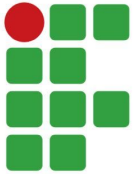


INSTITUTO FEDERAL
Bahia
Campus Santo Antônio de Jesus

http://www.portal.ifba.edu.br/santoantonio

ESTRUTURA DE DADOS

Prof. George Pacheco Pinto

Agenda

- Listas Lineares
  - Listas sequenciais

LISTAS LINEARES

- É um conjunto de dados dispostos e/ou acessíveis em uma sequência determinada.
  - Pode possuir uma ordem intrínseca (Lista Ordenada) ou não.
  - Pode ocupar espaços de memória fisicamente consecutivos, espelhando a sua ordem, ou não.
  - Se os dados estiverem dispersos fisicamente, para que este conjunto seja uma lista, ele deve possuir operações e informações adicionais que permitam que seja tratado como tal (Lista Encadeada).

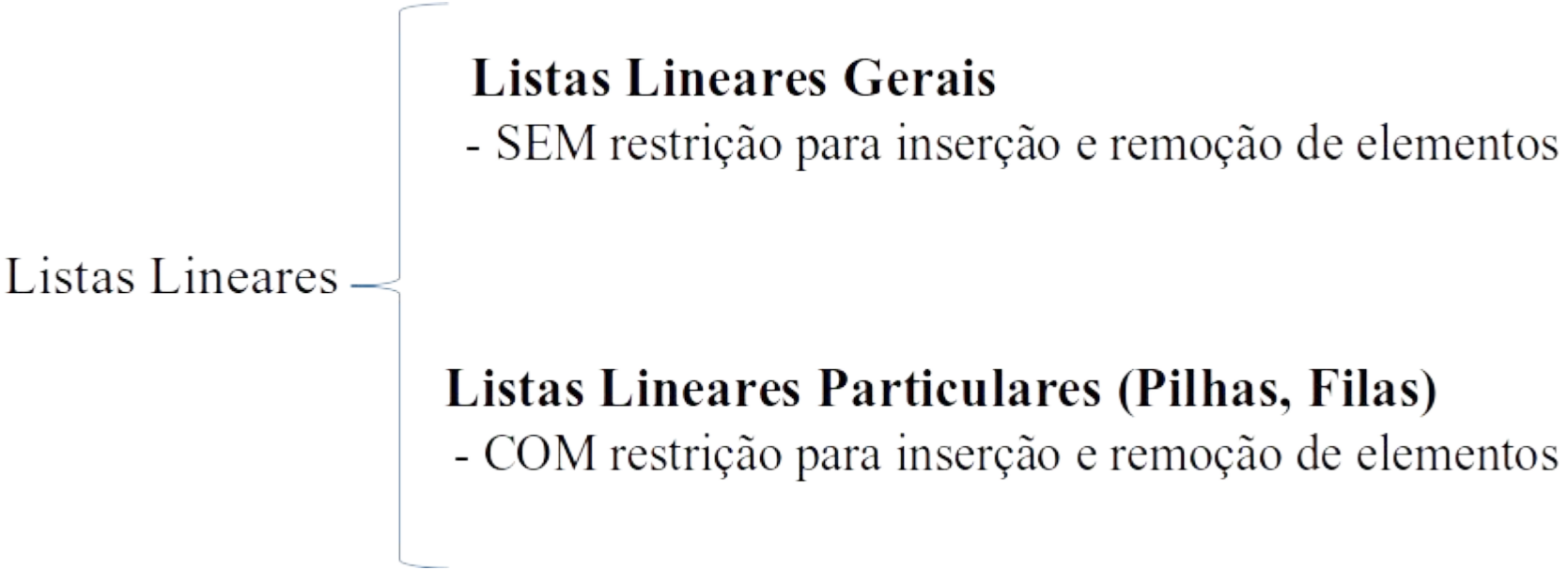
LISTAS LINEARES

- Definição: sequência de zero ou mais elementos a1,a2, ...,an sendo
  ai elementos de um mesmo tipo
  n o tamanho da lista linear

- Sua propriedade fundamental refere-se a posição relativa dos itens, então
  Se n=0 dizemos que a lista está vazia, senão
  a1 é o primeiro elemento da lista
  an é o último elemento da lista
  ai precede ai+1 (e ai sucede ai-1)
  ai é dito estar na i-ésima posição da lista

LISTAS LINEARES

- Classificação



LISTAS LINEARES

- Listas lineares gerais:
  - A inclusão e remoção de elementos pode ser realizada em qualquer posição da lista. Essas listas não apresentam nenhuma restrição de acesso

- Casos particulares de listas:
  - Pilha: Inserções e remoções são realizadas em apenas um extremo (topo).

  - Fila: inserções realizadas em um extremo (fim) e remoções em outro (início).

LISTAS LINEARES

- Para criar um TAD Lista, é necessário definir um conjunto de operações sobre os objetos do tipo Lista.

- O conjunto de operações a ser definido depende de cada aplicação, não existindo um conjunto de operações que seja adequado a todas as aplicações.

LISTAS LINEARES - OPERAÇÕES USUAIS

- Iniciar/Criar uma lista linear vazia
- Inserir um novo elemento após o i-ésimo elemento
- Retirar i-ésimo elemento
- Buscar o i-ésimo elemento
- Combinar duas ou mais listas em uma lista única
- Dividir uma lista em duas ou mais listas
- Fazer uma cópia da lista
- Ordenar os itens da lista em ordem crescente ou decrescente

IMPLEMENTAÇÃO DO TAD LISTA

- Tipos de Implementação:
  - Através de vetores - listas estáticas
  - Através de apontadores ou ponteiros - listas dinâmicas

- Em qualquer uma das implementações, deve-se:
  - Definir os Dados
  - Definir as Operações

TAD LISTA - OPERAÇÕES BÁSICAS

Inicia_Lista (L)
parâmetros: TipoLista L;
pós-condição: Lista L vazia;
funcionalidade: cria uma lista vazia
resultado: retorno de uma lista vazia criada

Lista_Vazia (L)
parâmetros: TipoLista L;
funcionalidade: Testa se a lista está vazia ou não
resultado: retorna *true* se a lista está vazia; senão *false*

TAD LISTA - OPERAÇÕES BÁSICAS

Retira_Lista (pos, L, x)
parâmetros: TipoLista L, TipoItem x, int pos;
pré-condição: Lista L tem n>1 elementos e pos < posição do último;
pós-condição: Lista L tem n - 1 elementos;
funcionalidade: Retorna o item x que está na posição p da lista mantendo a ordenação dos demais
resultado: x contém o item removido da lista

TAD LISTA - OPERAÇÕES BÁSICAS

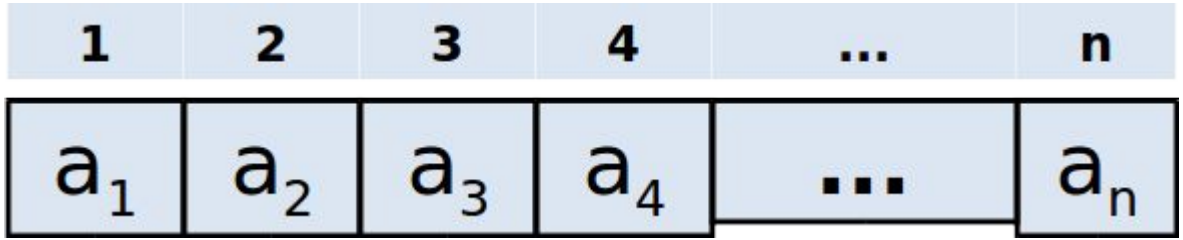
Insere_Lista (x, L)
parâmetros: TipoItem x, TipoLista L;
pré-condição: Lista L tem n>= 0 elementos e n <TamMax;
pós-condição: Lista L tem n+1 elementos;
funcionalidade: Insere o elemento x após o último elemento da lista
resultado: x foi inserido na lista

Imprime_Lista (L)
parâmetros: TipoLista L;
funcionalidade: Imprime os itens da lista
resultado: impressão dos itens da lista na ordem de ocorrência

LISTAS ESTÁTICAS SEQUENCIAIS

- Lista sequencial: conjunto de registros de mesmo tipo na qual o sucessor de um elemento ocupa a posição física subsequente a este elemento

- Implementação desta estrutura em C feita usando vetores e estruturas



Listas Estáticas Sequenciais - Vetores

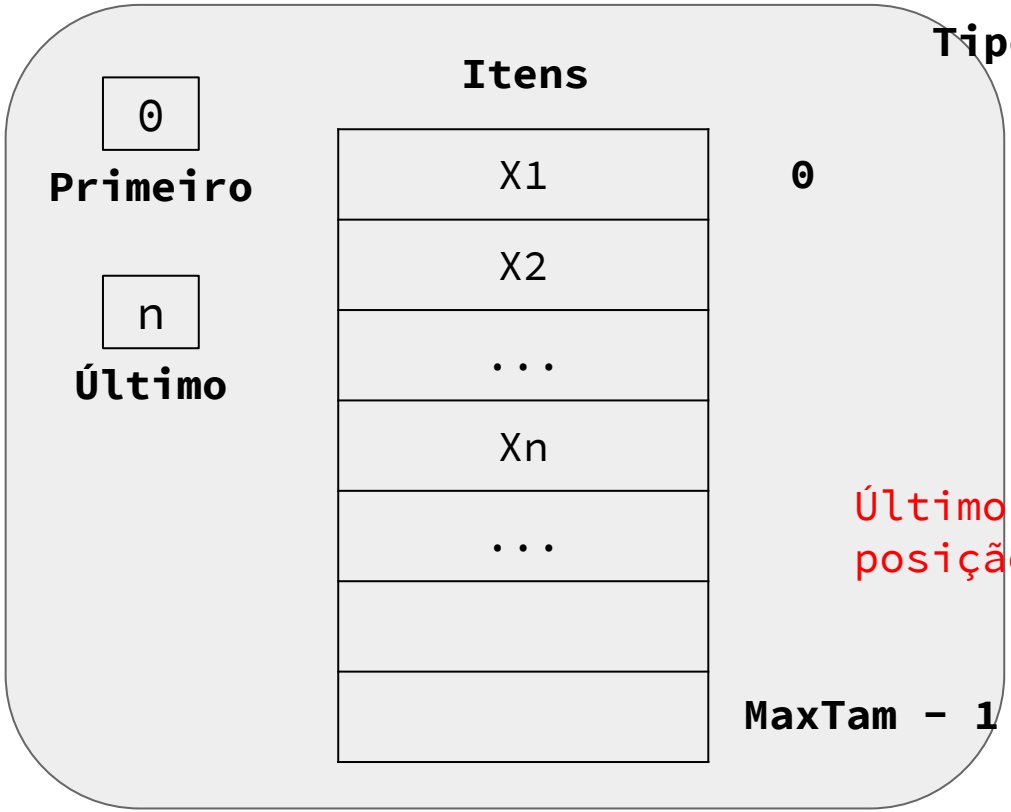
- Características
  - Os itens são armazenados em posições contíguas de memória;
  - A lista pode ser percorrida em qualquer direção;
  - A inserção de um novo item pode ser realizada após o último item com custo constante.
  - A inserção de um novo item no meio da lista requer um deslocamento de todos os itens localizados após o ponto de inserção.
  - Retirar um item do início da lista requer um deslocamento de itens para preencher o espaço deixado vazio

Listas Estáticas Sequenciais - Vetores

- Vantagens
  - Acesso direto indexado a qualquer elemento (facilita modificações dos conteúdos);
  - Tempo constante para acessar os elementos (depende só do índice).

- Desvantagens
  - Movimentação para a inserção e remoção de elementos;
  - Conhecimento a priori do tamanho do número máximo da lista. Depois da alocação, não mais do que essa quantidade poderá ser solicitada;
  - Quantidade fixa de memória permanece alocada, mesmo quando a estrutura estiver vazia.

LISTAS ESTÁTICAS SEQUENCIAIS - TAD



Último: corresponde à 1ª posição vaga para inserção

LISTAS ESTÁTICAS SEQUENCIAIS - TAD

```
#define MaxTam 100
#define InicioVetor 0

typedef int TipoChave;

typedef int Apontador;
```

```
typedef struct {
  TipoChave Chave;
  /* outros componentes */
} TipoItem;

typedef struct {
  TipoItem item[MaxTam];
  Apontador Primeiro, Ultimo;
} TipoLista;
```

Implementação Operações Básicas

```
void Inicia_Lista(TipoLista *Lista)
{
   Lista->Primeiro = InicioVetor;
   Lista->Ultimo = Lista->Primeiro;
}

int Lista_Vazia(TipoLista Lista)
{
   return Lista.Primeiro == Lista.Ultimo;
}
```

Implementação Operações Básicas

```
int Insere_Lista(TipoItem x, TipoLista *Lista) {
   if (Lista->Ultimo > MaxTam - 1) {
      printf("Lista esta cheia\n");
      return (0);
   } else  {
      Lista->itens[Lista->Ultimo] = x;
      Lista->Ultimo++;
      return (1);
   }
}
```

Implementação Operações Básicas

```
void Retira_Lista(Apontador p, TipoLista *Lista, TipoItem *Item)
{
    int Aux;
    if (Lista_Vazia(*Lista) || p-1 >= Lista->Ultimo)
    {
        printf(" Erro Posicao nao existe\n");
        return;
    }
    *Item = Lista->itens[p - 1];
    for (Aux = p; Aux < Lista->Ultimo; Aux++)
        Lista->itens[Aux - 1] = Lista->itens[Aux];
    Lista->Ultimo--;
}
```

Implementação Operações Básicas

```
void ImprimeLista(TipoLista Lista)
{
    int Aux;

    for (Aux = Lista.Primeiro; Aux<=(Lista.Ultimo - 1); Aux++)
       printf("%d\n", Lista.itens[Aux].chave);
}
```

EXERCÍCIOS

1. Escreva uma função para exibir todos os alunos cadastrados numa lista.

2. Escreva uma função que recebe como parâmetros uma lista de alunos L e um valor z e que modifique o conteúdo de L, retirando todos os alunos com nota maior que z. Após a execução da função, o conteúdo de L deverá estar alterado.

Referências

- Consultar ementário.