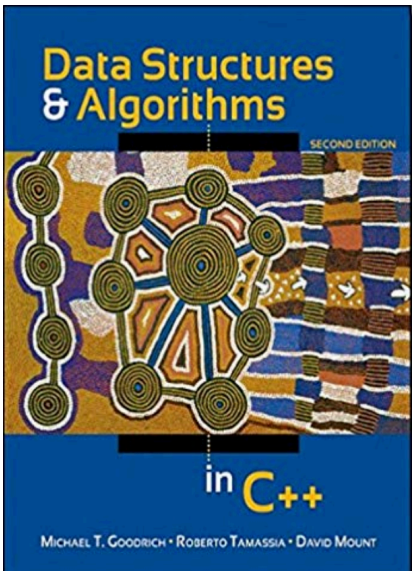
FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
ESTRUTURA DE DADOS I – Aula 09 – 2º SEMESTRE/2018
PROF. LUCIANO SILVA/MARCIO PORTO FEITOSA

TEORIA: PILHAS

Nossos **objetivos** nesta aula são:

- conhecer o tipo abstrato de dados pilha
- conhecer e implementar a estrutura de dados pilha



Para esta aula, usamos como referência a **Seção 5.1 (Stacks)** do nosso livro-texto:

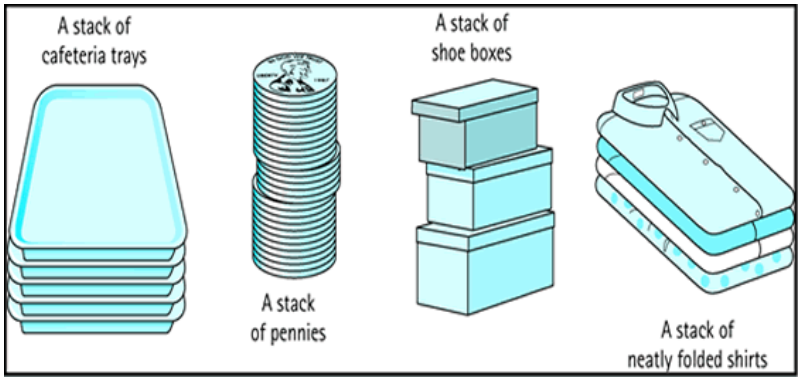
GOODRICH,M., **Data Structures and Algorithms**. 2.ed. New York: Wiley, 2011.

Não deixem de ler esta seção depois desta aula!!

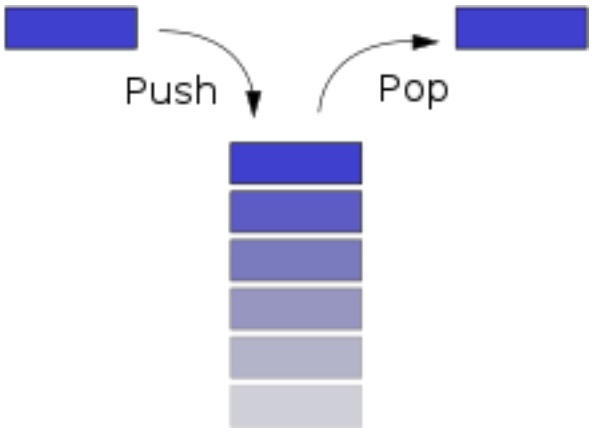
PILHA COMO TIPO ABSTRATO DE DADO

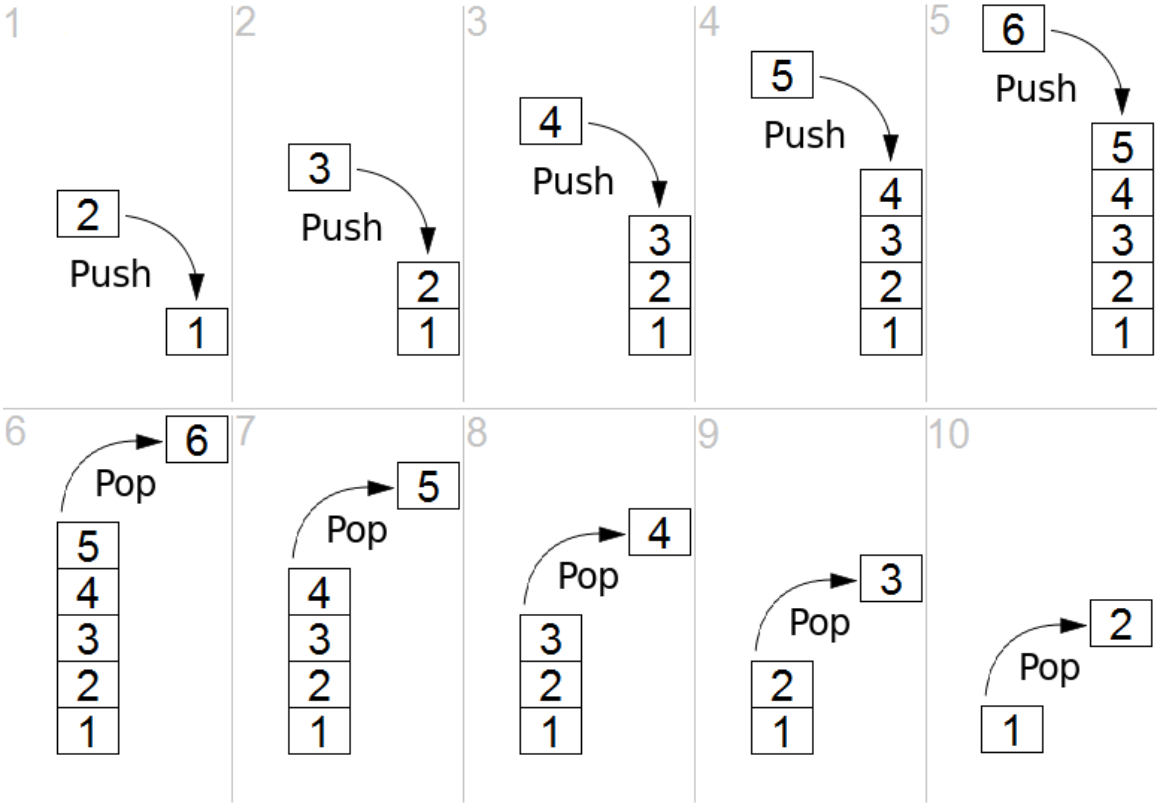
- Uma **pilha (stack)** é uma tipo de dado linear, com uma política de inserção e remoção de elementos bem definida: o primeiro elemento que entra na estrutura é o primeiro que sai. Esta política é conhecida como **LIFO (Last-IN First-Out)**.

- Pilha é usado como uma metáfora a diversas pilhas conhecidas:



- Uma pilha utiliza duas operações básicas: **empilha (push)** (para inserir elementos) e **desempilha (pop)** (para remover elementos):





- Além destas operações básicas, temos também as operações de construção de pilha vazia, verificação de pilha vazia e consulta ao **último elemento inserido na pilha**, chamado **topo**.

IMPLEMENTAÇÃO DA ESTRUTURA DE DADOS PILHA

- Qualquer uma das estruturas lineares que estudamos anteriormente (vetor, lista ligada e lista duplamente ligada) podem ser de base para implementarmos uma pilha. Só precisamos garantir a política de inserção-remoção LIFO.

- A seguir, vamos implementar uma pilha tendo como base uma lista duplamente ligada:

Lista Duplamente Ligada

```
#ifndef LISTA_DUPLAMENTE_LIGADA_H
#define LISTA_DUPLAMENTE_LIGADA_H
#include "dno.h"

template <typename E>
class ListaDuplamenteLigada{

  private:
    DNo<E> * cabeca;
    DNo<E> * fim;
  public:
    ListaDuplamenteLigada();
    ~ListaDuplamenteLigada();
    bool vazia() const;
    const E& inicio() const;
    const E& final() const;
    void insereInicio(const E& e);
    void insereFinal(const E& e);
    void removeInicio();
    void removeFinal();
};

#endif
```

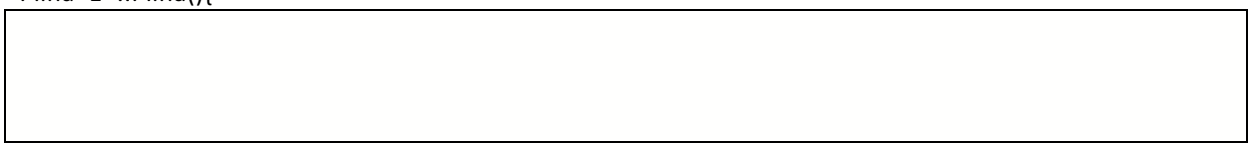
Pilha

```
#ifndef PILHA_H
#define PILHA_H

#include "lista_duplamente_ligada.cpp"

template <typename E>
class Pilha{

 private:
   ListaDuplamenteLigada<E>* p;

 public:
   Pilha();
   ~Pilha();
   bool vazia();
   void empilha(const E& e);
   const E& topo() const;
   void desempilha();
};


#endif
```

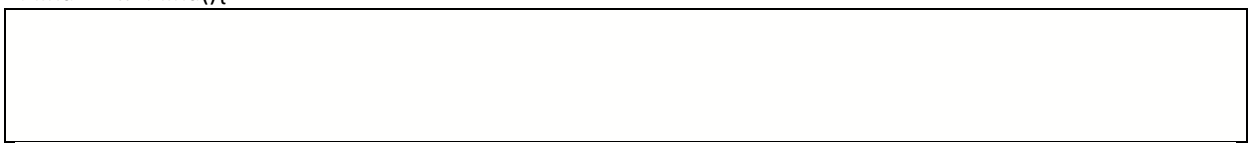
EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Implementar os métodos da estrutura de dados Pilha.

```
#include "pilha.h"

template <typename E>
Pilha<E>::Pilha(){
```

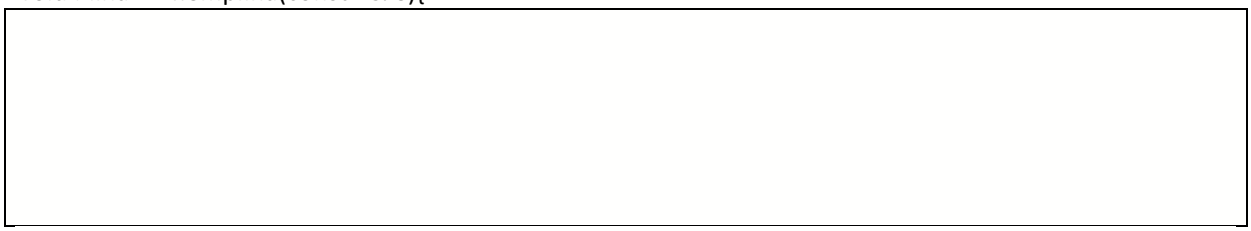


```
}

template <typename E>
Pilha<E>::~Pilha(){
```



```
}

template <typename E>
bool Pilha<E>:: vazia(){
```

```
}

template <typename E>
void Pilha<E>::empilha(const E& e){
```



```
}

template <typename E>
const E& Pilha<E>::topo() const{
```

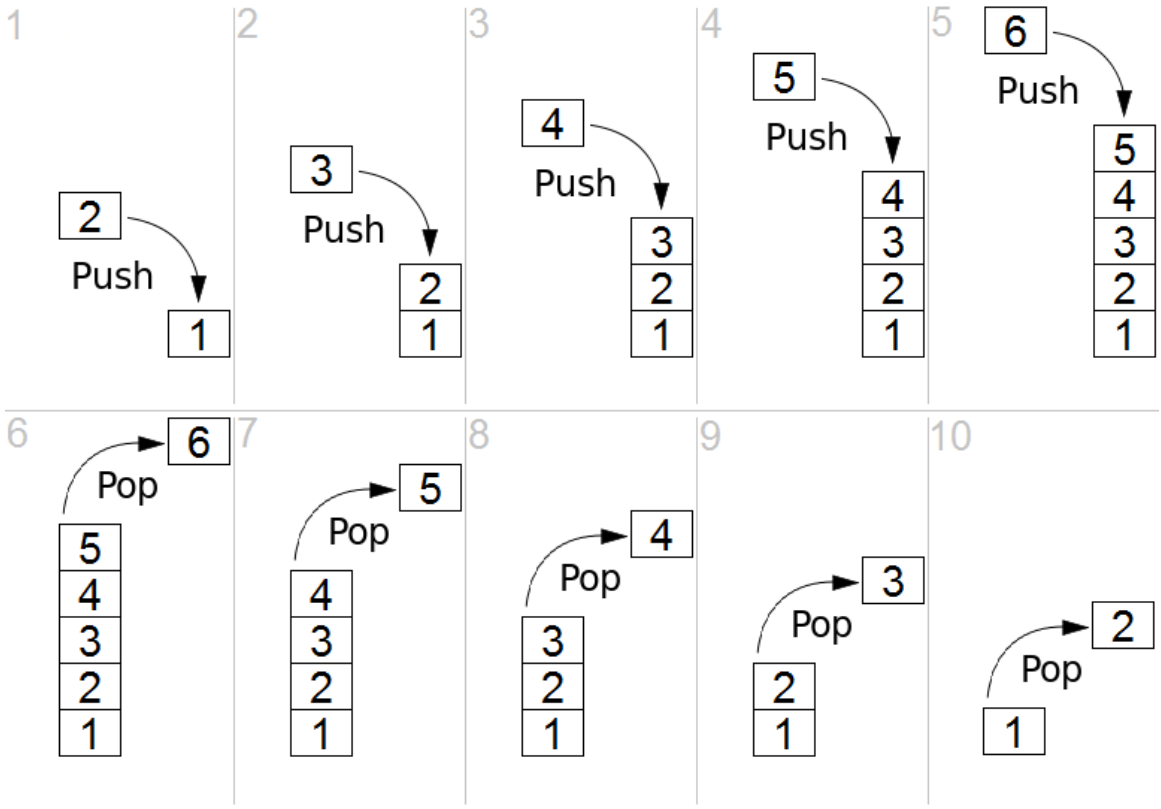
```
}

template <typename E>
void Pilha<E>::desempilha(){
```

```
}
```

EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Teste sua implementação, realizando as operações abaixo no módulo main.cpp:

```
#include <iostream>
#include "pilha.cpp"

int main() {
  Pilha<int> p;









}
```



APLICAÇÃO: RESOLUÇÃO DE EXPRESSÕES NA NOTAÇÃO POLONESA REVERSA (RPN)

A notação polonesa reversa (RPN), também chamada notação posfixa, coloca os operadores após os operandos em expressões aritméticas. Por exemplo:

2+20*4 (notação infixa) → 2 20 4 * + (notação posfixa)

RPN é o sistema usado por calculadoras HP para resolução de expressões matemáticas. Seu objetivo neste exercício será, utilizando uma pilha, obter o valor da expressão posfixa 2 20 4 * + com uma pilha.

EXERCÍCIOS EXTRA-CLASSE

1. Implementar uma pilha com base na estrutura Vetor.

2. Implementar uma pilha com base na estrutura Lista Ligada Simples.

3. Construir uma classe Solver, que utilize como atributo uma Pilha, que disponibilize um método chamado **double solve(String expr)**, que recebe uma expressão na notação RPN e devolve o seu valor no formato double.

4. A notação prefixa trabalha de modo contrário à posfixa: os operadores aparecem antes dos operandos. Por exemplo, 2+20*4 (notação infixa) → + * 20 4 2 (notação prefixa). Utilizando uma pilha, converta uma expressão da notação prefixa para posfixa.