SCC-223 Estruturas de Dados I

Lista Linear Encadeada
(Estática)

Profa. Elaine Parros Machado de Sousa

ICMC USP SÃO CARLOS

Aula baseada em material dos professores Adenilso Simão e Rudinei Goularte.

Grupo de Bases de Dados e Imagens
USP - São Carlos

Relembrando...

Lista Linear $\Rightarrow$ estrutura que armazena uma sequência de elementos (itens) do mesmo tipo

- **posição relativa dos elementos**
  - $L = (e_1, e_2, \ldots, e_n)$

1) **Ordenação de elementos**
   - Lista ordenada
   - Lista não ordenada

Relembrando...

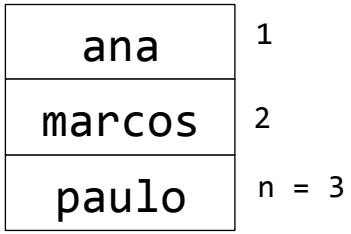
Lista Linear $\Rightarrow$ $L=(e_1,e_2,...,e_n)$

2) Organização em memória

- Lista sequencial

L = (ana,marcos,paulo)

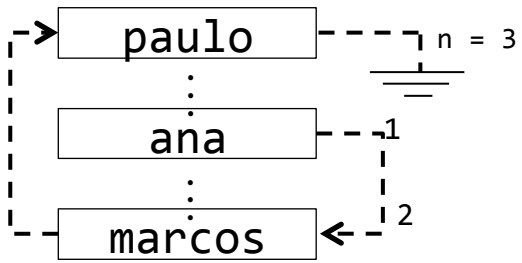
Memória Principal



- Lista encadeada

L = (ana,marcos,paulo)

Memória Principal



Relembrando...

Lista Linear $\Rightarrow$ $L=(e_1, e_2, \ldots, e_n)$

3) Alocação de Memória

- Alocação Estática
- Alocação Dinâmica

Sequencial e estática => *array*

Sequencial e dinâmica => alocação dinâmica de *array*

Encadeada e estática => *array* simulando encadeamento

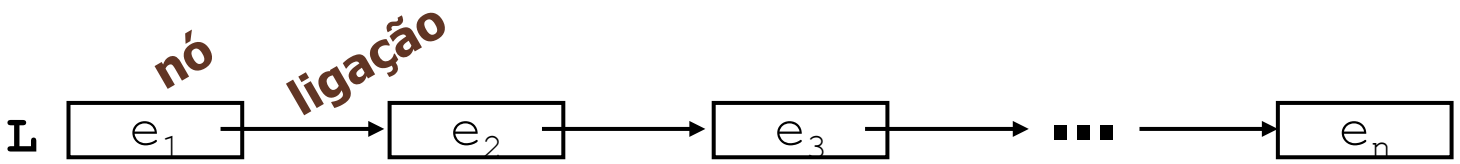
Encadeada e Dinâmica=> ponteiros

Lista Encadeada

Lista Encadeada

- Lista definida como uma sequência de **NÓS ENCADEADOS**
  - sequência lógica
- Cada nó contém:
  - e_i: elemento da lista
  - **ligação**: endereço do nó que armazena o elemento sucessor

Lista Encadeada – ORGANIZAÇÃO LÓGICA

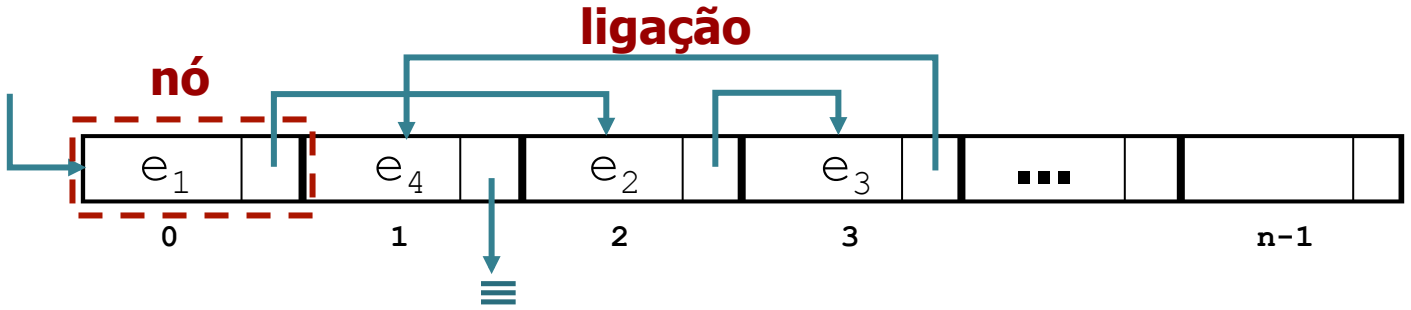


Lista Encadeada

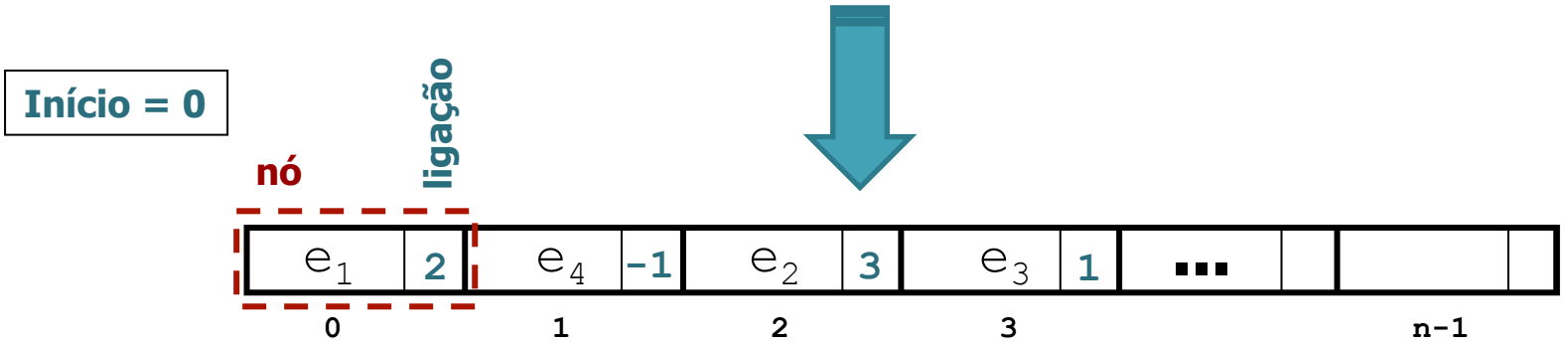
- Lista Encadeada Estática 
  - *arrays*
- Lista Encadeada Dinâmica
  - ponteiros

Lista Encadeada Estática

- Usando um *Array*....

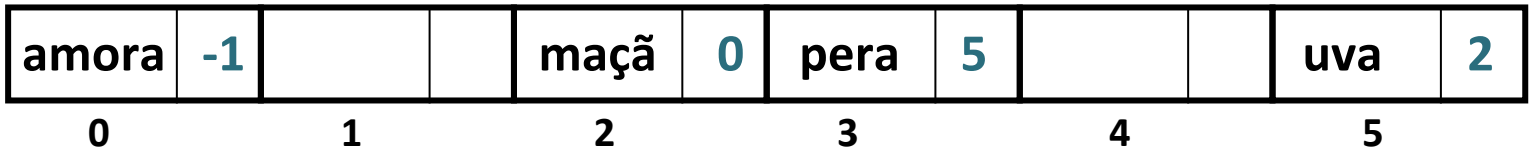


  - **Índice** (endereço) de um elemento do *array* **não indica sua posição** na lista
  - **Índice** é utilizado para **relacionar** cada **elemento** a seu **sucessor**



Lista Encadeada Estática

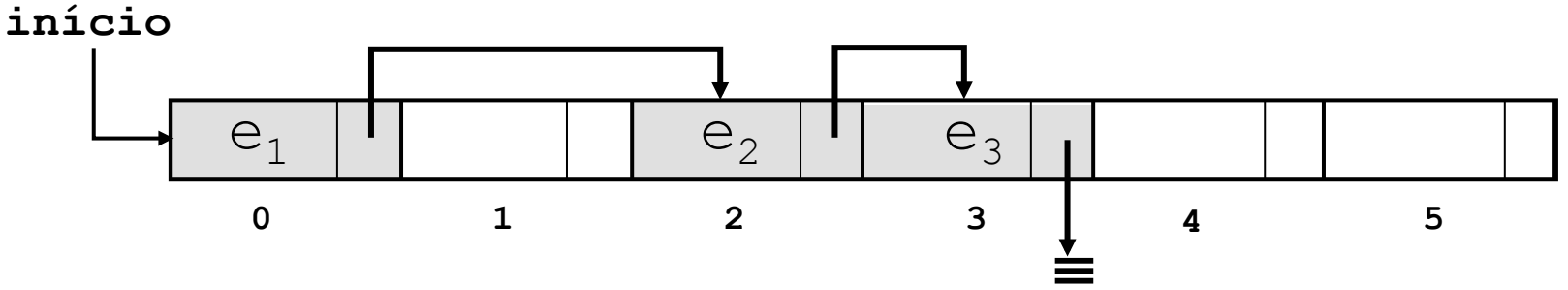
- Exemplo:
  - **L = (pera, uva, maçã, amora)**



  - início = **3**
  - Obs:
    - variável especial (**início**) indica o índice do 1º elemento
    - valor especial no último elemento (**-1**)

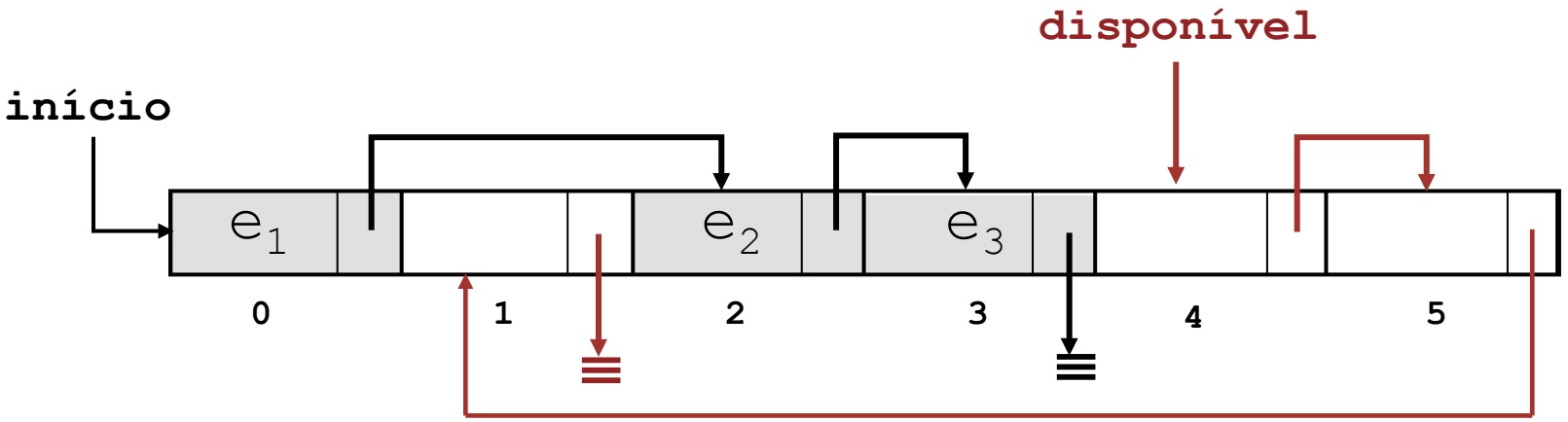
Lista Encadeada Estática

- No mesmo *array* :
  - nós com **elementos da lista** (ocupados)
  - nós **disponíveis** (vazios)
- Necessário diferenciá-los
  - **disponíveis** são usados para futuras **inserções**
  - nas **eliminações**, nós ocupados tornam-se **disponíveis**.



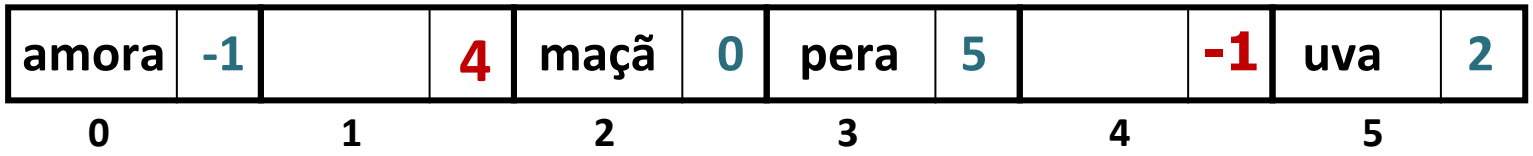
Lista Encadeada Estática

- Solução => juntar os nós **disponíveis** numa outra **lista encadeada** no mesmo vetor!
- No mesmo array:
  - Lista encadeada de elementos (Lista L)
  - Lista encadeada de nós disponíveis



Lista Encadeada Estática

- Exemplo:
  - **L = (pera, uva, maçã, amora)**
    - Lista L = 3→5 → 2 → 0
    - disponível = 1 → 4



  - início = **3**
  - disponível = **1**
    - variável especial que indica o índice do 1º nó vazio

Lista Encadeada Estática

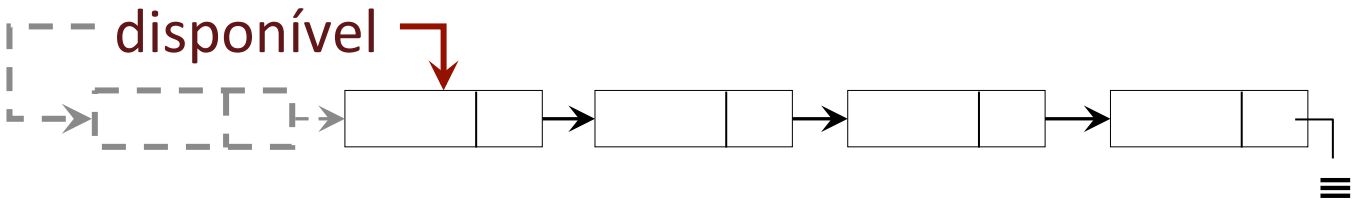
Lista de nós disponíveis

- Observações:
  - **inserção** na Lista L ⇒ **eliminação** na lista disponível
  - **eliminação** na Lista L ⇒ **inserção** na lista disponível
  - lista disponível **cheia** ⇒ Lista L **VAZIA**
  - lista disponível ⇒ todos os nós estão vazios
    - qualquer nó pode ser eliminado da lista disponível para ser inserido na Lista L
      - maneira eficiente?

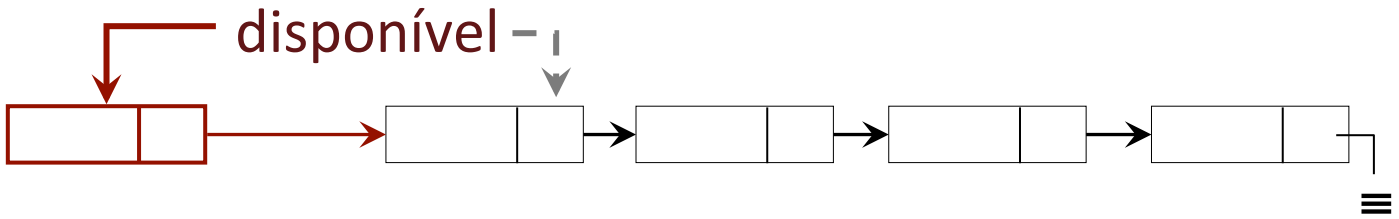
Lista Encadeada Estática

Lista de nós disponíveis

- Inserção na Lista L => **eliminar o 1º nó da disponível**



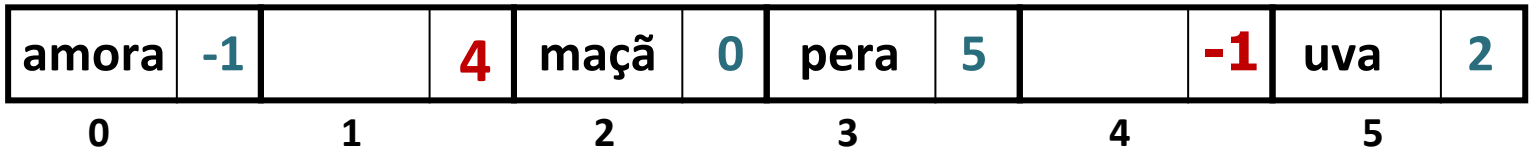
- Remoção na Lista L => **inserir como 1º nó da disponível**



- Inserção e eliminação ocorrem numa única extremidade (cabeça) da lista disponível ⇒ comportamento de uma **PILHA**.

Lista Encadeada Estática

- Exemplo:
  - **L = (pera, uva, maçã, amora)**
    - Lista L = 3→5 → 2 → 0
    - disponível = 1 → 4



  - início = **3**
  - disponível = **1**
    - variável especial que indica o índice do 1º nó vazio

TAD Lista Não Ordenada – Exemplo de Definição da Interface

Arquivo Lista.h /* o mesmo definido para implementação de lista sequencial com inclusão do define NULO */

```
1  #ifndef LISTA_H
2  #define LISTA_H
3
4  #define TAM_MAX 100 /*estimativa do tamanho máximo da lista*/
5  #define boolean int /*define tipo booleano – não existe em C*/
6  #define FALSE 0
7  #define TRUE 1
8  #define inicial 0
9  #define ERRO -32000
10 #define NULO -1 /* define para NULO (ligação) */
11
12 typedef int ITEM; /*Tipo ITEM (da lista) é um inteiro*/
13
14 typedef struct lista_ LISTA;
15
...
```

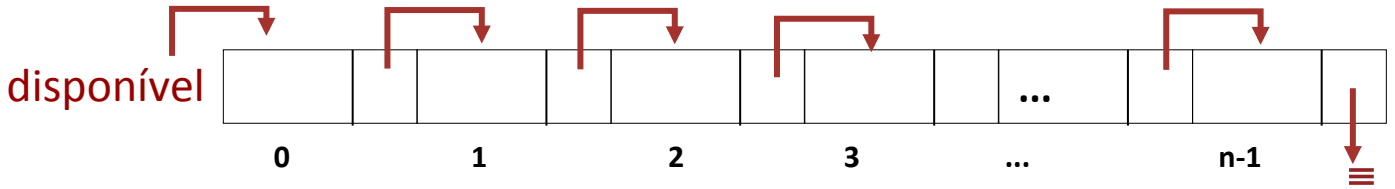
Arquivo Lista.h

```
...
16
17 LISTA *lista_criar(void);
18 boolean lista_apagar(LISTA *lista);
19 int lista_inserir(LISTA *lista, ITEM item);
20 boolean lista_inserir_pos(LISTA *lista, int pos, ITEM item);
21 boolean lista_remover(LISTA *lista, int chave);
22 boolean lista_remover_pos(LISTA *lista, int pos);
23 int lista_busca(int chave, LISTA *lista);
24 int lista_tamanho(LISTA *lista);
25 boolean lista_vazia(LISTA *lista);
26 boolean lista_cheia(LISTA *lista);
27 void lista_imprimir(LISTA *lista);
28
29 #endif
```

TAD Lista Não Ordenada – Exemplo de Implementação Encadeada com *Array*

Arquivo Lista.c

```
#include "lista.h”

struct no_ { /*Definicao da estrutura do no */
     ITEM item;
      int proximo;
};


typedef struct no_ NO; /*tipo NO */


struct lista_ {
  NO lista[TAM_MAX];
  int inicio;       /* inicio da lista de elementos */
  int disponivel;  /* início da lista de disponíveis */
};
```

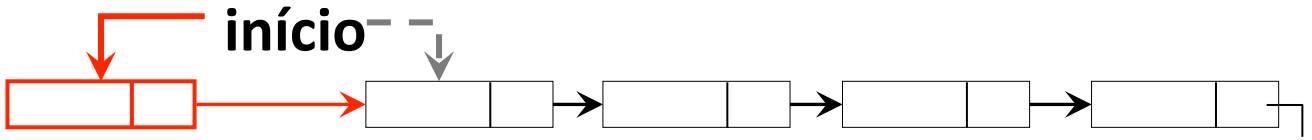


Arquivo Lista.c

```
#include "lista.h”
....
 /*Cria logicamente uma lista, inicialmente vazia*/
 LISTA *lista_criar(void){
     LISTA *lista = (LISTA *) malloc(sizeof(LISTA));
     int i;
     if (lista != NULL){
        lista->inicio = NULO; /*lista vazia*/
        lista->disponivel = inicial; /*lista disp. cheia*/
        for (i = 0; i < (TAM_MAX - 1); i++) /*encadeamento*/
             lista->lista[i].proximo = i + 1;
        lista->lista[TAM_MAX - 1].proximo = NULO;
      }
   return (lista);
 }
```

Arquivo Lista.c

```
...
/* verifica se lista está vazia */
boolean lista_vazia(LISTA *lista) {
	return (lista->inicio == NULO);
};

/* verifica se lista está cheia*/
boolean lista_cheia(LISTA *lista) {
	return (lista->disponivel == NULO);
};
```

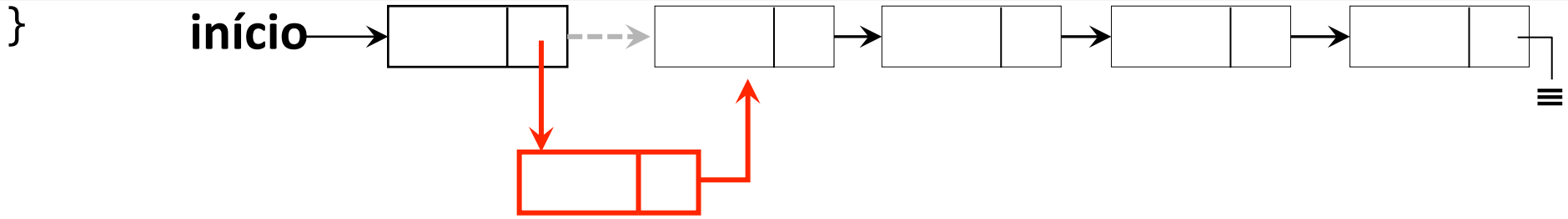


Arquivo Lista.c

```
/*Insere um elemento no início da lista.*/
 boolean lista_inserir(LISTA *lista, ITEM item){
  int dispo;
  if ((lista != NULL) && !lista_cheia(lista)){
        dispo = lista->disponivel;
        lista->disponivel = lista->lista[dispo].proximo;
        lista->lista[dispo].item = item;
        lista->lista[dispo].proximo = lista->inicio;
        lista->inicio = dispo;
        return (TRUE);
  }
  return(FALSE);
}
```

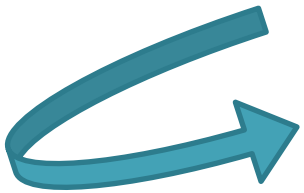
Arquivo Lista.c

```
/*Insere o k-ésimo elemento na lista.*/
boolean lista_inserir_pos(LISTA *lista, int pos, ITEM item) {
  int atual, dispo;
  if ((lista != NULL) && !lista_cheia(lista))
       ...
     while (....) /*busca posição para inserção */

     ... /*remoção na lista de disponíveis*/

     lista->lista[dispo].item = item;
     lista->lista[dispo].proximo = lista->lista[atual].proximo;
     lista->lista[atual].proximo = dispo;
     ....
}
```



Lista Encadeada Estática

- Comparação com lista sequencial
  - Ainda exige previsão de espaço ⇒ ponto fraco em comum (*)
  - **Vantagem:**
    - não há movimentos durante inserção e remoção de elementos da lista ⇒ apenas ligações são alteradas.
  - **Desvantagens:**
    - acesso ao i-ésimo elemento deixa de ser direto ⇒ requer acesso aos `i-1` elementos anteriores;
    - requer gerenciamento da lista de nós disponíveis (*)
    - não permite busca binária em listas ordenadas
    - alternativa para reduzir desvantagens (*)?



Lista Encadeada Dinâmica próxima aula...

Exercícios (não é preciso entregar)

- Implemente e teste as demais operações para o TAD Lista Não Ordenada (implementação encadeada em *array*)

- Implemente e teste o TAD Lista Ordenada (implementação encadeada em *array*)