

# Algoritmos e Estrutura de Dados PCO001

## Resolução de: **Exercícios - Pilha**

Carlos Augusto Ribeiro

2024-01

**Tarefa 1** - Considerando a Figura 1, apresente a sequência de operações empilha e desempilha que devem ser aplicadas sobre as pilhas X, Y e Z para que, partindo do estado inicial, possamos chegar ao estado final.

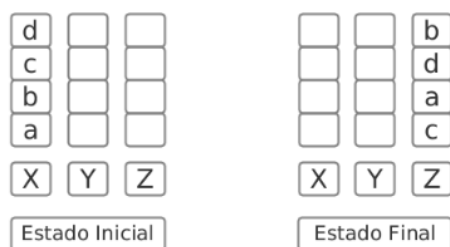


Figure 1: Estado inicial e final das pilhas X, Y e Z.

### Solução

1. desempilha X
2. empilha Y
3. desempilha X
4. empilha Y
5. desempilha X
6. empilha Z
7. desempilha Y
8. empilha Z
9. desempilha X
10. empilha Z
11. desempilha Y
12. empilha Z

**Tarefa 2** - Considere as seguintes funções:

- **Push(P, a)**: Aumenta o tamanho da pilha **P**, acrescentando o elemento **a** no seu topo (empilha);

- **Pop(P)**: Remove e retorna o elemento que está no topo da pilha **P** (desempilha);
- **Top(P)**: Retorna o elemento do topo da pilha **P**, sem desempilhar.

Observe a seguir como estas operações interagem para alterar o estado de uma pilha **P**, inicialmente vazia (**P**[]), cuja extremidade esquerda foi escolhida como topo. Por exemplo, imagine a pilha **P** com os elementos [**r,g**]. Ao executar o comando **Push(P,b)**, a pilha **P** ficaria [**b,r,g**].

- (a) **Push(P,a)**; → [**a**]
- (b) **Push(P,Pop(P))**; → [**a**]
- (c) **Push(P,b)**; → [**b,a**]
- (d) **Pop(P)**; → **b** → [**a**]
- (e) **Push(P,c)**; → [**c,a**]
- (f) **Push(P,e)**; → [**e,c,a**]
- (g) **Push(P,Top(P))**; → [**e,e,c,a**]
- (h) **Pop(P)**; → **e** → [**e,e,c,a**]

**Tarefa 3** - Considere uma pilha **P** de valores inteiros. Implemente uma função que apresenta o número de elementos contidos em **P**.

**Protótipo:** int TamanhoPilha (Pilha).

**Solução**

```
int TamanhoPilha(Pilha &pilha) {
    int tamanho = 0;
    char elemento;
    Pilha aux; IniciaPilha(aux);
    while (Desempilha(pilha, elemento)) {
        tamanho++;
        Empilha(aux, elemento);
    }
    while (Desempilha(aux, elemento)) {
        Empilha(pilha, elemento);
    }
    return tamanho;
}
```

**Tarefa 4** - Considere uma pilha **P** de valores inteiros. Implemente uma função que apresenta a média dos valores contidos em **P** (Figura 2).

**Protótipo:** float MediaPilha (Pilha).

**Solução**

```

float MediaPilha(Pilha &pilha) {
    int tamanho = 0;
    float sum = 0;
    int elemento;
    Pilha aux; IniciaPilha(aux);
    while (Desempilha(pilha, elemento)) {
        tamanho++;
        sum+=elemento;
        Empilha(aux, elemento);
    }
    while (Desempilha(aux, elemento)) {
        Empilha(pilha, elemento);
    }
    if (tamanho == 0) return 0;
    return sum / tamanho;
}

```

**Tarefa 5** - Implemente uma função para retornar o menor elemento de uma pilha de inteiros.

**Protótipo:** int MenorPilha (Pilha).

#### Solução

```

int MenorPilha(Pilha &pilha) {
    int menor;
    int elemento;
    Pilha aux; IniciaPilha(aux);
    while (Desempilha(pilha, elemento)) {
        if (menor == NULL) menor = elemento;
        else if (elemento < menor) menor = elemento;
        Empilha(aux, elemento);
    }
    while (Desempilha(aux, elemento)) {
        Empilha(pilha, elemento);
    }
    return menor;
}

```