Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital

AULA 11

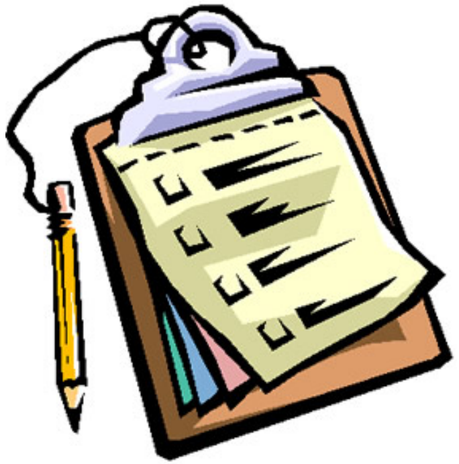
Estrutura de dados básico I (EDB1)

Prof. Msc. Janiheryson Felipe (Felipe)

Natal, RN
2023

OBJETIVOS DA AULA

- Apresentar os conceitos de lista sequencial e sua implementação em memória.
  - Conhecer as listas seguenciais;
  - Implementar uma lista sequencial a partir do zero;

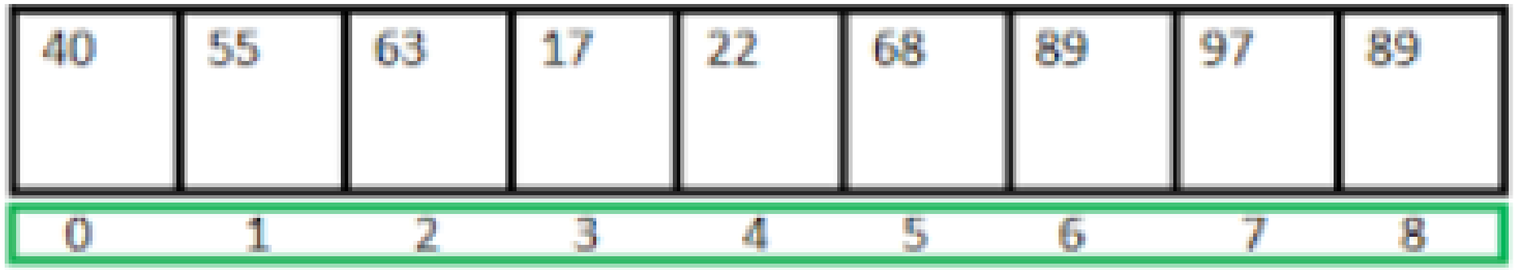


LISTAS SEQUENCIAIS

LISTAS SEQUENCIAIS (LINEARES)

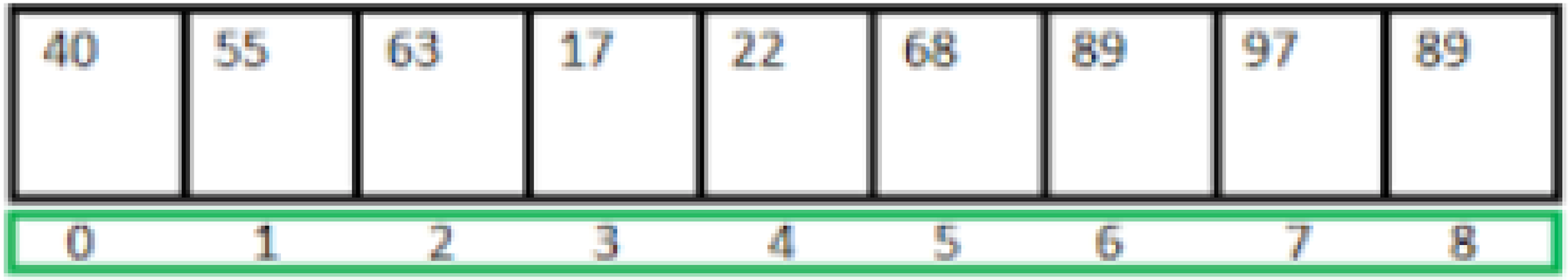
Estrutura de dados na qual **cada elemento é precedido por um elemento e sucedido por outro** (exceto o primeiro que não tem predecessor e o último que não tem sucessor).

Os elementos estão em uma dada ordem (por exemplo, a ordem de inclusão ou ordenados por uma chave).



LISTAS SEQUENCIAIS (LINEARES)

É uma lista linear na qual **a ordem lógica** dos elementos (a ordem "vista" pelo usuário) é a **mesma ordem física** (em memória principal) dos elementos. Isto é, elementos vizinhos na lista estarão em posições vizinhas de memória.



LISTAS SEQUENCIAIS (LINEARES)

Modelagem:

Modelaremos usando um arranjo de registros (struct);
Registros conterão as informações importantes para o usuário; Nosso arranjo terá um tamanho fixo e controlaremos o número de elementos com uma variável adicional.

LISTAS SEQUENCIAIS (LINEARES)

```c
struct Lista {
    int *vetor;
    int tamanho;
    int capacidade;
```

FUNÇÕES DE GERENCIAMENTO

Implementaremos funções para:

- Inicializar a estrutura;
- Retornar a quantidade de elementos válidos;
- Exibir os elementos da estrutura;
- Buscar por um elemento na estrutura;
- Inserir elementos na estrutura;
- Excluir elementos da estrutura;
- Reinicializar a estrutura.

INICIALIZAÇÃO

Para inicializar uma estrutura qualquer, precisamos pensar nos valores adequados para cada um dos campos de nossa estrutura

Para inicializar uma estrutura qualquer, precisamos pensar nos valores adequados para cada um dos campos de nossa estrutura Para inicializar uma lista sequencial já criada pelo usuário, só precisamos colocar o valor 0 (zero) no número de elementos válidos

INICIALIZAÇÃO

```
void CriaLista(int capacidade) {
    this->capacidade = capacidade;
    this->tamanho = 0;
    this->vetor = new int[capacidade];
}
```

RETORNAR NÚMERO DE ELEMENTOS

Para esta estrutura basta **retornar** o valor do campo **tamanho**

```
int tamanhoLista(){
    return tamanho;
}
```

EXIBIÇÃO/IMPRESSÃO

Para exibir os elementos da estrutura precisaremos iterar pelos **elementos válidos** e, por exemplo, **imprimir suas chaves**.

```
void imprimir() {
    for (int i = 0; i < tamanho; i++) {
        cout << vetor[i] << " ";
    }
    cout << endl;
}
```

BUSCAR POR ELEMENTO

A função de busca deverá:

- **Receber** uma chave do usuário
- **Retornar** a posição em que este elemento se encontra na lista (caso seja encontrado)
- **Retornar** -1 caso não haja um registro com essa chave na lista

BUSCAR POR ELEMENTO

```
int buscarElemento(int chave){
  for(int i = 0; i < tamanho; i++){
    if(vetor[i] == chave){
      return i;
    }
  }
  return -1;
}
```

INSERÇÃO DE UM ELEMENTO

O usuário passa como parâmetro um registro a ser inserido na lista

Há diferentes possibilidades de inserção:

- No início
- No fim
- Ordenada pela chave
- Numa posição indicada pelo usuário

INSERÇÃO DE UM ELEMENTO

O usuário passa como parâmetro um registro a ser inserido na lista

Há diferentes possibilidades de inserção:

- **No início**
- No fim
- Ordenada pela chave
- Numa posição indicada pelo usuário

INSERÇÃO DE UM ELEMENTO: INICIO

Como inserir?

Se a lista **não estiver cheia**: desloca todos os elementos posteriores uma posição para a direita; insere o elemento na na primeira posição, soma um no campo tamanho e retorna **true** Caso contrário retorna **false**

INSERÇÃO DE UM ELEMENTO: INICIO

```
bool inserirInicio(int elemento) {
    if (tamanho < capacidade) {
      tamanho++;
      for(int i = tamanho; i >= 0; i--){
        vetor[i] = vetor[i - 1];
      }
      vetor[0] = elemento;
      return true;
    }
    else {
      cout << "Lista cheia" << endl;
      return false;
    }
}
```

INSERÇÃO DE UM ELEMENTO

O usuário passa como parâmetro um registro a ser inserido na lista

Há diferentes possibilidades de inserção:

- No início
- **No fim**
- Ordenada pela chave
- Numa posição indicada pelo usuário

INSERÇÃO DE UM ELEMENTO : FIM

Como inserir?

Se a lista **não estiver cheia**; insere o elemento na ultima posição e soma um no campo tamanho e retorna **true** Caso contrário retorna **false**

INSERÇÃO DE UM ELEMENTO : FIM

```
void inserirFim(int elemento) {
    if (tamanho < capacidade) {
        vetor[tamanho] = elemento;
        tamanho++;
    }
    else {
        cout << "Lista cheia" << endl;
    }
}
```

INSERÇÃO DE UM ELEMENTO

O usuário passa como parâmetro um registro a ser inserido na lista

Há diferentes possibilidades de inserção:

- No início
- No fim
- Ordenada pela chave
- **Numa posição indicada pelo usuário**

INSERÇÃO DE UM ELEMENTO : QUALQUE POSIÇÃO

Como inserir?

Se a lista não estiver cheia e o índice passado pelo usuário for válido: **desloca todos os elementos posteriores** uma posição para a direita; **insere o elemento na posição desejada**, soma um no campo nroElem e retorna **true** Caso contrário retorna **false**

INSERÇÃO DE UM ELEMENTO : QUALQUE POSIÇÃO

```
bool inserir(int elemento, int posicao) {
    if (tamanho < capacidade) {
      tamanho++;
      for(int i = tamanho; i > posicao; i--){

        vetor[i] = vetor[i - 1];
      }
      vetor[posicao] = elemento;
      return true;
    }
    else {
      cout << "Lista cheia" << endl;
      return false;
    }
}
```

DELETAR UM ELEMENTO

O usuário **passa a chave** do elemento que ele quer excluir **Se houver** um elemento com esta chave na lista, “**exclui** este elemento”, **desloca todos os elementos posteriores** uma posição para a esquerda, diminui em um o campo tamanho e retorna **true** Caso contrário, retorna **false**

DELETAR UM ELEMENTO

```
bool deletarElemento(int chave){
  int posicao = buscarElemento(chave);
  if(posicao != -1){
    for(int i = posicao; i < tamanho; i++){
      vetor[i] = vetor[i + 1];
    }
    tamanho--;
    return true;
  }
  return false;
}
```

ATIVIDADE 01

Crie as funções para deletar o primeiro elemento da lista, para deletar o ultimo elemento da lista e para deletar a lista inteira.

QUESTÃO 01 - BANCA: FCC

Em relação a tipos abstratos de dados, é correto afirmar que

A) o TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações disponibilizadas sobre esses dados.

B) algumas pilhas admitem serem declaradas como tipos abstratos de dados.

C) filas não permitem declaração como tipos abstratos de dados.

D) os tipos abstratos de dados podem ser formados pela união de tipos de dados primitivos, mas não por outros tipos abstratos de dados.

E) são tipos de dados que escondem a sua implementação de quem o manipula; de maneira geral as operações sobre estes dados são executadas sem que se saiba como isso é feito.

Ano: 2009 Banca: FCC Órgão: TRE-PI Prova: FCC - 2009 - TRE-PI - Técnico Judiciário - Programação de Sistemas

DÚVIDAS???