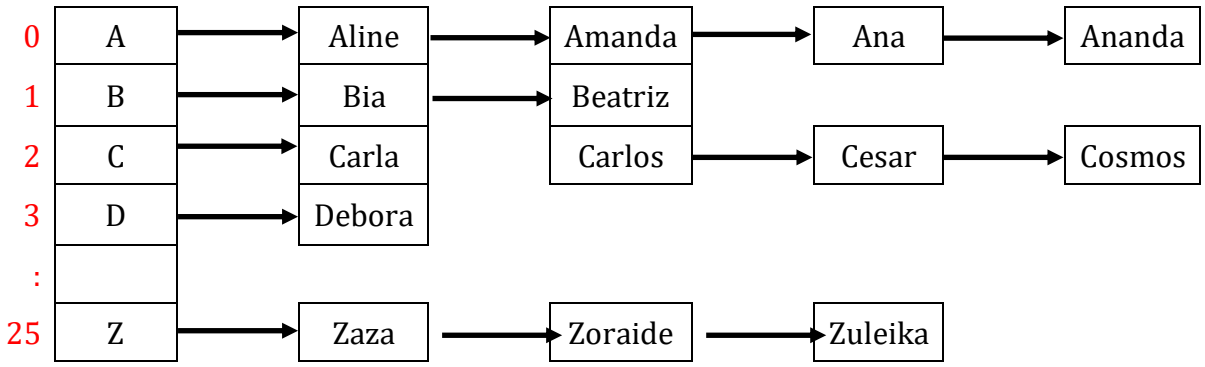
Estrutura de Dados

Laboratório de Matrizes, Listas, Filas, Pilhas

Profa. Eliane Oliveira Santiago

Exercício 1

1. Criar uma estrutura de dados para armazenar nomes organizados em ordem alfabética. Para garantir a classificação, a estrutura de dados deverá ser organizada com um vetor de índices e uma lista duplamente encadeada contendo os nomes armazenados. O vetor de índices faz referência a inicial dos nomes e a lista duplamente encadeada guarda os nomes em ordem alfabética, conforme exemplo abaixo:



2. Criar uma função com a assinatura `int indice(String nome)` que receberá um nome como parâmetro de entrada e retornará o índice correspondente do vetor de índice para a sua estrutura de dados. Sua função deverá capturar a primeira letra do nome e retornar o índice do vetor onde a letra será armazenada.
3. Cada posição do vetor deverá guardar uma lista encadeada de nomes, indexada de acordo com a letra inicial.
4. Implemente as operações com listas, considerando as adaptações do modelo.
    a. Uma função para adicionar um nome na sua estrutura de dados. Ao adicionar um nome, a sua função deverá identificar qual lista encadeada o nome deverá ser inserido, de acordo com o vetor de índices, e inserir o nome na lista, garantindo a classificação.
    b. Uma função para pesquisar se um determinado nome existe na sua estrutura de dados.
    c. Uma função para excluir um determinado nome da sua Estrutura de Dados. Observe que a exclusão não poderá quebrar o encadeamento da lista.
    d. Uma função para renomear um determinado nome da lista. Ao renomear, sua função deverá garantir a classificação.
    e. Uma função que retornará se a sua estrutura de dados está vazia. Sua estrutura de dados está vazia quando as 26 listas estão vazias, ou seja, não há nomes cadastrados.
    f. Uma função para retornar a quantidade de nomes cadastrados na estrutura de dados.

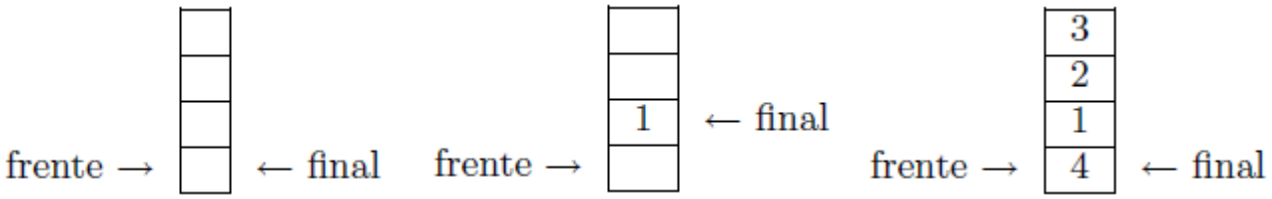
Estrutura de Dados

Laboratório de Matrizes, Listas, Filas, Pilhas

Profa. Eliane Oliveira Santiago

Exercício 2

Considere uma implementação de fila utilizando vetor circular, que utiliza apontadores (índices) para a frente e o final da fila: frente aponta para a posição imediatamente anterior ao primeiro elemento da fila e final aponta para o último elemento inserido, se existir.



(a) Fila Vazia **(b) Fila com um elemento** **(c) Fila com quatro elementos**

a) Faça a implementação da Fila com os dois apontadores.
b) Crie os métodos abaixo para a fila:
   void enfileira(Object obj)
   Object desenfileira()
   estaVazia()

a) `criaFila()` que inicia os valores de **`frente`** e **`final`**. Poderá ser um método construtor.
b) `void enfileira(Object obj)` enfileira o obj no fim da fila.
c) `Object desenfileira()` retorna o objeto do final da fila, retirando-o da fila.
d) `Object cabeca()` retorna o primeiro objeto da fila sem retirá-lo.
e) `Object cauda()` retorna o objeto do final da fila, sem retira-lo da fila.
f) `boolean vazia()` retorna true se a fila está vazia e falso caso contrário.
g) `boolean cheia()` retorna true se a fila está cheia e falso caso contrário.
h) `void mostrarFila()` escreve todos os elementos da fila.
i) `boolean pesquisa(Object obj)` pesquisa se o objeto está na fila. Retorna true se encontrou e false, caso contrário.
j) `int tamanho()` retorna o tamanho da fila.

Comente a dificuldade para se diferenciar fila cheia de fila vazia.

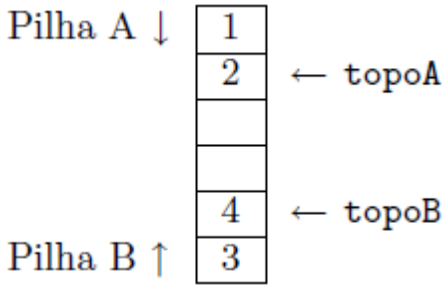
Estrutura de Dados

Laboratório de Matrizes, Listas, Filas, Pilhas

Profa. Eliane Oliveira Santiago

Exercício 3

Duas pilhas A e B podem compartilhar o mesmo vetor, como esquematizado na figura abaixo.



Crie os métodos abaixo para a estrutura de dados que representa a pilha descrita acima.

a) `criaPilha()` que inicia os valores de **`topoA`** e **`topoB`**. Poderá ser um método construtor.
b) `void empilhaA(Object obj)` empilha o obj no topo da pilha A
c) `void empilhaB(Object obj)` empilha o obj no topo da pilha B
d) `boolean vaziaA()` retorna true se a pilha A está vazia e falso caso contrário.
e) `boolean vaziaB()` retorna true se a pilha B está vazia e falso caso contrário.
f) `Object desempilhaA()` retorna o objeto empilhado na PilhaA, retirando-o da pilha.
g) `Object desempilhaB()` retorna o objeto empilhado na PilhaB, retirando-o da pilha.
h) `boolean cheiaA()` retorna true se a pilha A está cheia e falso caso contrário.
i) `boolean cheiaB()` retorna true se a pilha B está cheia e falso caso contrário.
j) `Object topoA()` retorna o objeto empilhado no topo da pilha A, sem retirá-lo.
k) `Object topoB()` retorna o objeto empilhado no topo da pilha B, sem retirá-lo.
l) `void mostrarPilhaA()` escreve todos os elementos da pilha A.
m) `void mostrarPilhaB()` escreve todos os elementos da pilha B.
n) `int tamanhoA()` retorna o tamanho da pilha A.
o) `int tamanhoB()` retorna o tamanho da pilha B.

Responda:
Na estrutura de dados que você implementou, é possível que uma pilha esteja cheia e a outra não?