Estrutura de Dados
Pilha

Fabio Lubacheski
fabio.aglubacheski@sp.senac.br

Problema: parênteses e colchetes aninhados

- Considere o problema abaixo:

  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 ?
- Alguma forma de armazenamento poderia nos ajudar ?

O que seria uma Estrutura de Dados ?

- Sabe-se que programas manipulam dados. Quando estes dados estão organizados (dispostos) de forma coerente caracterizam uma **Estrutura de Dados**.

- Segundo a wikipedia, uma **Estrutura de Dados** é um modo particular de armazenamento e organização de dados em um programa de modo que possam ser usados **eficientemente**, facilitando sua busca e alteração.

O que é uma Estrutura de Dados

- As Estruturas de Dados diferem umas das outras pelo **relacionamento** e **manipulação** de seus dados.
  - **Relacionamento**: como os dados estão organizados.
  - **Manipulação**: operações para a manipulação de seus dados. Exemplos de operações: **inserção** de um novo elemento ou **remoção** de um elemento.

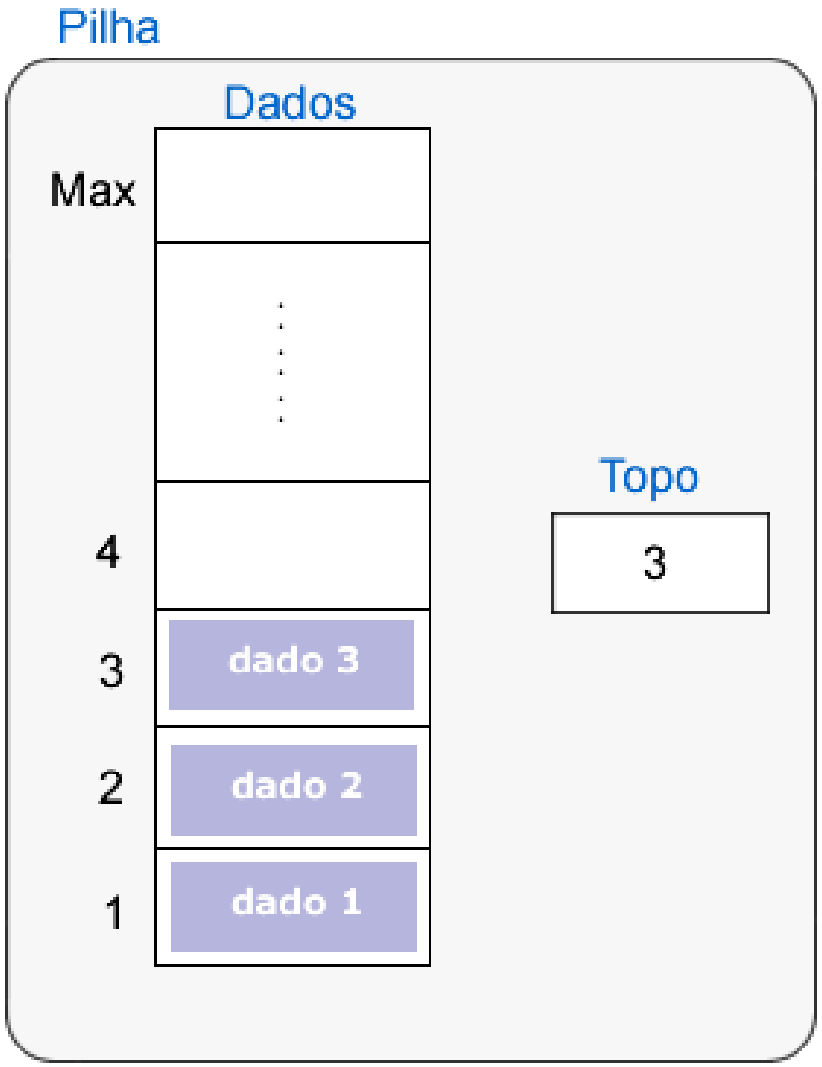
- **Diferentes tipos de Estrutura de Dados** são adequadas a diferentes **tipos de aplicações** e algumas são altamente especializadas, destinando-se a algumas tarefas específicas.

Estrutura de dados Pilha

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

- A Estrutura de Dados **Pilha** tem as seguintes operações (**Manipulação**):
  - **Empilhamento** que **insere** um dado no topo da **Pilha**,
  - **Desempilhamento** que **remove e devolve** um item do topo da pilha.
  - Além disso, precisaremos saber quando a pilha está **vazia** ou **cheia**.

Pilha



Definindo uma estrutura de dados Pilha

- Para implementar uma estrutura de dados Pilha pode-se usar **vetor** ou listas encadeadas (**???**). Como numa Pilha, a manipulação dos elementos é realizada em apenas em uma extremidade, chamada de topo, poderíamos ter o a seguinte declaração para definir uma **estrutura de dados Pilha** em Java

```
// vetor para representar a Pilha
// supondo que os elementos da Pilha sejam
// caracteres
char elementos[]=new char[TAM];
// variável com o topo da Pilha
// inicialmente a Pilha está vazia
int topo=-1;
```

Definindo uma estrutura de dados Pilha

- A operação de **empilhamento**, que insere um elemento na Pilha, pode ser feita da seguinte forma:

```
// sobe o topo
topo++;
//insere o elemento no topo
elementos[topo]=elemento;
```

- A operação de **desempilhamento,** que remove e devolve um item do topo da pilha, pode ser feita da seguinte forma:

```
//retira elemento do topo
elemento=elementos[topo];
// desce o topo da pilha
topo--;
```

Definindo um TAD - Pilha

- **Armazenamento**: Um TAD Pilha teria os seguintes atributos com **visibilidade interna** (**encapsulados**). A **criação** da Pilha será feita no construtor da classe do TAD.

```
private char elementos[];
private int topo;
```

- **Manipulação**: Para operação de **empilhamento** teremos o método **`void push(char elemento)`**, para o **desempilhamento** temos o método **`char pop()`** e ainda é possível **consultar** se a pilha está vazia **`boolean isEmpty()`** ou cheia **`boolean isFull()`**.

Definindo um TAD - Pilha

Agora é só implementar o TAD Pilha

Exercícios

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

2) Escreva uma versão recursiva para a função `bemFormada()`, sem a utilização de uma Pilha como estrutura de dados.

3) 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

4) Sejam os inteiros 1, 2, e 3 que são lidos nesta ordem para serem colocados numa *Pilha*. Considerando-se todas as possíveis seqüências de operações *Incluir* e *Retira*r, decida quais das 6 (3!) permutações possíveis podem ser obtidas a partir dessas operações. Por exemplo, a permutação 2, 3, 1 pode ser obtida da seguinte forma:

Inclui 1
Inclui 2
Retira 2
Inclui 3
Retira 3
Retira 1

De um modo geral, uma permutação é chamada *admissível* quando ela puder ser obtida mediante uma sucessão de inclusões e remoções em uma pilha a partir da permutação 1, 2, ..., *n*. Assim, por exemplo, a permutação 2, 3, 1 é admissível. Pede-se:

a) Determinar a permutação correspondente a *IIIRRIRR*, *n*=4

b) Dê um exemplo de permutação não admissível.

Exercícios

5) Imagine que as palavras de uma certa linguagem são formadas somente pelas letras *a*, *b* e *c*. As palavras seguem o seguinte formato *WcM*, onde *W* é uma sequência de letras que só contém *a* e *b* e *M* é o reverso de *W*, ou seja, *M* é *W* lido de trás para frente. Exemplo:

c, aca, bcb, abcba, bacab, aacaa, bbcbb, . . .

Escreva um método que recebe uma palavra e, usando uma **Pilha**, determina se a palavra está no formato certo (*WcM*) ou não, se estiver o método retorna **true** e caso não esteja retorna **false**.

6) 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.

Fim