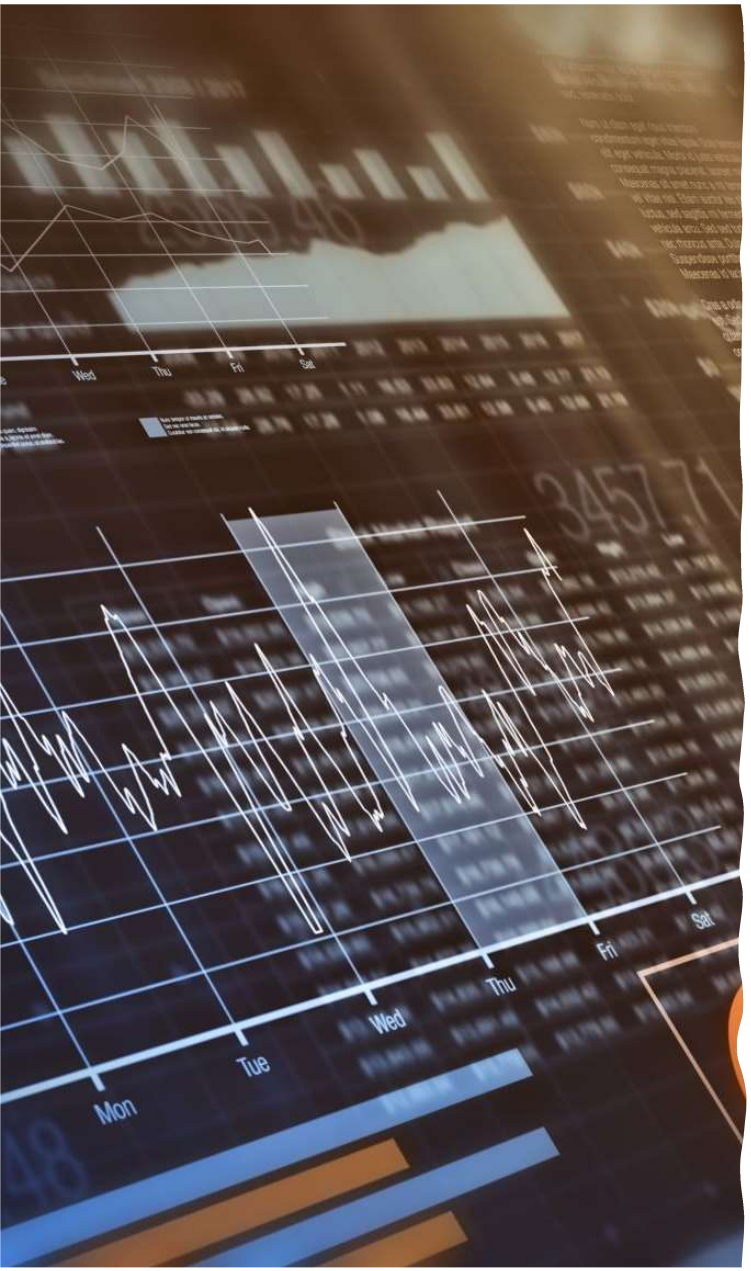
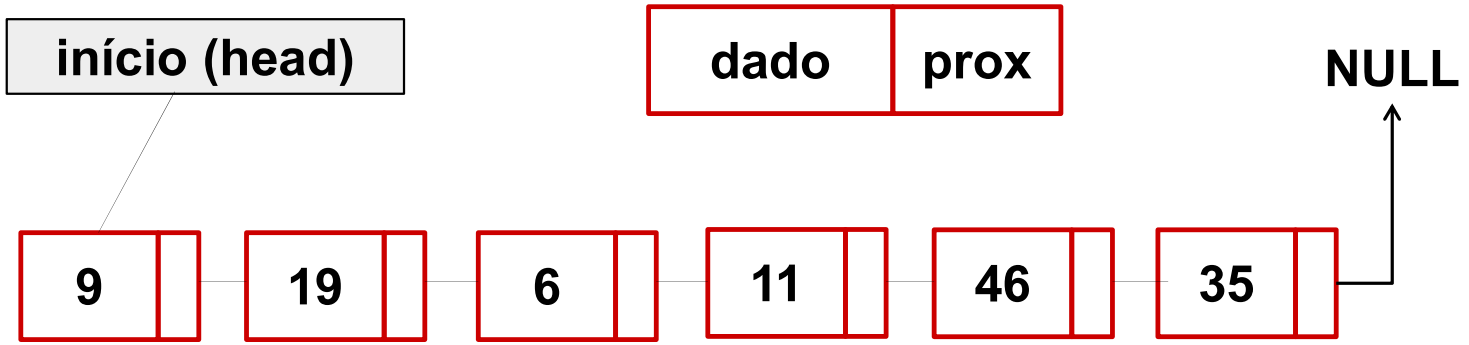
LISTAS

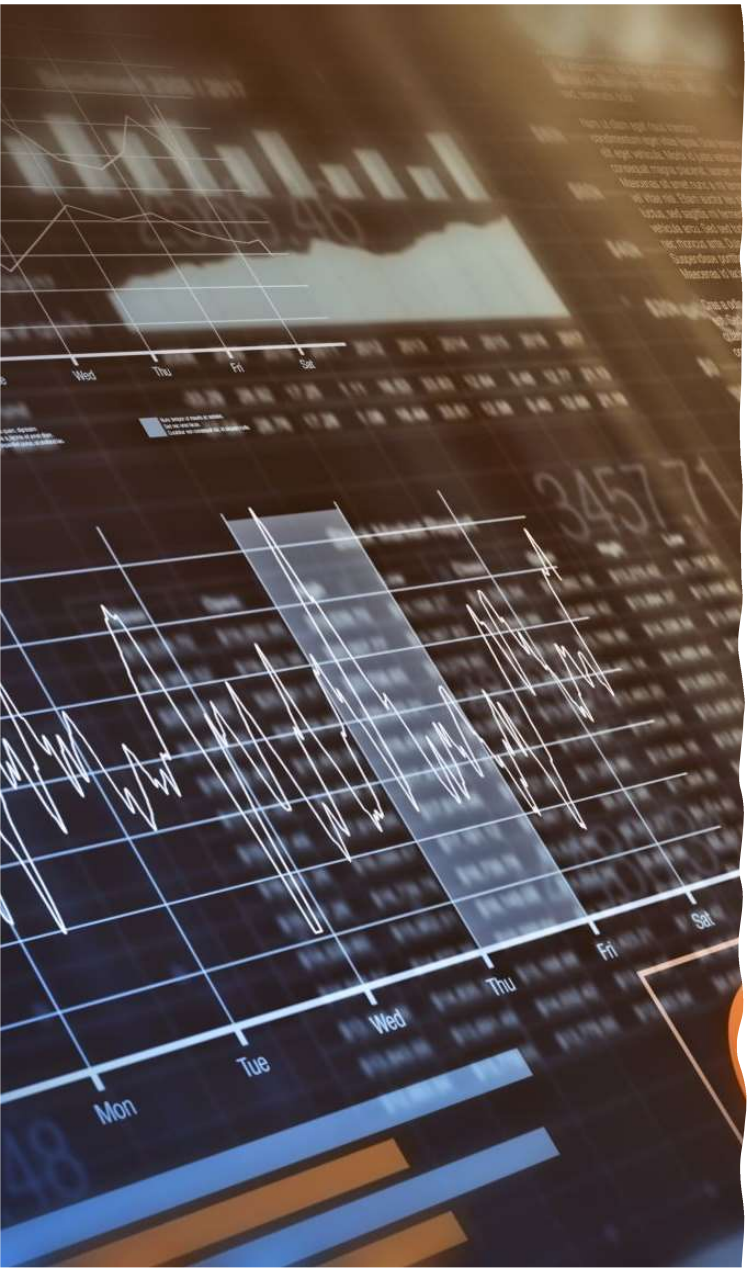
Lista Dinâmica Encadeada



LISTA DINÂMICA ENCADEADA

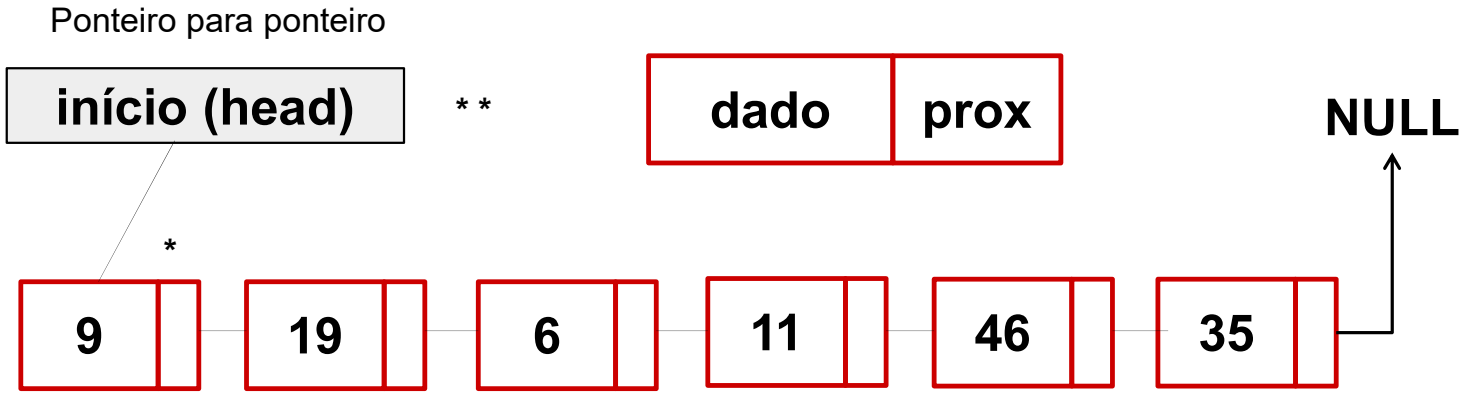
- Tipo de lista onde cada elemento aponta para o seu sucessor na “lista”.
- Usa um ponteiro especial para o primeiro elemento da lista e uma indicação de final de lista.





LISTA DINÂMICA ENCADEADA

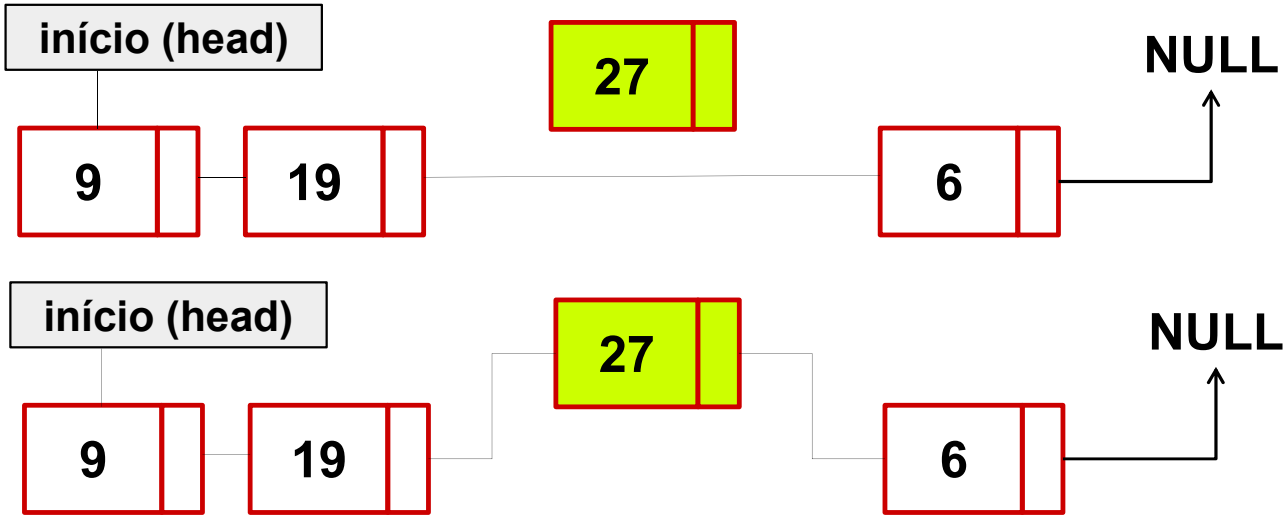
- Cada elemento é tratado como um ponteiro que é alocado dinamicamente à medida que os dados são inseridos.
- Para guardar o primeiro elemento, utilizamos um “ponteiro para ponteiro (guarda o endereço de um ponteiro)”.
- Desta forma é possível alterar facilmente quem está no início da lista alterando o “conteúdo” do “ponteiro para ponteiro”.



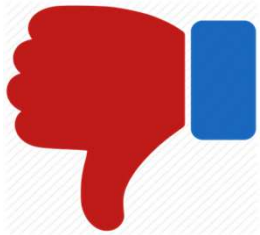
LISTA DINÂMICA ENCADEADA

Vantagens:

- Melhor utilização dos recursos de memória.
- Não precisa deslocar os elementos nas operações de inserção e remoção.

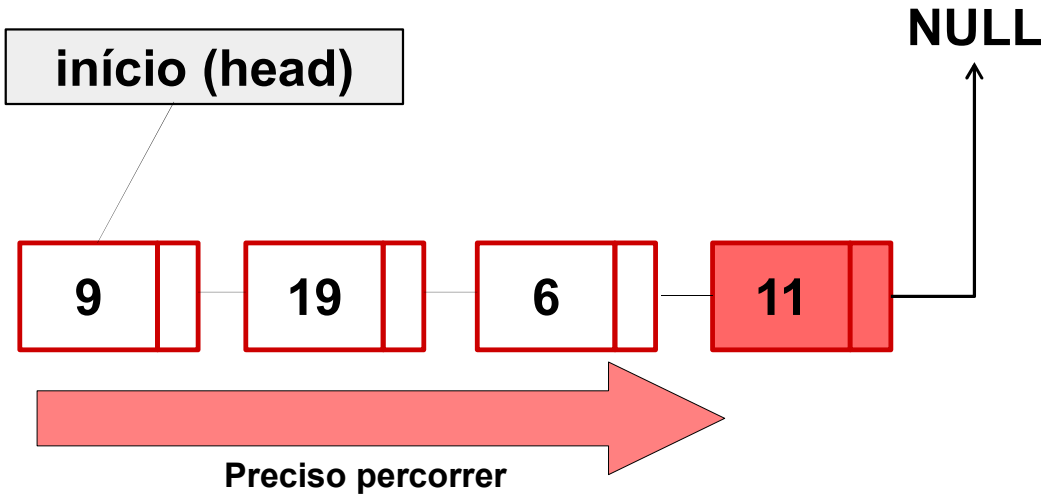


LISTA DINÂMICA ENCADEADA



Desvantagens:

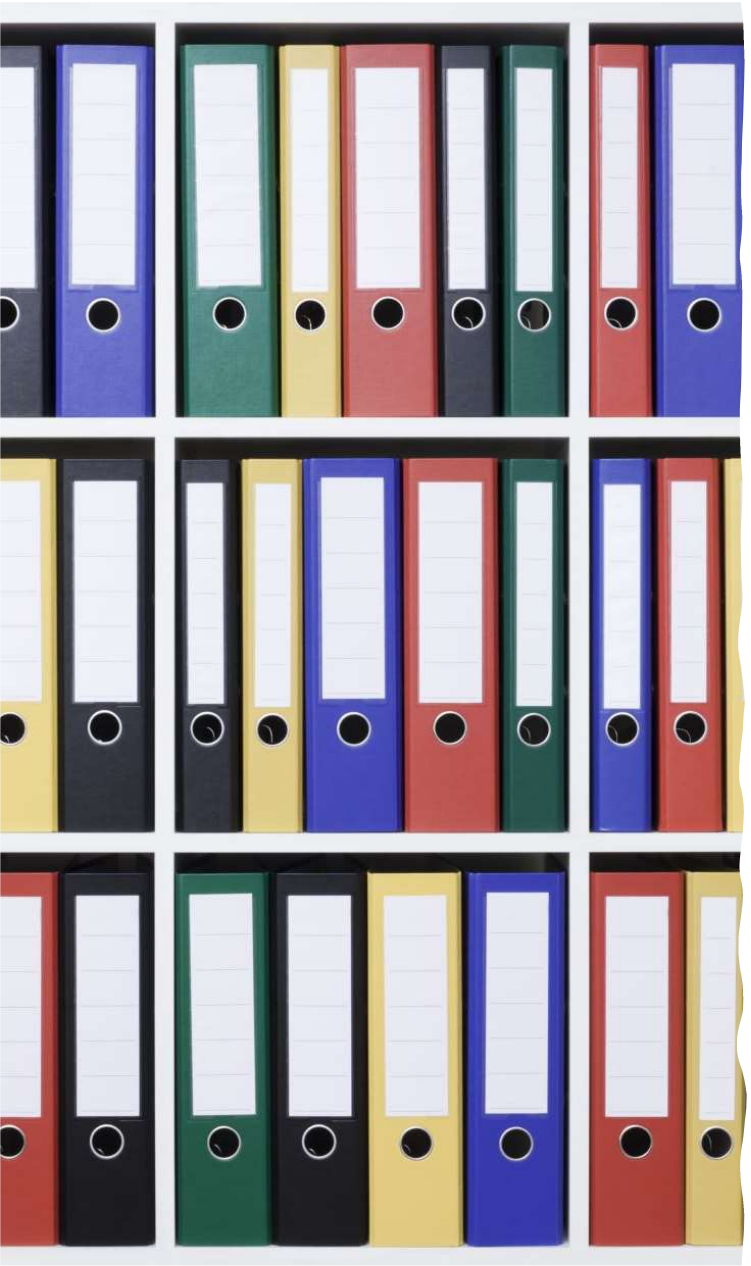
- Acesso indireto aos elementos
- Necessidade de percorrer a lista para acessar um elemento



LISTA DINÂMICA ENCADEADA

Quando utilizar?

- Quando não posso garantir um espaço mínimo para a execução da aplicação.
- Quando temos inserção e remoção em lista ordenada como operações mais frequentes.



LISTA DINÂMICA ENCADEADA

//Arquivo ListaDinEncadeada.h

- Protótipos das funções
- O tipo de dado armazenado na lista
- O ponteiro “lista”
- ~~Tamanho do vetor usado na lista~~

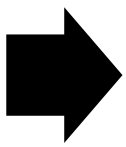
//Arquivo ListaDinEncadeada.c

- O tipo de dado “lista”
- Implementa as suas funções

//main.c

- Interface com o usuário

LISTA DINÂMICA ENCADEADA



Implementando

```
//Arquivo ListaSequencial.h
#define MAX 10
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,media;
};
typedef struct lista Lista;
Lista* cria_lista();
```

```
//Arquivo ListaDinEncadeada.h
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,media;
};
typedef struct elemento* Lista;
Lista* cria_lista();
```

LISTA DINÂMICA ENCADEADA

Implementando

//Arquivo ListaSequencial.c

```
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

//Definição do tipo lista
struct lista{
    int qtd;
    struct aluno dados[MAX];
};
typedef struct lista Lista;
```

//Arquivo ListaDinEncadeada.c

```
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadeada.h"

//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox; // * para um struct
};
typedef struct elemento Elem;
```

LISTA DINÂMICA ENCADEADA

Implementando

//main.c

```
#include "ListaSequencial.h"
// li = ponteiro para ponteiro **
Lista *li = cria_lista();
```

//Arquivo ListaDinEncadeada.c

```
Lista* cria_lista(){
    Lista* li = (Lista*)malloc(sizeof(Lista));
    if(li != NULL)
        *li = NULL;
    return li;
}
```

LISTA DINÂMICA ENCADEADA

LIBERA LISTA

//Arquivo ListaDinEncadeada.h

```
void libera_lista(Lista* li);
```

//main.c

```
libera_lista(li);
```

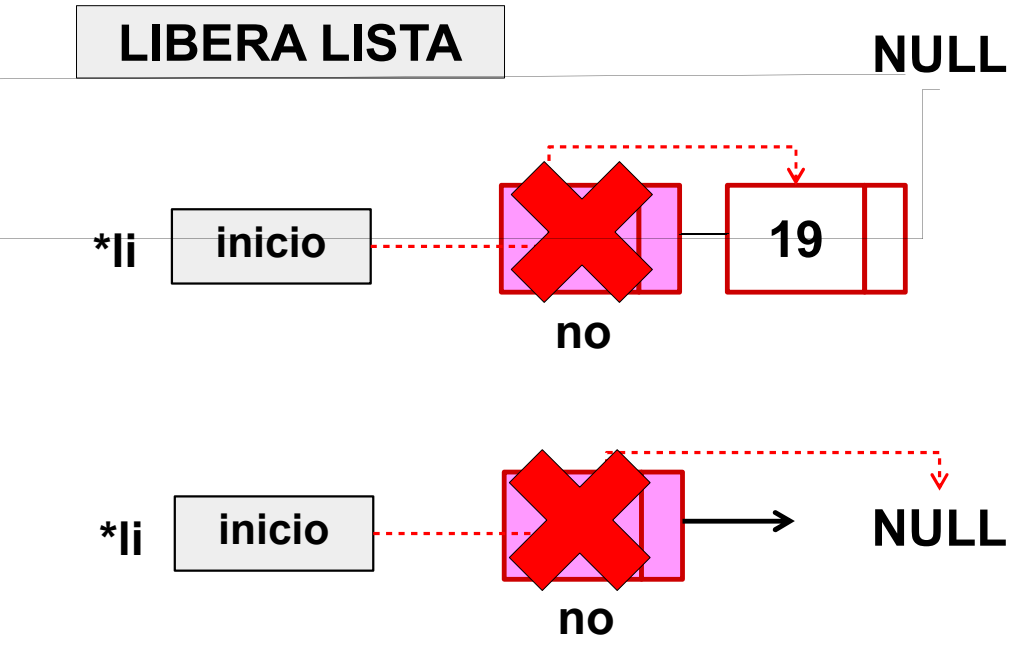
Implementando

//Arquivo ListaDinEncadeada.c

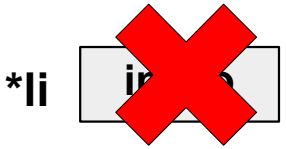
```
void libera_lista(Lista* li){
    if(li != NULL){
        Elem* no;
       // verifico 1º elemento
        while((*li) != NULL){
            no = *li;
            *li = (*li)->prox;
            free(no);
        }
        free(li);
    }
}
```

LISTA DINÂMICA ENCADEADA

Implementando




```
// verifico 1º elemento
 while((*li) != NULL){
      no = *li;
      *li = (*li)->prox;
      free(no);
}
free(li);
```



LISTA DINÂMICA ENCADEADA

Informações básicas

Tamanho | Lista cheia | Lista vazia

LISTA DINÂMICA ENCADEADA

Informações básicas

Tamanho

//Arquivo ListaDinEncadeada.h

```
int tamanho_lista(Lista* li);
```

//main.c

```
int x = tamanho_lista(li);
```

//Arquivo ListaDinEncadeada.c

```
int tamanho_lista(Lista* li){
    if(li == NULL) return 0;
    int cont = 0;
    Elem* no = *li;
    while(no != NULL){
        cont++;
        no = no->prox;
    }
    return cont;
}
```

LISTA DINÂMICA ENCADEADA

Informações básicas

Lista cheia

—

//Arquivo ListaDinEncadeada.h

```
int lista_cheia(Lista* li);
```

//main.c

```
if(lista_cheia(li))
```

//Arquivo ListaDinEncadeada.c

```
int lista_cheia(Lista* li){
    return 0;
}
```

LISTA DINÂMICA ENCADEADA

Informações básicas

Lista vazia

//Arquivo ListaDinEncadeada.h

```
int lista_vazia(Lista* li);
```

//main.c

```
if(lista_vazia(li))…
```

//Arquivo ListaDinEncadeada.c

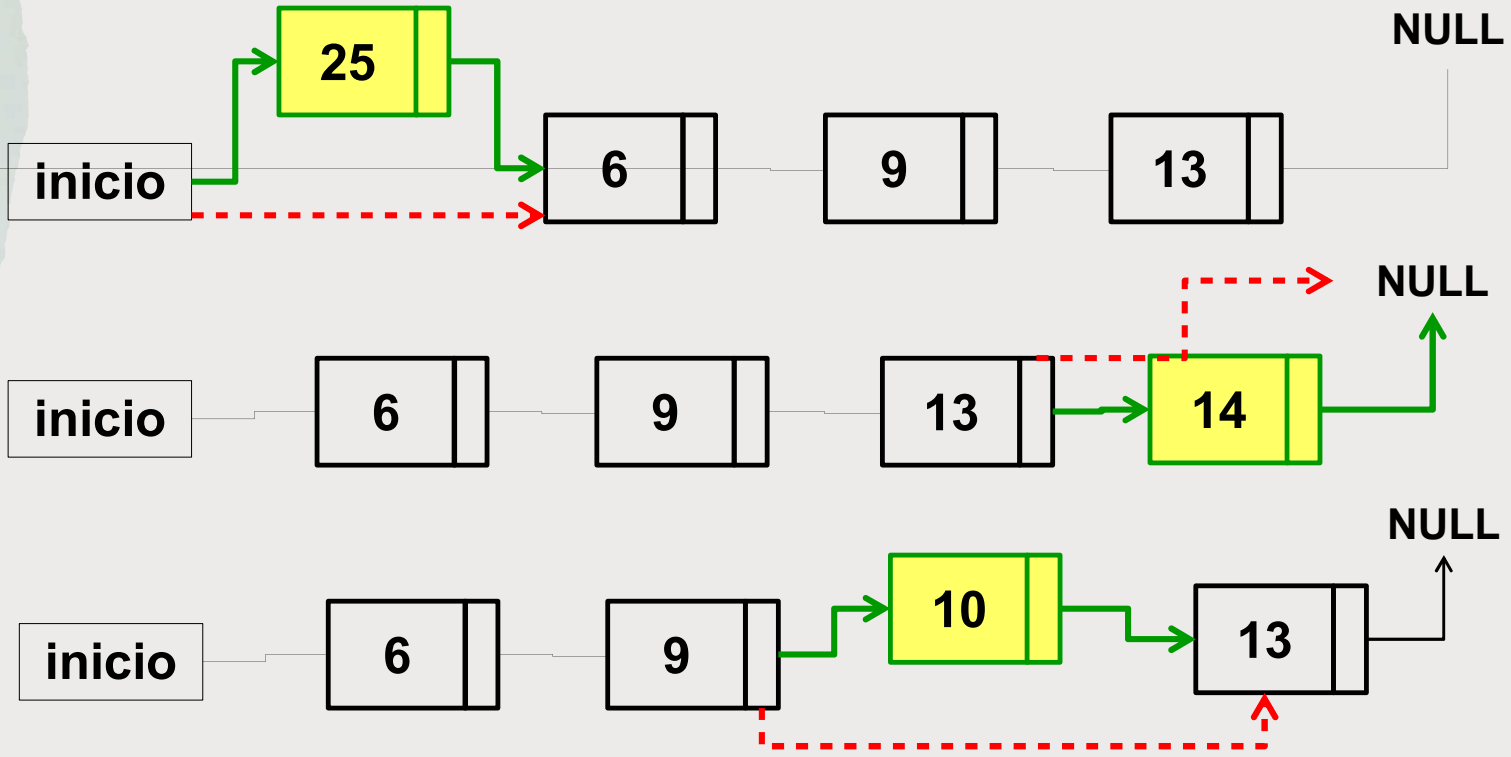
```
int lista_vazia(Lista* li){
    if(li == NULL)
        return 1;
    if(*li == NULL)
        return 1;
    return 0;
}
```

LISTA DINÂMICA ENCADEADA

Inserção

LISTA DINÂMICA ENCADEADA

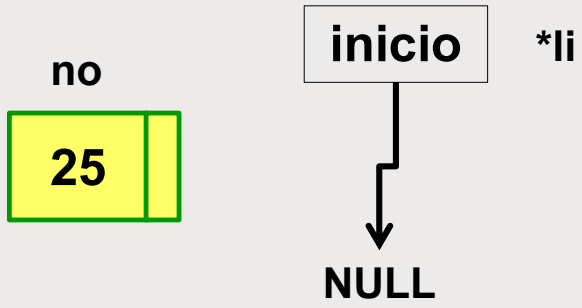
Inserção

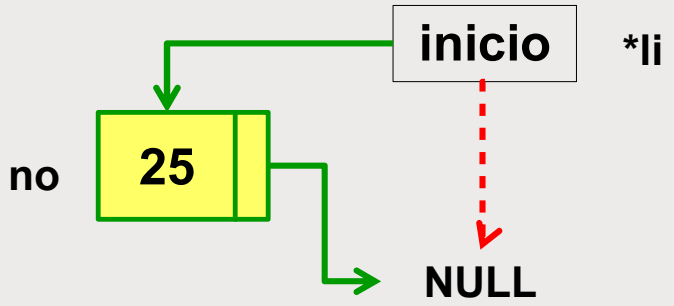


LISTA DINÂMICA ENCADEADA

Inserção

Lista vazia



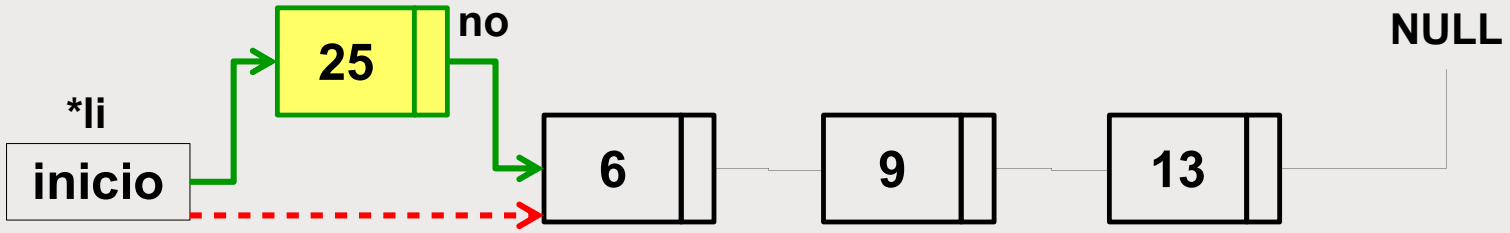



```
no->dados = al;
no->prox = NULL;
*li = no;
```

LISTA DINÂMICA ENCADEADA

Inserção

Inicio da lista



//Arquivo ListaDinEncadeada.h

```
int insere_lista_inicio(Lista* li, struct aluno al);
```

//main.c

```
Int x = insere_lista_inicio(li, dados_aluno)
```

//Arquivo ListaDinEncadeada.c

```
int insere_lista_inicio(Lista* li, struct aluno al)
{
    if(li == NULL) return 0;
    Elem* no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL) return 0;
    no->dados = al;
    no->prox = (*li); // recebe o end. do 1° no
    *li = no;
    return 1;
}
```

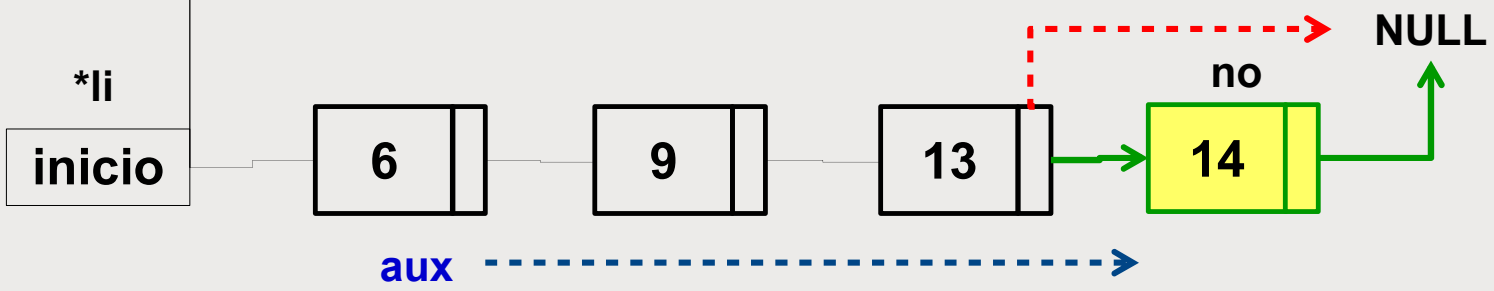
LISTA DINÂMICA ENCADEADA

Inserção

Final da lista

//Arquivo ListaDinEncadeada.h
int insere_lista_final(Lista* li, struct aluno al);

//main.c
Int x = insere_lista_final(li, dados_aluno)



//Arquivo ListaDinEncadeada.c

```
int insere_lista_final(Lista* li, struct aluno al)
{
    if(li == NULL) return 0;
    Elem* no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL) return 0;
    no->dados = al;
    no->prox = NULL;
    if((*li) == NULL){ //lista vazia
     *li = no;
}else{
     Elem *aux = *li;
     while(aux->prox != NULL){
          aux = aux->prox;
     }
     aux->prox = no;
   }
   return 1;
}
```

LISTA DINÂMICA ENCADEADA

Inserção

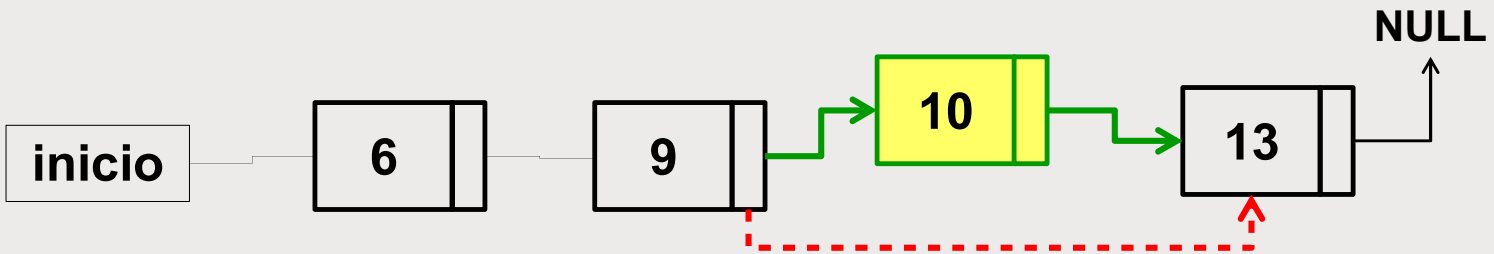
Meio da lista (ordenada)

//Arquivo ListaDinEncadeada.h

```
int insere_lista_ordenada(Lista* li, struct aluno al);
```

//main.c

```
Int x = insere_lista_ordenada(li, dados_aluno)
```



//Arquivo ListaDinEncadeada.c

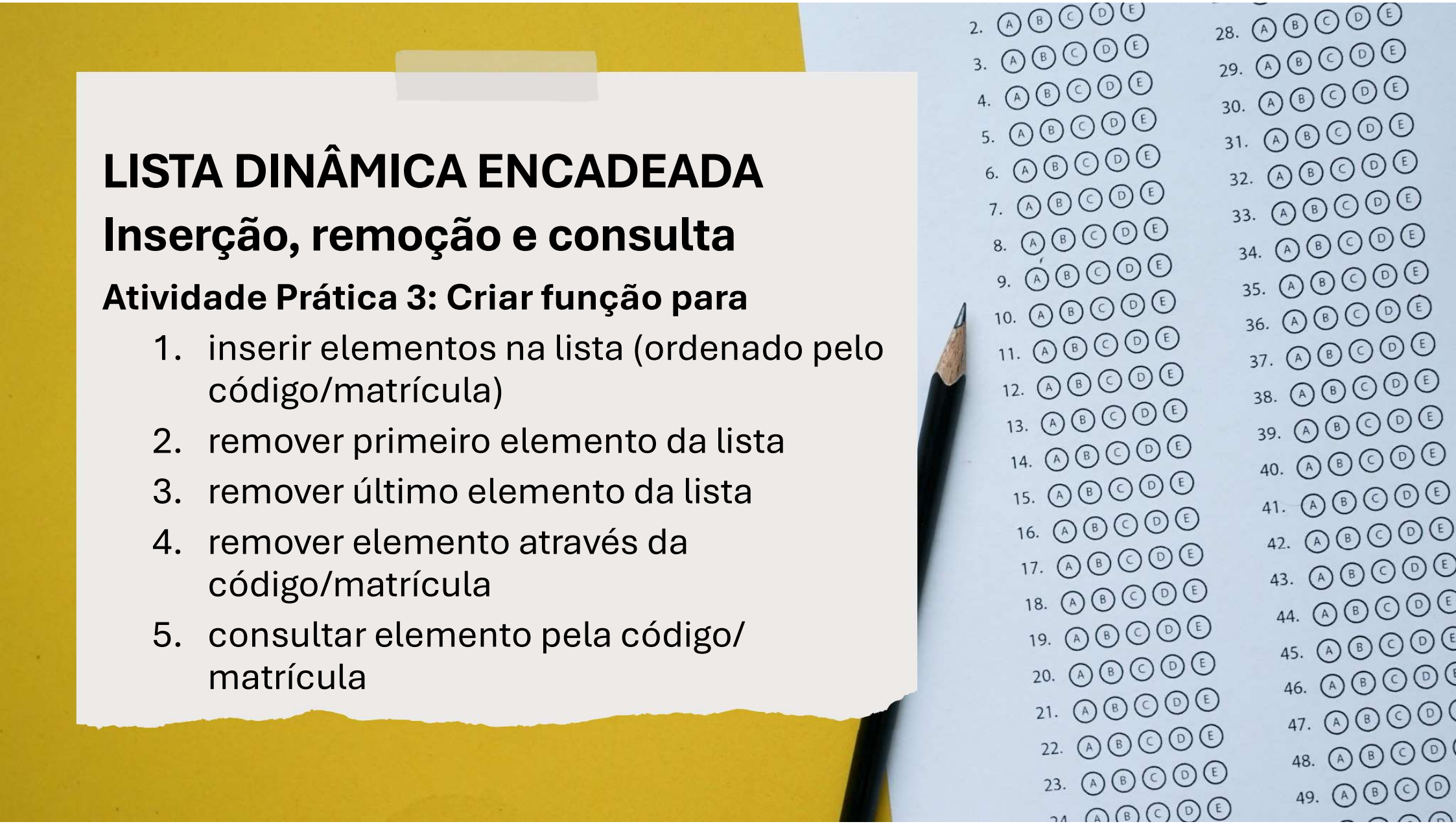
```
int insere_lista_ordenada(Lista* li, struct aluno al)
{
    if(li == NULL) return 0;
    Elem* no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL) return 0;
    no->dados = al;
    if((*li) == NULL){ //lista vazia
      no->prox = NULL;
      *li = no;
      return 1;
    }else{
      Elem *ant, *atual = *li;
      while(atual != NULL && atual->dados.matricula < al.matricula){
                ant = atual;
                atual = atual->prox;
      }
      if(atual == *li){//insere início
        no->prox = (*li);
        *li = no;
      }else{
        no->prox = atual;
        ant->prox = no;
      }
      return 1;
    }
}
```

LISTA DINÂMICA ENCADEADA

Inserção, remoção e consulta

Atividade Prática 3: Criar função para

1. inserir elementos na lista (ordenado pelo código/matrícula)
2. remover primeiro elemento da lista
3. remover último elemento da lista
4. remover elemento através da código/matrícula
5. consultar elemento pela código/ matrícula



1. CONTROLE DE NOTAS (AV3)

I. Usando lista dinâmica encadeada, escreva um programa que seja capaz de calcular a avaliação final dos alunos de uma disciplina a partir da seguinte fórmula expressa pela Equação abaixo:

$$AV_1 = \left(\left(\frac{AP_{1ªmn} + AP_{2ªmn} + AP_{3ªmn} + AP_{4ªmn}}{4}\right) * 0{,}4\right) + (NP * 0{,}6))$$

AP – Atividade Prática: Subscrito "1ªmn": "primeira maior nota". Os demais subscritos seguem a ordem representada pelo número.

1. CONTROLE DE NOTAS (AV3)

O programa deve ler os seguintes dados: **código do aluno, nome do aluno, notas das atividades práticas e nota final.**

O programa deve permitir:

- inserir alunos na lista (**ordenado pela nota final**)
- remover primeiro aluno da lista
- remover último aluno da lista
- remover aluno através do código/matrícula
- consultar aluno pelo código/matrícula