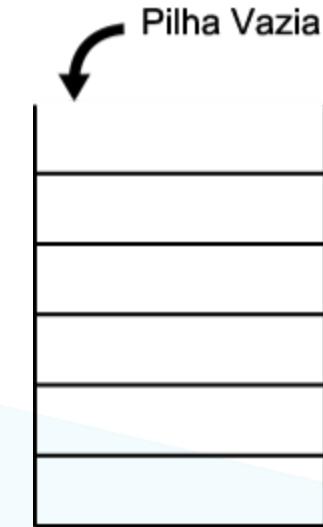




## Informar se a pilha está vazia

Uma pilha não possuirá nenhum elemento em sua inicialização, assim, não é possível remover elementos, apenas inserir elementos na pilha.

É interessante tratar esse método para o sistema informar ao usuário que a remoção do elemento não é possível e que a pilha está vazia, como na imagem.



Fonte: elaborado pelo autor.





Explore a galeria e conheça a programação de algumas etapas estudadas.

A declaração da estrutura inicial para criação de uma pilha pode ser implementada por:

```
struct Pilha {  
    int topo;  
    int capacidade;  
    float * proxElem;  
};  
struct Pilha minhaPilha;
```





## Operações com pilhas: problema do labirinto

Os labirintos são desafios criados como problematização de estrutura de dados. As pilhas podem ser aplicadas também no uso de algoritmos de *Backtracking*, que consiste em criar marcações para onde o algoritmo pode retornar.

Em um labirinto, por exemplo, para encontrar um caminho correto, podemos andar pelo labirinto até que se encontre uma divisão nesse caminho. Caso o caminho escolhido não possua uma saída, é removido o ponto anterior da pilha, voltando ao último ponto em que o labirinto se dividiu, recomeçamos por um outro caminho ainda não escolhido, adicionando na pilha o novo caminho.



A seguir, um trecho de implementação de criação de uma solução para labirinto:

```
/*Chama a função inicLabirinto, passando o labirinto, a pilha, o valor da linha e da coluna como  
passagem de parâmetros*/  
void inicLabirinto(Labirinto *l, Pilha *p_l, int linha, int coluna){  
    int i, j, flag = 0;  
    char aux;  
    elem_t_pilha origem;
```



```
/*Aplicamos uma rotina em matriz para verificar se a posição foi visitada (1) ou não (0)*/
for(i = 0 ; i < linha ; i++){
    for(j = 0 ; j < coluna ; j++){
        if(l->p[i][j].tipo == '0'){
            l->p[i][j].visitado = 1; //Visitado
            origem.x = i;
            origem.y = j;
            /*Insere na pilha a posição de origem*/
            push(p_l, origem);
        }
    }
}
```



As pilhas estão presentes no nosso dia a dia sem que possamos perceber. Tenho certeza de que você já deve ter visto os caminhões do tipo cegonha pelas estradas, por foto ou até mesmo por televisão. Os carros que estão no caminhão são inseridos um por vez, e se for preciso remover o primeiro carro colocado no caminhão, será necessário remover o que está atrás dele.



## Você já conhece o Saber?

Aqui você tem na palma da sua mão a **biblioteca digital** para sua **formação profissional**.

Estude no celular, tablet ou PC em qualquer hora e lugar sem pagar mais nada por isso.

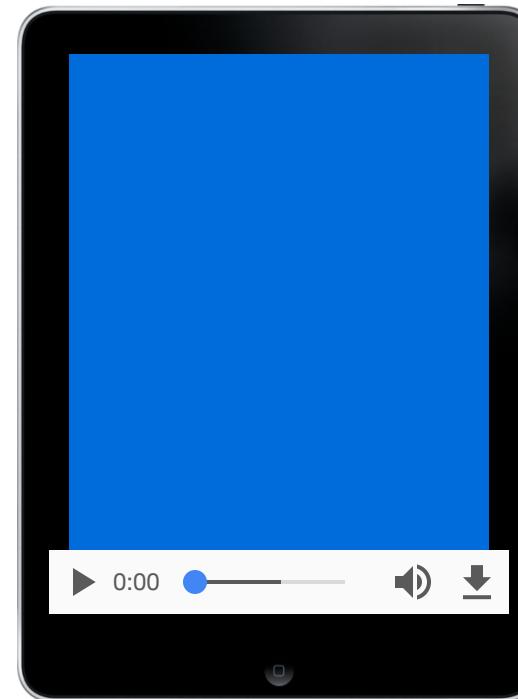
Mais de 450 livros com interatividade, vídeos, animações e jogos para você.



Android:  
<https://goo.gl/yAL2Mv>



iPhone e iPad - IOS:  
<https://goo.gl/OFWqcq>





**Bons estudos!**