ESTRUTURA DE DADOS

Curso de Licenciatura em Ciências da Computação

Lucas Sampaio Leite

Tipos abstratos de dados

INSTITUTO FEDERAL
Baiano

- Um Tipo de Dado Abstrato (TDA) é uma especificação conceitual que define um tipo de dado a partir de seu comportamento, ou seja, das operações disponíveis e das regras que essas operações devem obedecer, sem descrever como elas são implementadas.
- Em outras palavras, o TDA define o que o tipo de dado faz, mas não como ele faz.
    - Exemplos de TADs incluem pilha, fila, lista, conjunto, mapa, entre outros.

Tipos abstratos de dados

INSTITUTO FEDERAL
Baiano

- Um Tipo de Dado Abstrato (TDA) é uma especificação conceitual que define um tipo de dado a partir de seu comportamento, ou seja, das operações disponíveis e das regras que essas operações devem obedecer, sem descrever como elas são implementadas.
- Em outras palavras, o TDA define o que o tipo de dado faz, mas não como ele faz.
    - Exemplos de TADs incluem pilha, fila, lista, conjunto, mapa, entre outros.

Uma estrutura de dados é uma implementação concreta de um TAD em uma linguagem de programação.

Tipos abstratos de dado - Lista

INSTITUTO FEDERAL
Baiano

- As operações comuns de um Tipo Abstrato de Dados (TAD) do tipo Lista envolvem um conjunto de funcionalidades que permitem armazenar, acessar e modificar elementos de forma organizada.
  - Inserir um elemento;
  - Remover um elemento;
  - Obter um elemento dada uma posição;
  - Verificar se a lista está vazia;
  - Obter o tamanho da lista;
  - Limpar a lista;

Tipos abstratos de dado - Lista

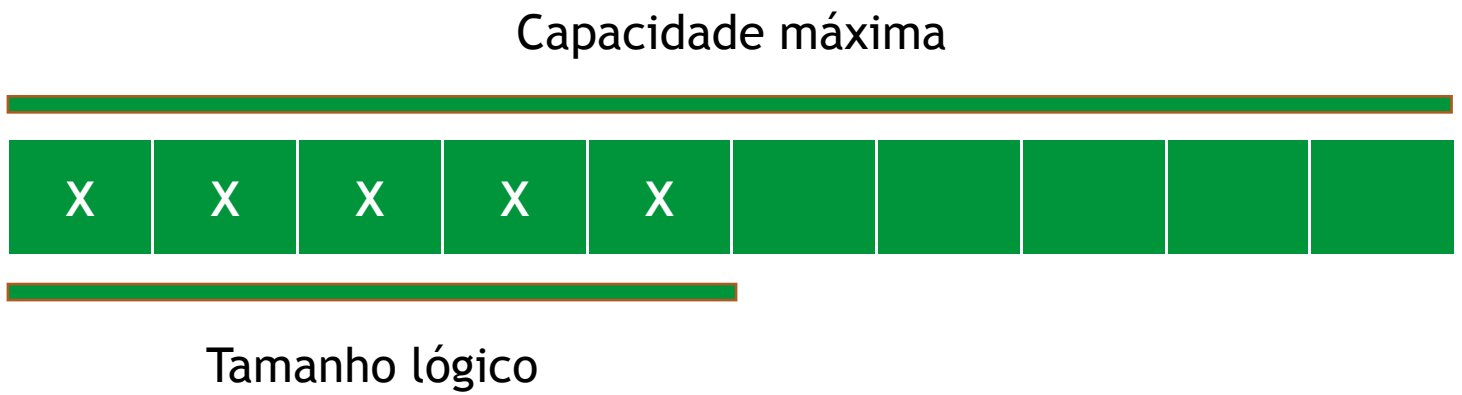
INSTITUTO FEDERAL
Baiano

- As operações comuns de um Tipo Abstrato de Dados (TAD) do tipo Lista envolvem um conjunto de funcionalidades que permitem armazenar, acessar e modificar elementos de forma organizada.
  - Verificar se um elemento está na lista;
  - Iterar sobre os elementos da lista;
  - Inserir todos os elementos de outra lista em uma lista existente;
  - Inserir um elemento em uma posição específica;
  - Remover todos os elementos que satisfazem uma condição específica;
  - Obter o índice de um elemento.

Listas implementadas a partir de vetores

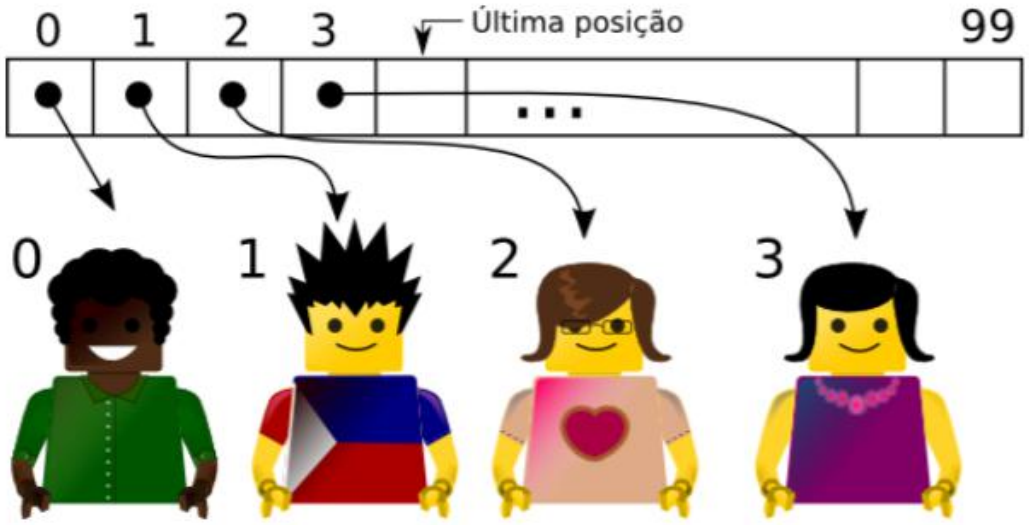
INSTITUTO FEDERAL
Baiano

- Listas implementadas a partir de vetores são estruturas lineares em que os elementos são armazenados em posições contíguas de memória.
- Nessa abordagem, utiliza-se um vetor com capacidade máxima previamente definida e uma variável de controle chamada tamanho lógico, que indica quantas posições estão efetivamente ocupadas.

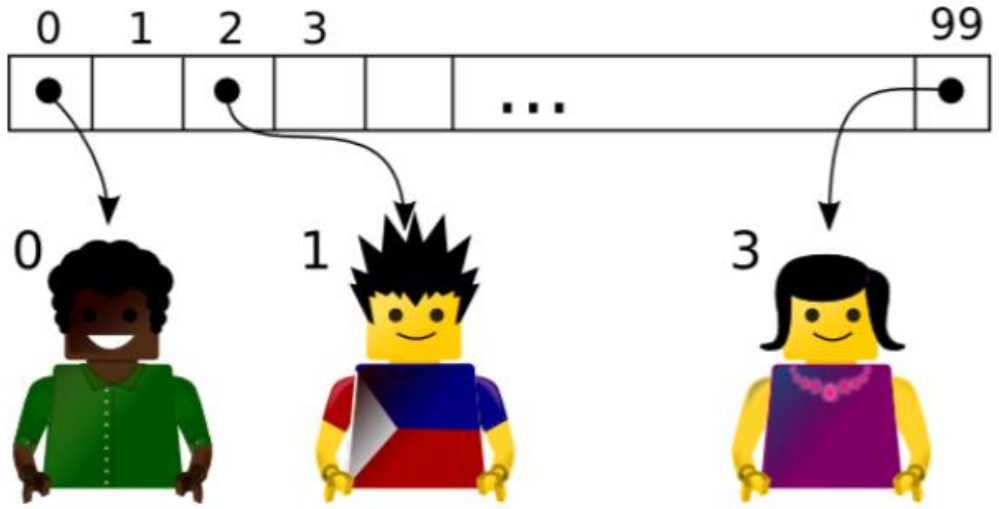


Listas implementadas a partir de vetores

- Um vetor compactado à esquerda é fundamental em uma implementação de lista baseada em vetor, pois garante que todos os elementos sejam armazenados em posições consecutivas, sem “lacunas” entre eles.
- Sem a compactação à esquerda, a lista pode apresentar espaços vazios, comprometendo a eficiência e a coerência das operações definidas pelo TAD.



Vetor compactado a esquerda



Vetor não compactado a esquerda

Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

- Definindo uma lista para armazenar pessoas:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 100

typedef struct {
    int id;
    char nome[50];
    int idade;
} Pessoa;

typedef struct {
    Pessoa dados[MAX];
    int tamanho;
} Lista;
```

Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

- Em uma lista baseada em vetor, é fundamental inicializar o tamanho lógico da lista antes de utilizá-la.

```
void inicializarLista(Lista *lista) {
    lista->tamanho = 0;
}
```

Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

- Para adicionar um elemento no final de uma lista baseada em vetor, utilizamos o campo tamanho, que indica a próxima posição livre válida.
- A inserção ocorre na posição tamanho, seguida da atualização do tamanho lógico da lista.

```
int adicionarFinal(Lista *lista, Pessoa p) {
    if (lista->tamanho >= MAX) {
        return 0;
    }

    lista->dados[lista->tamanho] = p;
    lista->tamanho++;
    return 1;
}
```

Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

- Para exibir os elementos armazenados em uma lista baseada em vetor, deve-se percorrer apenas as posições válidas, ou seja, do índice 0 até tamanho - 1.

```
void imprimirLista(Lista *lista) {
    for (int i = 0; i < lista->tamanho; i++) {
        printf("Posicao %d -> ID: %d | Nome: %s | Idade: %d\n",
               i,
               lista->dados[i].id,
               lista->dados[i].nome,
               lista->dados[i].idade);
    }
}
```

Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

- A função responsável por retornar o tamanho da lista fornece a quantidade de elementos atualmente armazenados, isto é, o tamanho lógico da estrutura.

```
int tamanhoLista(Lista *lista) {
    return lista->tamanho;
}
```

Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

- A função contemElemento verifica se existe uma pessoa com determinado nome armazenada na lista.
- A busca é realizada percorrendo as posições válidas do vetor e comparando os nomes.

```
int contemElemento(Lista *lista, char nome[]) {
    for (int i = 0; i < lista->tamanho; i++) {
        if (strcmp(lista->dados[i].nome, nome) == 0) {
            return 1;
        }
    }
    return 0;
}
```

Listas implementadas a partir de vetores

Em C, a função strcmp é utilizada para comparar strings porque o operador == verifica apenas os endereços de memória, e não o conteúdo armazenado. A função strcmp realiza a comparação caractere por caractere, garantindo que a verificação seja feita com base no texto das strings.

```c
int contemElemento(Lista *lista, char nome[]) {
    for (int i = 0; i < lista->tamanho; i++) {
        if (strcmp(lista->dados[i].nome, nome) == 0) {
            return 1;
        }
    }
    return 0;
}
```

Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

- A função recuperarPorPosicao permite acessar um elemento da lista a partir de seu índice, retornando o elemento apenas se a posição informada estiver dentro dos limites válidos da lista.

Quando uma posição é válida?

Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

- A função recuperarPorPosicao permite acessar um elemento da lista a partir de seu índice, retornando o elemento apenas se a posição informada estiver dentro dos limites válidos da lista.

```
Pessoa recuperarPorPosicao(Lista *lista, int posicao) {
    Pessoa vazio = {0, "", 0};

    if (posicao < 0 || posicao >= lista->tamanho) {
        return vazio;
    }

    return lista->dados[posicao];
}
```

Quando uma posição é válida?

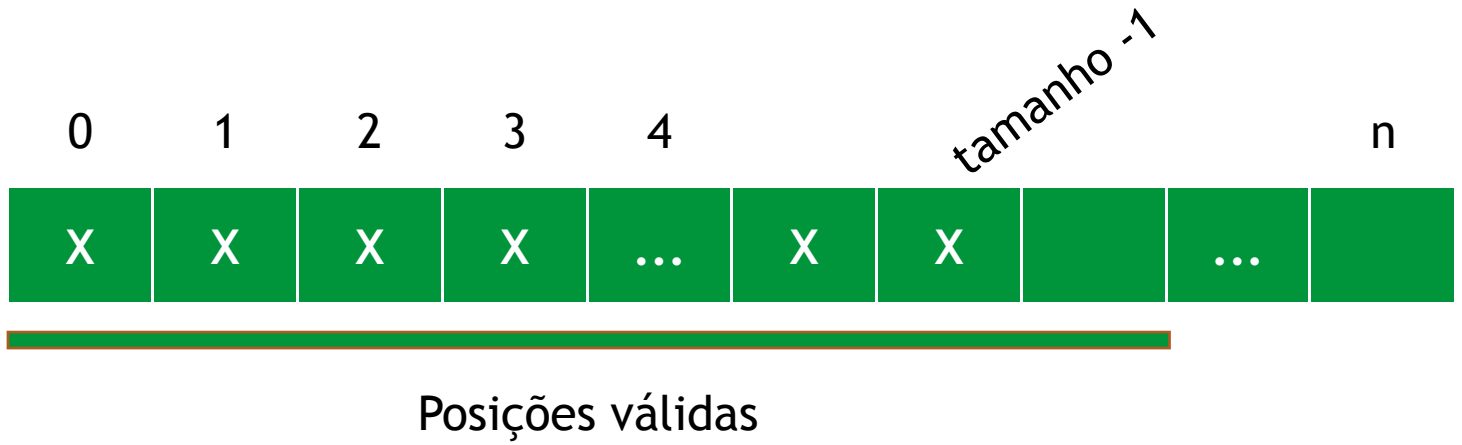
Listas implementadas a partir de vetores

- Adicionar um elemento em uma posição específica é uma operação que consiste em inserir o elemento no índice informado, podendo exigir o deslocamento dos elementos posteriores para manter a ordem da lista.
- Antes da inserção, é fundamental verificar se a posição é válida.

Quando uma posição é válida?

Listas implementadas a partir de vetores

- Adicionar um elemento em uma posição específica é uma operação que consiste em inserir o elemento no índice informado, podendo exigir o deslocamento dos elementos posteriores para manter a ordem da lista.
- Antes da inserção, é fundamental verificar se a posição é válida.



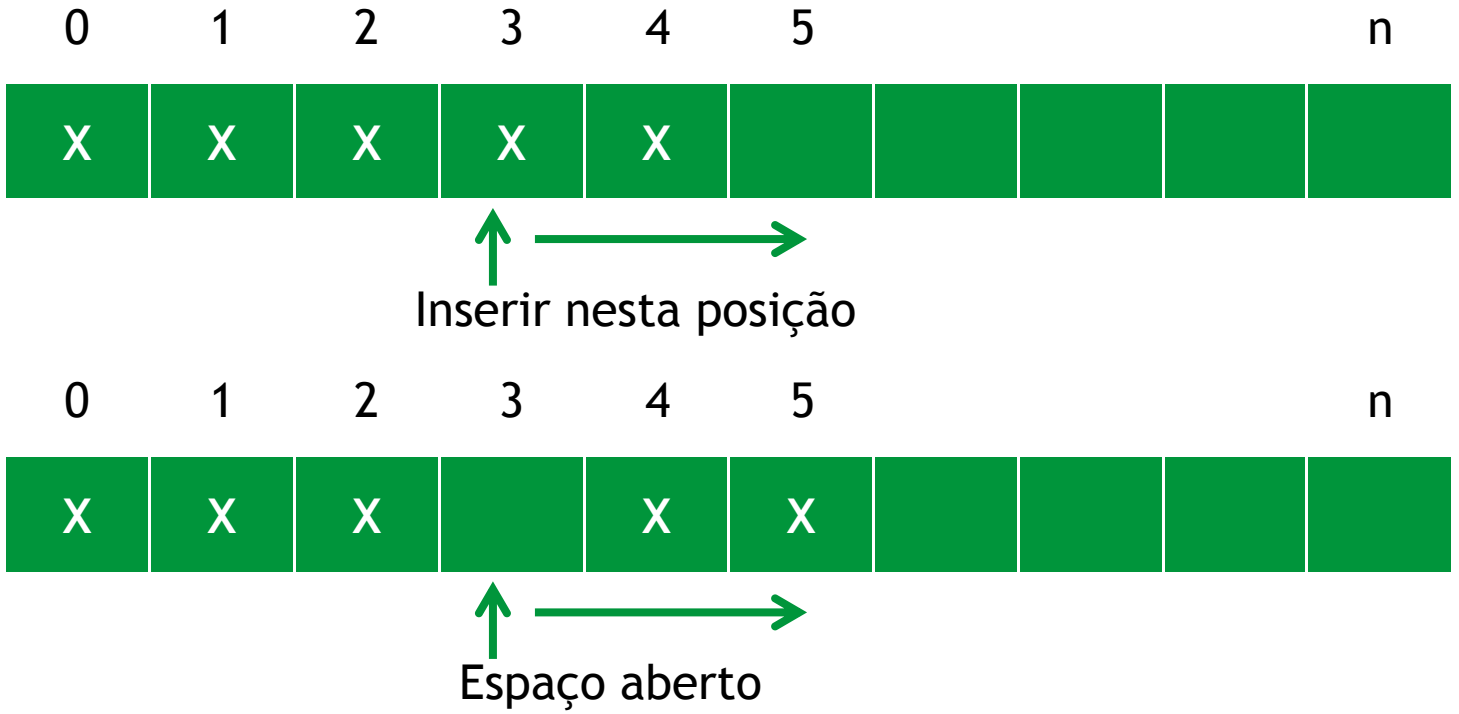
Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

- Para que o vetor permaneça compactado à esquerda, o elemento deve ser inserido em uma posição válida da lista, o que pode exigir o deslocamento dos elementos subsequentes.
- Além disso, é necessário verificar se a posição informada está dentro do intervalo [0, tamanho], garantindo que a inserção preserve a consistência da estrutura.

Listas implementadas a partir de vetores

- Após verificar que a posição é válida, deve-se assegurar que nenhum elemento será sobrescrito indevidamente. É necessário deslocar todos os elementos à direita da posição de inserção (shift right), abrindo espaço para armazenar o novo objeto e mantendo o vetor compactado à esquerda.



Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

```c
int inserirEmPosicao(Lista *lista, Pessoa p, int posicao) {
    if (lista->tamanho >= MAX) {
        return 0;
    }

    if (posicao < 0 || posicao > lista->tamanho) {
        return 0;
    }

    for (int i = lista->tamanho; i > posicao; i--) {
        lista->dados[i] = lista->dados[i - 1];
    }

    lista->dados[posicao] = p;
    lista->tamanho++;
    return 1;
}
```

Listas implementadas a partir de vetores

- Remover um elemento em uma posição específica exige, inicialmente, verificar se o índice informado está dentro dos limites válidos da lista. Se a posição for válida, o elemento pode ser removido.
- Em seguida, para manter o vetor compactado à esquerda, é necessário deslocar todos os elementos posteriores uma posição para a esquerda (shift left), preservando a ordem e a consistência da estrutura.

Quando uma posição é válida?

Listas implementadas a partir de vetores

Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

```c
int removerPorPosicao(Lista *lista, int posicao) {
    if (posicao < 0 || posicao >= lista->tamanho) {
        return 0;
    }

    for (int i = posicao; i < lista->tamanho - 1; i++) {
        lista->dados[i] = lista->dados[i + 1];
    }

    lista->tamanho--;
    return 1;
}
```

Listas implementadas a partir de vetores

INSTITUTO FEDERAL
Baiano

- Listas implementadas a partir de vetores, são estruturas de dados que fornecem uma maneira dinâmica de armazenar e manipular conjuntos de elementos.
- Principais vantagens:
  - Simplicidade de implementação;
  - Desempenho de acesso aleatório.
- Principais desvantagens:
  - Desempenho de inserção e remoção;
  - Redimensionamento.

Exercícios

INSTITUTO FEDERAL
Baiano

1. Implemente uma Lista baseada em vetor capaz de armazenar inicialmente até 10 carros, contemplando todas as operações apresentadas em aula (inicialização, inserção no final, inserção por posição, remoção, busca, recuperação por índice, impressão e obtenção do tamanho). Cada carro deve possuir os seguintes campos: placa, marca e modelo.

2. Implemente uma função redimensionar, que deverá ser acionada quando houver tentativa de inserção do 11º elemento. Nesse caso, deve-se criar um novo vetor com maior capacidade, copiar todos os elementos da lista original para o novo vetor e atualizar a estrutura da lista para referenciar o novo espaço de memória.

Atenção!!!

INSTITUTO FEDERAL
Baiano

- Lista 2 liberada.
- Questões do Beecrowd possuem correção automática (com detecção de plágio).

Dúvidas

INSTITUTO FEDERAL
Baiano

ESTRUTURA DE DADOS

Curso de Licenciatura em Ciências da Computação

Lucas Sampaio Leite