Estrutura de Dados

Pilha

Profa. Ana Cristina dos Santos
email: ana.csantos@sp.senac.br

Estrutura de Dados Pilha

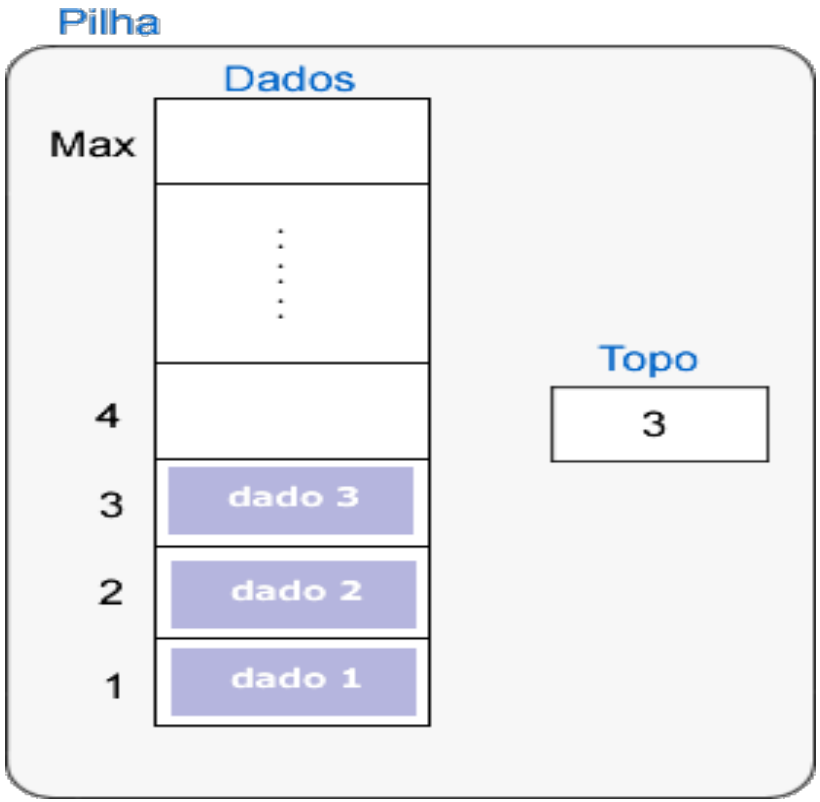
- Uma **Pilha** (=*Stack*) é uma Estrutura de Dados baseada na política **LIFO** (**Last-In-First-Out**), na qual os dados que foram inseridos primeiros na pilha serão os últimos a serem removidos. (**Relacionamento lógico entre dados**)

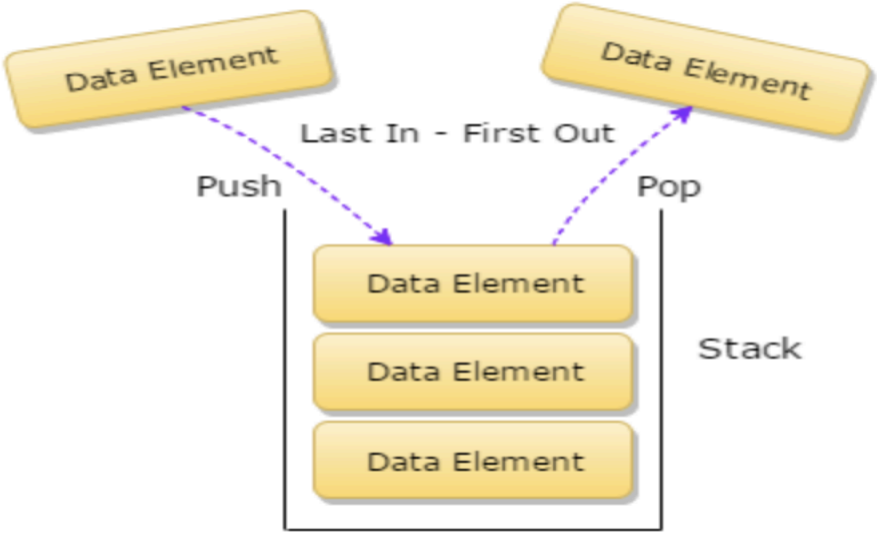
- Uma **Pilha** é uma coleção ordenada de dados ou itens, em que a inserção de novos itens e a remoção de itens existentes ocorrem na mesma extremidade, chamada **topo.** A extremidade oposta é chamada de **base**.

Estrutura de Dados Pilha

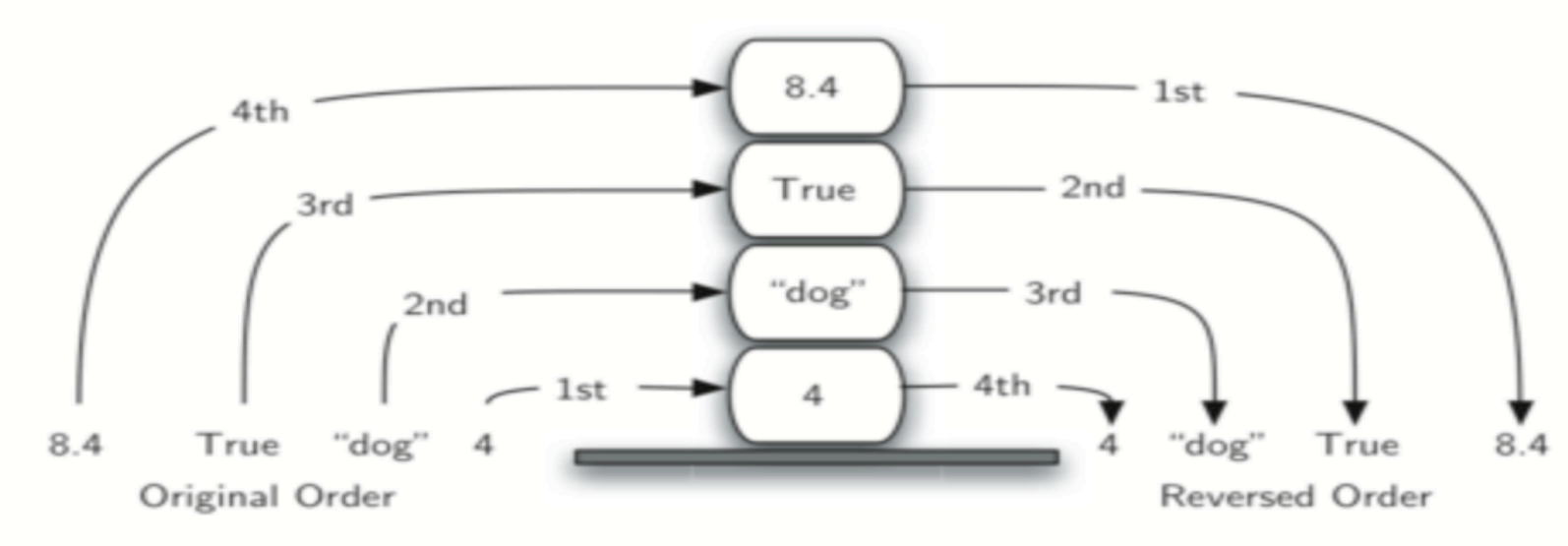
- Uma Pilha tem as seguintes operações (**Manipulação**):
  - empilhamento que **insere** um dado no topo da Pilha,
  - o desempilhamento que **remove** um item do topo da pilha,
  - também deve ser possível **consultar** se a Pilha está vazia.

Estrutura de Dados Pilha





Estrutura de Dados Pilha



Definindo um TAD - Pilha

- Para representar a Estrutura de Dados Pilha como um TAD precisamos especificar um **conjunto de dados** e **operações** que podem ser executadas sobre esses dados, os conjuntos de dados poderia ser uma lista de elementos da **Pilha**, armazenados em uma lista.

Definindo um TAD - Pilha

- **Operações**:
  - **Empilhamento**: insere o elemento e no topo da pilha - **push(item)**
  - **Desempilhamento**: remove o elemento do topo da pilha e o retorna - **pop(),**
  - **Consultar**: retorna um booleano indicando se a pilha está vazia - `isEmpty()`,
  - **obter** o tamanho da pilha **size()**,
  - **obter** o elemento do topo, sem remover, `top()` e
  - **criação** da Pilha será feita no construtor da classe.

TAD Pilha – Implementação

- Existem várias opções de estruturas de dados que podem ser usadas para
- representar pilhas.

- As duas representações mais utilizadas são as implementações por meio de vetores e de lista encadeada.

TAD Pilha – Implementação com Vetor

```
public class Pilha {
      private Object item[] ;
      private int topo;

public Pilha ( int maxTam) { // Cria uma Pilha vazia
      if(this.vazia())
      this.item = new Object[maxTam] ;
      this.topo = 0;
}
```

TAD Pilha – Implementação com Vetor

```
public void empilha (Object x) throws Exception {
       if (this.topo == this.item.length)
         throw new Exception ( "Erro : A pilha esta cheia" );

      this.item[this.topo++] = x;
 }
```

TAD Pilha – Implementação com Vetor

```
public Object desempilha() throws Exception {
      if (this.topo == 0)
      throw new Exception ( "Erro : A pilha está vazia" );
      this.item[--this.topo];
}
public boolean vazia() {
      return (this.topo == 0);
}
public int tamanho () {
   return this.topo;
}
```

TAD Pilha – Implementação com Lista Encadeada

```
public class No {
      private Object elemento;
      private No prox;

public No(Object elemento, No prox ){
      this.elemento = elemento;
      this.prox = prox;
}
```

TAD Pilha – Implementação com Lista Encadeada

- Podemos decidir onde ficará o topo da pilha, se no início ou no final da lista. A melhor opção é inserir e remover em tempo constante, ou seja, no início da lista.
- A fim de termos a operação de size() em tempo constante, vamos inserir um o número corrente de elementos em uma variável de instância.

```
public class Pilha {
      private No topo;
      private int tam;
public Pilha(){ //Cria uma pilha vazia
      this.topo = null;  this.tam = 0;
}
```

TAD Pilha – Implementação com Lista Encadeada

```
// Operações
public void empilha (Object x) {
      //Inserção no início
      No aux = this.topo;
      this.topo = new No();
      this.topo.elemento = x;
      this.topo.prox = aux;
      this.tam++;
}
```

TAD Pilha – Implementação com Lista Encadeada

```
Public Object desempilha () throws Exception {
  if (this.vazia())
      throw new Exception ( "Erro : A pilha esta vazia" );
  Object elemento = this.topo.elemento;
  this.topo = this.topo.prox;
  this.tam--;
  return elemento;
}
```

TAD Pilha – Implementação com Lista Encadeada

```
public boolean vazia() {
      return (this.topo == null);
}
public int tamanho () {
      return this.tam;
}
```

Exercício: Parênteses e colchetes aninhados

- Suponha que queremos decidir se uma dada sequência de parênteses e colchetes está bem formada. Por exemplo:

  (( ) [( )]) está bem formada, enquanto

  ([ )] não está bem formada.

  ((( )) não está bem formada.

- Como resolver esse problema ? Daria pra usar uma Pilha ?

Exercícios

1) Qual seria a **complexidade de tempo** para as operações e inserção e remoção na Pilha ?

2) Construa um programa que leia 10 valores e empilhe-os conforme forem **pares ou ímpares** na pilha1 e pilha2 respectivamente. No final desempilhe os valores de cada pilha mostrando-os na tela.

Exercícios

3) Escreva uma função que receba um número inteiro e positivo representando um número **decimal**, determine o seu equivalente **binário**.

Exemplo: Dado 18 a saída deverá ser 10010.
Utilize uma Pilha no processo de conversão.

Exercícios

4) Escreva uma função que recebe uma String e usando uma pilha inverte as letras de cada palavra da String preservando a ordem das palavras. Note que sempre a String é finalizada por ponto, por exemplo, dado o texto:

ESTE EXERCÍCIO É MUITO FÁCIL.

a saída deve ser:

ETSE OICÍCREXE É OTIUM LICÁF.

Exercícios

5) Uma palavra é **palíndromo** se a sequência de caracteres que a constitui é a mesma quer seja lida da esquerda para a direita ou da direita para a esquerda.
Por exemplo, RADAR e MIRIM são palíndromos.
Escreva uma programa para reconhecer se uma dada palavra é palíndromo.