

Estrutura de Dados Lineares

Introdução e Conceitos Básicos

Prof. Luiz Gustavo Almeida Martins

Estruturas Lineares

- Usadas para **agrupar dados sequencialmente**
 - Representa uma **ordem (linear)** entre os dados
- **Definição geral:** Dada uma estrutura linear **E** , tal que:

$$E : [a_1, a_2, \dots, a_N], N \geq 0$$

- **a_1** é o 1^a elemento
- **a_N** é o último elemento
- $\forall i, 1 < i < N$, o elemento **a_i** é precedido por **a_{i-1}** e sucedido por **a_{i+1}**
- Se **$N = 0$** , então a estrutura é vazia

Exemplos de Operações Suportadas

- **Criar uma estrutura** vazia
- **Inserir um elemento** na estrutura
 - Qualquer posição: *insere*
 - Posição específica: *insere_inicio* e *insere_fim*
 - Posição adequada: *insere_ord*
- **Remover um elemento** da estrutura
 - Remove elemento específico (ex: 1ª ocorrência)
 - Remove todas ocorrências de um elemento específico
 - Remove em uma posição específica: *remove_inicio* e *remove_fim*

Exemplos de Operações Suportadas

- **Acessar o valor** do i -ésimo elemento da estrutura
- **Alterar o valor** do i -ésimo elemento da estrutura
- **Excluir** uma estrutura
- **Copiar** uma estrutura
- **Determinar o tamanho** da estrutura
- **Juntar duas estruturas** em uma terceira: **concatenar** e **intercalar**
- **Etc.**

Critérios Conceituais

- Existência de uma **disciplina de acesso**:
 - **Sem disciplina:**
 - Listas lineares
 - **Com disciplina:**
 - Filas (disciplina FIFO: *First In First Out*)
 - Pilhas (disciplina LIFO: *Last In First Out*)

Critérios Conceituais

- Existência de um **critério de ordenação**:
 - Estruturas **não ordenadas**:
 - Ordem definida implicitamente pela seqüência de entrada
 - **Ex:** lista não ordenada = $\{-5, 17, 3, 8\}$
 - Estruturas **ordenadas**:
 - Adota um critério de ordenação explícito (ascendente ou descendente)
 - **Ex:** Lista ordenada = $\{3, 5, 7, 8\}$ ou $\{8, 7, 5, 3\}$

Critérios Conceituais

- Possíveis estruturas:
 - Lista ordenada
 - Lista não ordenada
 - Pilha
 - Fila
 - Fila de prioridades
 - Deque

Critérios Conceituais

- Possíveis estruturas:

- Lista ordenada
- Lista não ordenada
- Pilha
- Fila
- Fila de prioridades
- Deque



**estruturas lineares
vistas no curso**

Critérios de Implementação

- **Forma de alocação** da memória:
 - Como armazenar os elementos na memória?

Critérios de Implementação

- **Forma de alocação da memória:**
 - Como armazenar os elementos na memória?
- **Alocação estática:**
 - Espaço **determinado na compilação**
 - Uso de vetores
 - **Vantagem:** implementação mais simples
 - **Desvantagem:** necessidade de definir antecipadamente o n^0 máximo de elementos
 - Pode haver sub ou superestimação

Critérios de Implementação

- **Forma de alocação** da memória:
 - Como armazenar os elementos na memória?
- **Alocação dinâmica:**
 - Espaço **alocado em tempo de execução**
 - Uso de *malloc()* e ponteiros
 - **Vantagem:** uso otimizado da memória
 - **Desvantagem:** necessidade de programar o controle de acesso à memória
 - Código mais complexo

Critérios de Implementação

- **Forma de acesso a estrutura:**
 - Como referenciar um elemento da estrutura?

Critérios de Implementação

- **Forma de acesso a estrutura:**
 - Como referenciar um elemento da estrutura?
- **Acesso sequencial:**
 - Elementos usam **posições consecutivas** na memória
 - Permite o uso da aritmética de ponteiros
 - **Vantagem:** Acesso direto ao i -ésimo elemento
 - **Desvantagem:** inserção e remoção no meio da lista, exige movimentação de elementos

Critérios de Implementação

- **Forma de acesso a estrutura:**
 - Como referenciar um elemento da estrutura?
- **Acesso encadeado:**
 - Elementos podem **ocupar qualquer área da memória** (não necessariamente consecutiva)
 - **Vantagem:** inserção e remoção no meio da lista é simples (não exige movimentação)
 - **Desvantagens:** usa mais memória e não permite acesso direto aos elementos
 - Elemento precisa guardar pelo menos seu sucessor
 - Necessidade de percorrimento da lista

Critérios de Implementação

- Possíveis esquemas:
 - Estática / Sequencial
 - Dinâmica / Encadeada
 - Estática / Encadeada
 - Dinâmica / Sequencial

Critérios de Implementação

- Possíveis esquemas:
 - Estática / Sequencial
 - Dinâmica / Encadeada
 - Estática / Encadeada
 - Dinâmica / Sequencial
- implementações
vistas no curso**

Exercícios

1. *Especificar o TAD lista não ordenada de inteiros com as seguintes operações:*
 - ***Cria_lista:*** *cria um lista vazia*
 - ***Lista_vazia:*** *verifica se a lista está vazia*
 - ***Lista_cheia:*** *verifica se a lista está cheia*
 - ***insere_elem:*** *insere um elemento na lista*
 - ***remove_elem:*** *retira uma ocorrência de um dado elemento da lista*
2. *Especificar o TAD lista ordenada de inteiros com as mesmas operações do exercício anterior*

Especificação TAD

Lista Não Ordenada

- **Cabeçalho:**

TAD **lista não ordenada**

- **Dados:** números inteiros
- **Lista de operações:** *cria_lista*, *lista_vazia*, *lista_cheia*, *insere_elem* e *remove_elem*

TAD Lista Não Ordenada

- Operação ***cria_lista***:
 - **Entrada**: nenhuma
 - **Pré-condição**: nenhuma
 - **Processo**: criar uma lista e deixá-la no estado de vazia
 - **Saída**: endereço da lista criada
 - **Pós-condição**: nenhuma

TAD Lista Não Ordenada

- Operação ***lista_vazia***:
 - **Entrada**: endereço de uma lista
 - **Pré-condição**: lista ser válida
 - **Processo**: verifica se a lista está na condição de vazia
 - **Saída**: 1 se vazia ou 0 caso contrário
 - **Pós-condição**: nenhuma

TAD Lista Não Ordenada

- Operação ***lista_cheia***:
 - **Entrada**: endereço de uma lista
 - **Pré-condição**: lista ser válida
 - **Processo**: verifica se a lista está na condição de cheia
 - **Saída**: 1 se cheia ou 0 caso contrário
 - **Pós-condição**: nenhuma

TAD Lista Não Ordenada

- Operação ***insere_elem***:
 - **Entrada**: endereço de uma lista e o elemento (número inteiro) a ser inserido
 - **Pré-condição**: lista ser válida e não estar cheia
 - **Processo**: inserir o elemento no final da lista
 - **Saída**: 1 - sucesso ou 0 - falha
 - **Pós-condição**: a lista de entrada com um elemento a mais

TAD Lista Não Ordenada

- Operação ***remove_elem***:
 - **Entrada:** endereço de uma lista e o elemento (número inteiro) a ser removido
 - **Pré-condição:** lista ser válida e não estar vazia
 - **Processo:** percorrer a lista até encontrar o elemento desejado ou chegar ao seu final. Se o elemento existe, remova-o da lista.
 - **Saída:** 1 - sucesso ou 0 - falha
 - **Pós-condição:** a lista de entrada c/ 1 elemento a -

Especificação TAD

Lista Ordenada

- Ordenação afeta apenas as operações:
 - ***Inserir_ord***: deve inserir na posição correta de modo a manter o critério de ordenação
 - ***Remove_ord***: deve aproveitar da ordenação para otimizar a busca
 - Pára antes quando não existe o elemento

TAD Lista Ordenada

- Operação ***insere_ord***:
 - **Entrada:** endereço de uma lista e o elemento (número inteiro) a ser inserido
 - **Pré-condição:** lista ser válida e não estar cheia
 - **Processo:** percorrer a lista até encontrar a posição correta de inserção para garantir a ordenação (próximo for maior que o elemento). Inserir o elemento na posição escolhida.
 - **Saída:** 1 - sucesso ou 0 - falha
 - **Pós-condição:** a lista de entrada c/ um elemento a +

TAD Lista Ordenada

- Operação ***remove_ord***:
 - **Entrada:** endereço de uma lista e o elemento (número inteiro) a ser removido
 - **Pré-condição:** lista ser válida e não estar vazia
 - **Processo:** percorrer a lista até encontrar o elemento desejado **ou um elemento maior**. Se o elemento existe, remova-o da lista.
 - **Saída:** 1 - sucesso ou 0 - falha
 - **Pós-condição:** a lista de entrada c/ 1 elemento a -

Referências

- *Oliveira, Gina M. B. de, TAD, material didático da disciplina de Algoritmos e Estruturas de Dados 1, UFU.*
- *Backes, André, Linguagem C Descomplicada, portal de vídeo-aulas, <https://programacaodescomplicada.wordpress.com/>, acessado em 09/03/2016.*
- *Celes, W., Cerqueira, R. e Rangel, J. L. Introdução a estruturas de dados. Ed. Campus Elsevier, 2004.*