Curso: Ciência da Computação

Disciplina: Estrutura de Dados 1

Professor: Clayton Zambon

centro universitário unifacvest

3. Lista

3.5.Dinâmica Encadeada Circular;

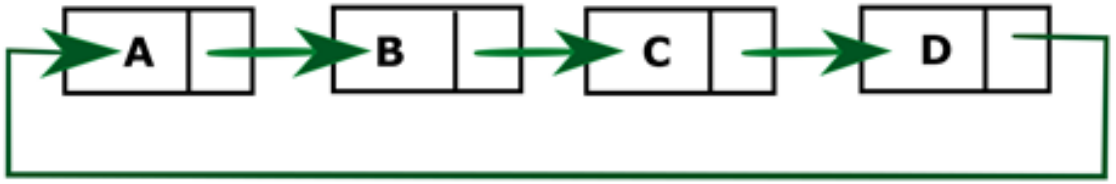
3. Lista

3.5. Dinâmica Encadeada Circular

- Tipo de lista onde cada elemento aponta para o seu sucessor e o último elemento aponta para o primeiro da Lista;

- Usa um ponteiro especial para o primeiro elemento da lista;

- Não existe uma indicação de final da Lista.



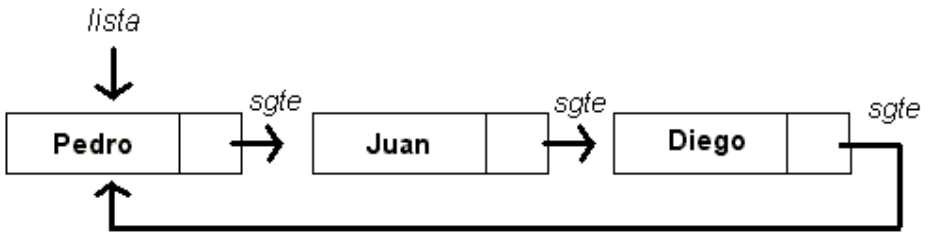
3. Lista
3.5. Dinâmica Encadeada Circular

- Cada elemento é tratado como um ponteiro que é alocado dinamicamente, a medida que os dados são inseridos;

- Para guardar o primeiro elemento, utilizamos um ponteiro para ponteiro;

- Um ponteiro para ponteiro pode guardar o endereço de um ponteiro;

- Assim , fica fácil mudar quem está no início da lista mudando o conteúdo do ponteiro para ponteiro;



3. Lista
3.5. Dinâmica Encadeada Circular

- Vantagens:

- Melhor utilização dos recursos de memória;
- Não precisa movimentar os elementos nas operações de inserção e remoção;
- Possibilidade de percorrer a lista diversas vezes;
- Não precisamos considerar casos especiais de inclusão e remoção de elementos (primeiro e último);

- Desvantagens:

- Acesso indireto aos elementos;
- Necessidade de percorrer a lista para acessar um elemento;
- Lista não possui final definido;

3. Lista
3.5. Dinâmica Encadeada Circular

- Quando utilizar este tipo de Lista?

- Quando não sei o espaço mínimo ou máximo da lista para execução do programa;
- Quando a Inserção/Remoção em lista ordenada são operações mais frequentes;
- Quando há necessidade de voltar ao primeiro item da lista depois de percorrê-la.

3. Lista
3.5. Dinâmica Encadeada Circular

Implementando um Lista Dinâmica Encadeada Circular

- Arquivo "ListaDinEncadCirc.h": definir

- Protótipos das funções;
- O tipo de dados armazenado na lista;
- O ponteiro lista;

- Arquivo "ListaDinEncadCirc.c": definir

- O tipo de dados lista;
- Implementar as funções;

- Arquivo "main.c": definir

- Fazer as chamadas das funções;

3. Lista
3.5. Dinâmica Encadeada Circular

- Definindo os Tipos, Structs e Ponteiro;

```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protótipos

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

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;

}
```

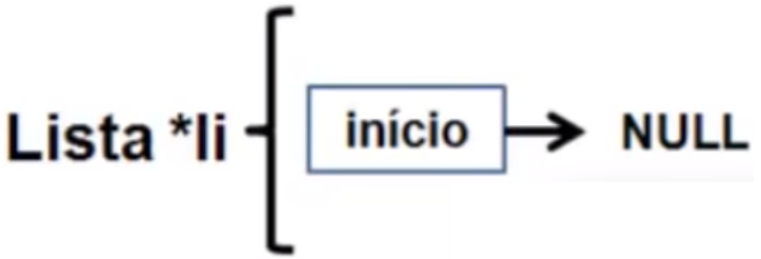
3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função de “criar_lista”;

```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
Lista* cria_lista();
```

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    li = cria_lista();
}
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protótipos

//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//*****************
Lista* cria_lista(){
    Lista* li = (Lista*) malloc(sizeof(Lista));
    if(li != NULL)
        *li = NULL;
    return li;
}
```



3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função de “liberar_lista”;

```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
void libera_lista(Lista* li);
```

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    li = cria_lista();
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protótipos

//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//*****************
void libera_lista(Lista* li){
    if(li != NULL && (*li) != NULL){
        Elem *aux, *no = *li;
        while((*li) != no->prox){
            aux = no;
            no = no->prox;
            free(aux);
        }
        free(no);
        free(li);
    }
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “tamanho_lista”;

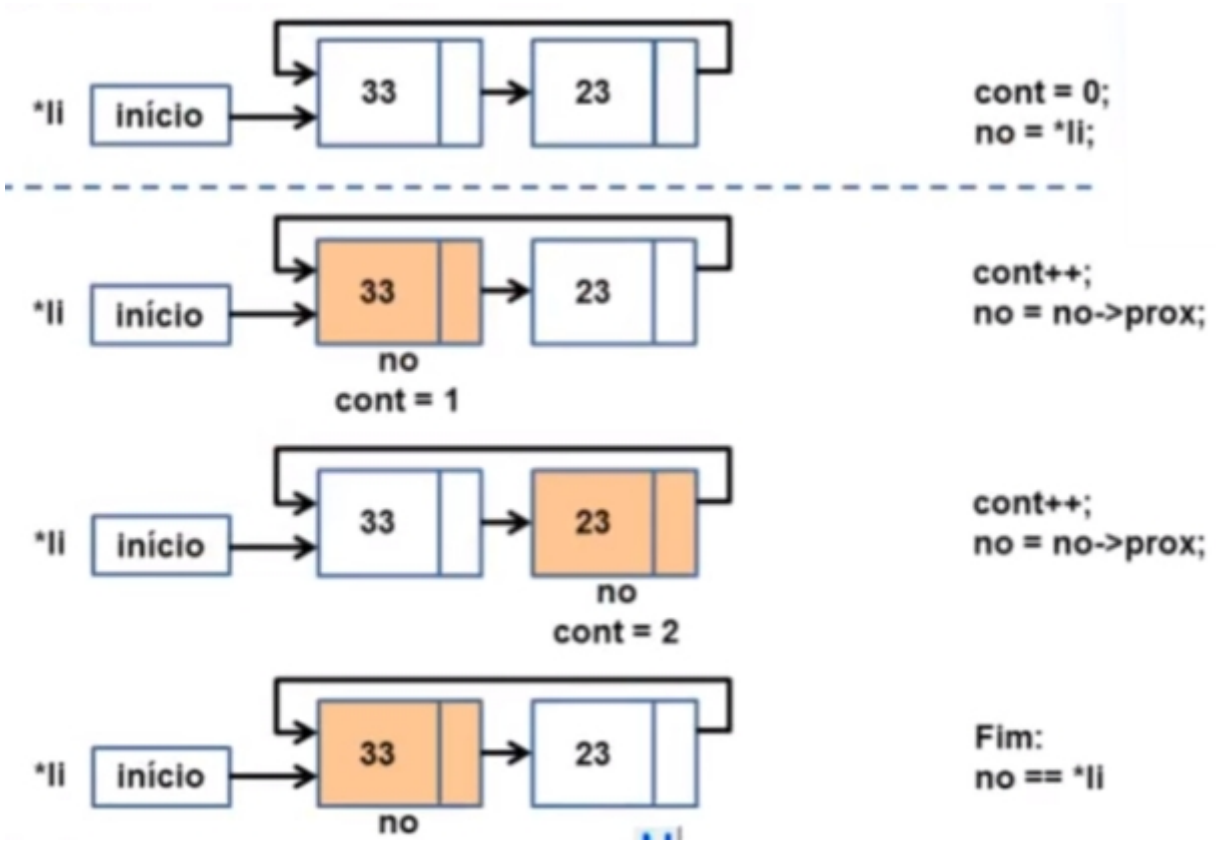
```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
int tamanho_lista(Lista* li);
```

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    li = cria_lista();
    int x = tamanho_lista(li);
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protótipo

//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//*****************
int tamanho_lista(Lista* li){
    if(li == NULL || (*li) == NULL)
        return 0;
    int cont = 0;
    Elem* no = *li;
    do{
        cont++;
        no = no->prox;
    }while(no != (*li));
    return cont;
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “tamanho_lista”;



3. Lista
3.5. Dinâmica Encadeada Circular

- **Implementando a Função "lista_cheia";**
  - Em uma lista Dinâmica não existe o conceito de lista cheia;

```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
int lista_cheia(Lista* li);
```

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    li = cria_lista();
    x = lista_cheia(li);
    int x = tamanho_lista(li);
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Pr

//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//*****************
int lista_cheia(Lista* li){
    return 0;
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função "lista_vazia";

```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
int lista_vazia(Lista* li);
```

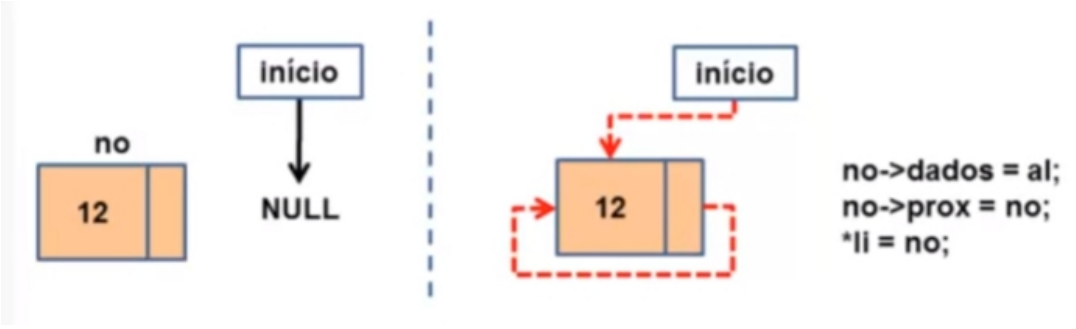
```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    li = cria_lista();
    x = lista_cheia(li);
    x = lista_vazia(li);
    int x = tamanho_lista(li);
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Prot

//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//*****************
int lista_vazia(Lista* li){
    if(li == NULL)
        return 1;
    if(*li == NULL)
        return 1;
    return 0;
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando as funções de inserção. Podemos inserir:

- No início;
- No meio;
- No final;
- Existe o caso onde a inserção é feita em uma Lista que está vazia;
- A inserção é parecida com a Lista Dinâmica Encadeada. Deve-se apenas considerar que agora temos o último elemento apontando para o primeiro.



3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “insere_lista_inicio”;

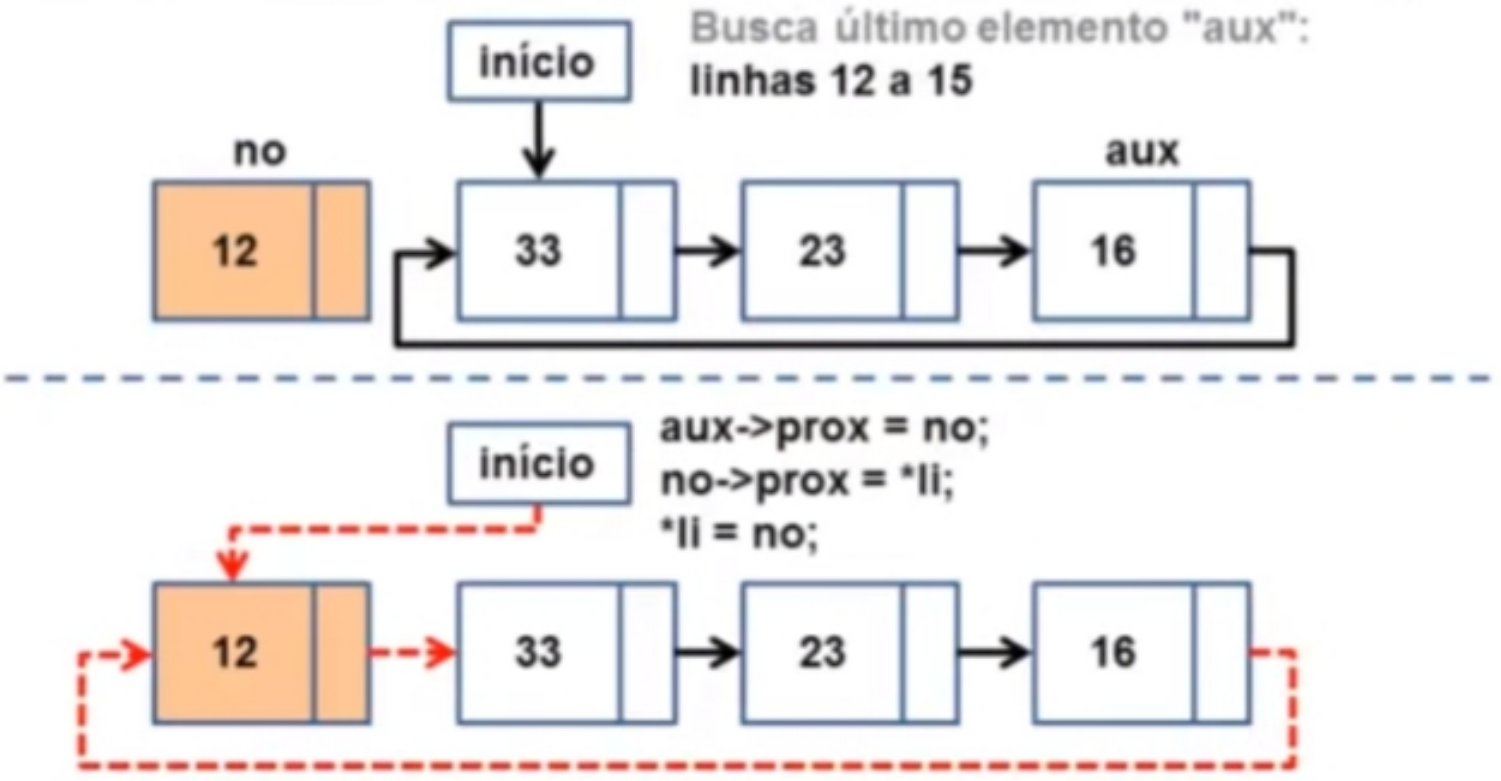
```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

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

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    struct aluno dados_aluno;
    li = cria_lista();
    x = lista_cheia(li);
    x = lista_vazia(li);
    x = insere_lista_inicio(li, dados_aluno);
    int x = tamanho_lista(li);
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protótipos
//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
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;
    if((*li) == NULL){//lista vazia: insere início
        *li = no;
        no->prox = no;
    }else{
        Elem *aux = *li;
        while(aux->prox != (*li)){
            aux = aux->prox;
        }
        aux->prox = no;
        no->prox = *li;
        *li = no;
    }
    return 1;
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “insere_lista_inicio”;



3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “insere_lista_final”;

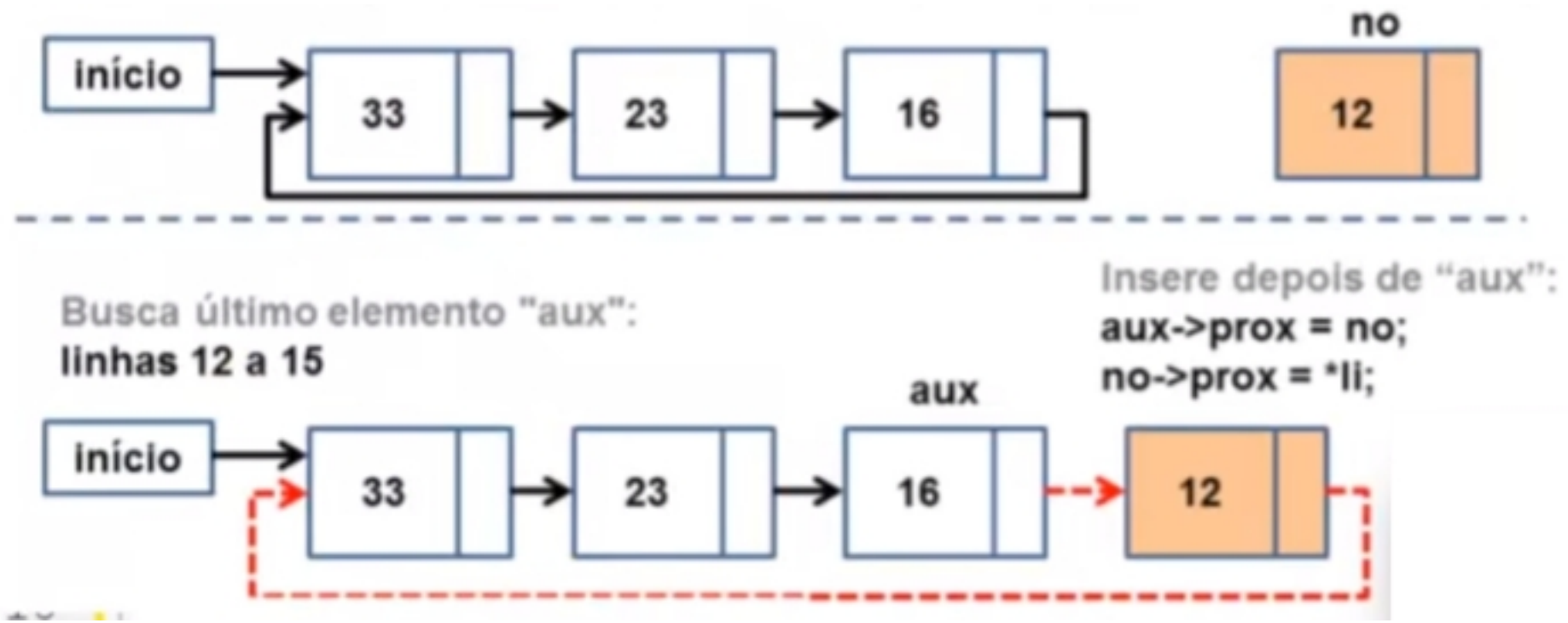
```h
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
int insere_lista_final(Lista* li, struct aluno al);
```

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    struct aluno dados_aluno;
    li = cria_lista();
    x = lista_cheia(li);
    x = lista_vazia(li);
    x = insere_lista_final(li, dados_aluno);
    int x = tamanho_lista(li);
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protótipos
//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//**************************************
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;
    if((*li) == NULL){//lista vazia: insere início
        *li = no;
        no->prox = no;
    }else{
        Elem *aux = *li;
        while(aux->prox != (*li)){
            aux = aux->prox;
        }
        aux->prox = no;
        no->prox = *li;
    }
    return 1;
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “insere_lista_final”;



3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “insere_lista_ordenada”;

```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

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

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    struct aluno dados_aluno;
    li = cria_lista();
    x = lista_cheia(li);
    x = lista_vazia(li);
    x = insere_lista_ordenada(li, dados_aluno);
    int x = tamanho_lista(li);
    libera_lista(li);
}
```

```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){//insere início
        *li = no;
        no->prox = no;
        return 1;
    }
    else{
        if((*li)->dados.matricula > al.matricula){//insere início
            Elem *atual = *li;
            while(atual->prox != (*li))//procura o último
                atual = atual->prox;
            no->prox = *li;
            atual->prox = no;
            *li = no;
            return 1;
        }
        Elem *ant = *li, *atual = (*li)->prox;
        while(atual != (*li) && atual->dados.matricula < al.matricula){
            ant = atual;
            atual = atual->prox;
        }
        ant->prox = no;
        no->prox = atual;
        return 1;
    }
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “imprime_lista”;

```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//*********************
void imprime_lista(Lista* li);
```

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    struct aluno dados_aluno;
    li = cria_lista();
    x = lista_cheia(li);
    x = lista_vazia(li);
    x = insere_lista_ordenada(li, dados_aluno);
    imprime_lista(li);
    int x = tamanho_lista(li);
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protótipos
//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//*************************************
void imprime_lista(Lista* li){
    if(li == NULL || (*li) == NULL)
        return;
    Elem* no = *li;
    do{
        printf("Matricula: %d\n",no->dados.matricula);
        printf("Nome: %s\n",no->dados.nome);
        printf("Notas: %f %f %f\n",no->dados.n1,
                                   no->dados.n2,
                                   no->dados.n3);
        printf("--------------------------------\n");
        no = no->prox;
    }while(no != (*li));
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

EXERCÍCIO:

- Fazer um menu para acessar as funções;

- Fazer as implementações descritas anteriormente;

- Popular a lista com informações do aluno na lista;

- Implementar mensagens para informar o usuário as ações que foram realizadas;

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando as funções de REMOÇÃO. Podemos REMOVER:

- Do início;
- Do meio;
- Do final;
- Os 3 tipos de remoção podem trabalhar juntos. A remoção pode remover um elementos específico da Lista, o qual pode estar no início, no meio ou no fim;
- Não é possível remover elementos de uma lista vazia. Após remover o último nó a lista ficará vazia;

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “remove_lista_inicio”;

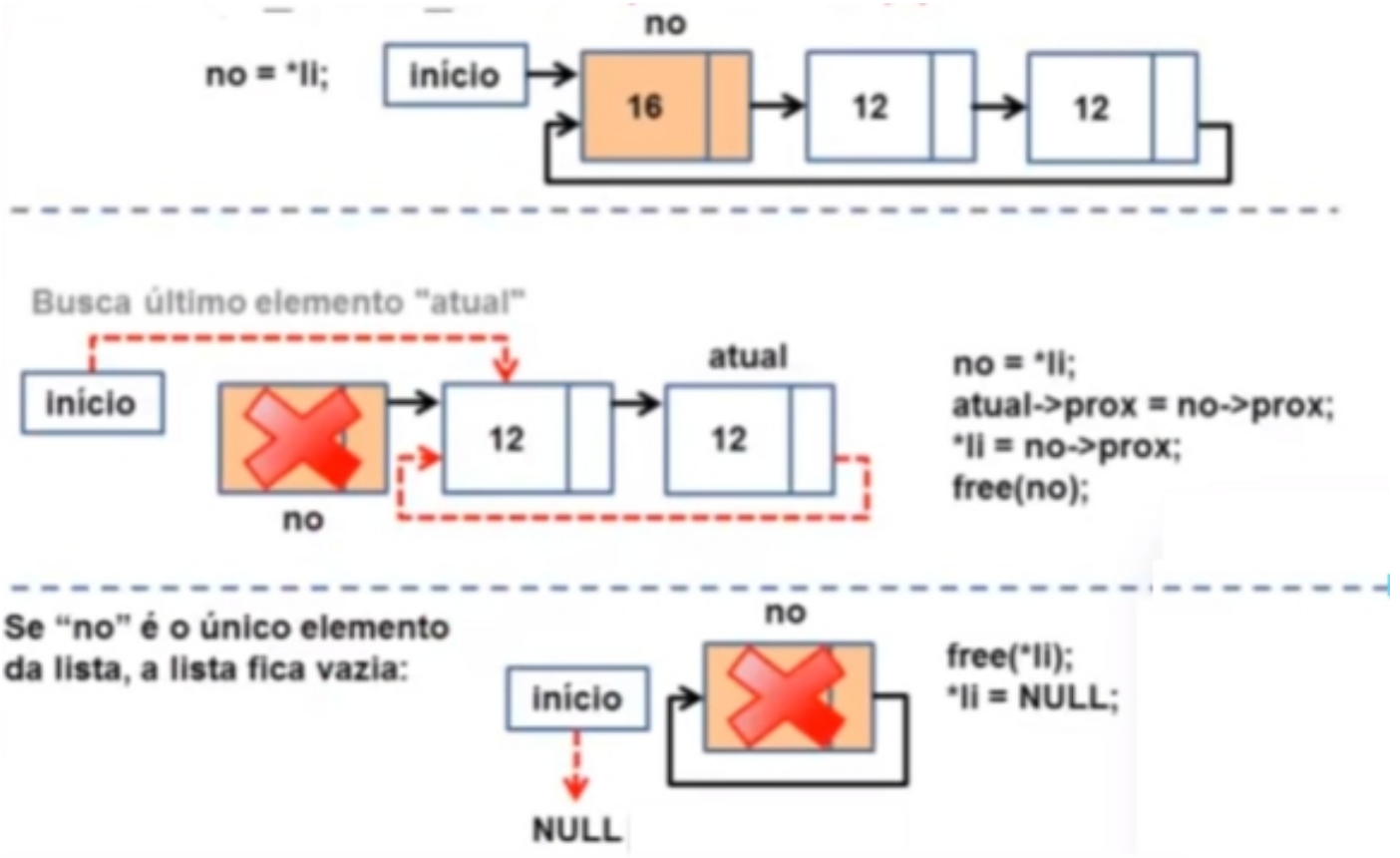
```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//*********************
int remove_lista_inicio(Lista* li);
```

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    struct aluno dados_aluno;
    li = cria_lista();
    x = lista_cheia(li);
    x = lista_vazia(li);
    x = insere_lista_ordenada(li, dados_aluno);
    imprime_lista(li);
    int x = tamanho_lista(li);
    remove_lista_inicio(li);
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protótipos
//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//*************************************
int remove_lista_inicio(Lista* li){
    if(li == NULL)
        return 0;
    if((*li) == NULL)//lista vazia
        return 0;

    if((*li) == (*li)->prox){//lista fica vaza
        free(*li);
        *li = NULL;
        return 1;
    }
    Elem *atual = *li;
    while(atual->prox != (*li))//procura o último
        atual = atual->prox;

    Elem *no = *li;
    atual->prox = no->prox;
    *li = no->prox;
    free(no);
    return 1;
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “remove_lista_inicio”;



3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “remove_lista_final”;

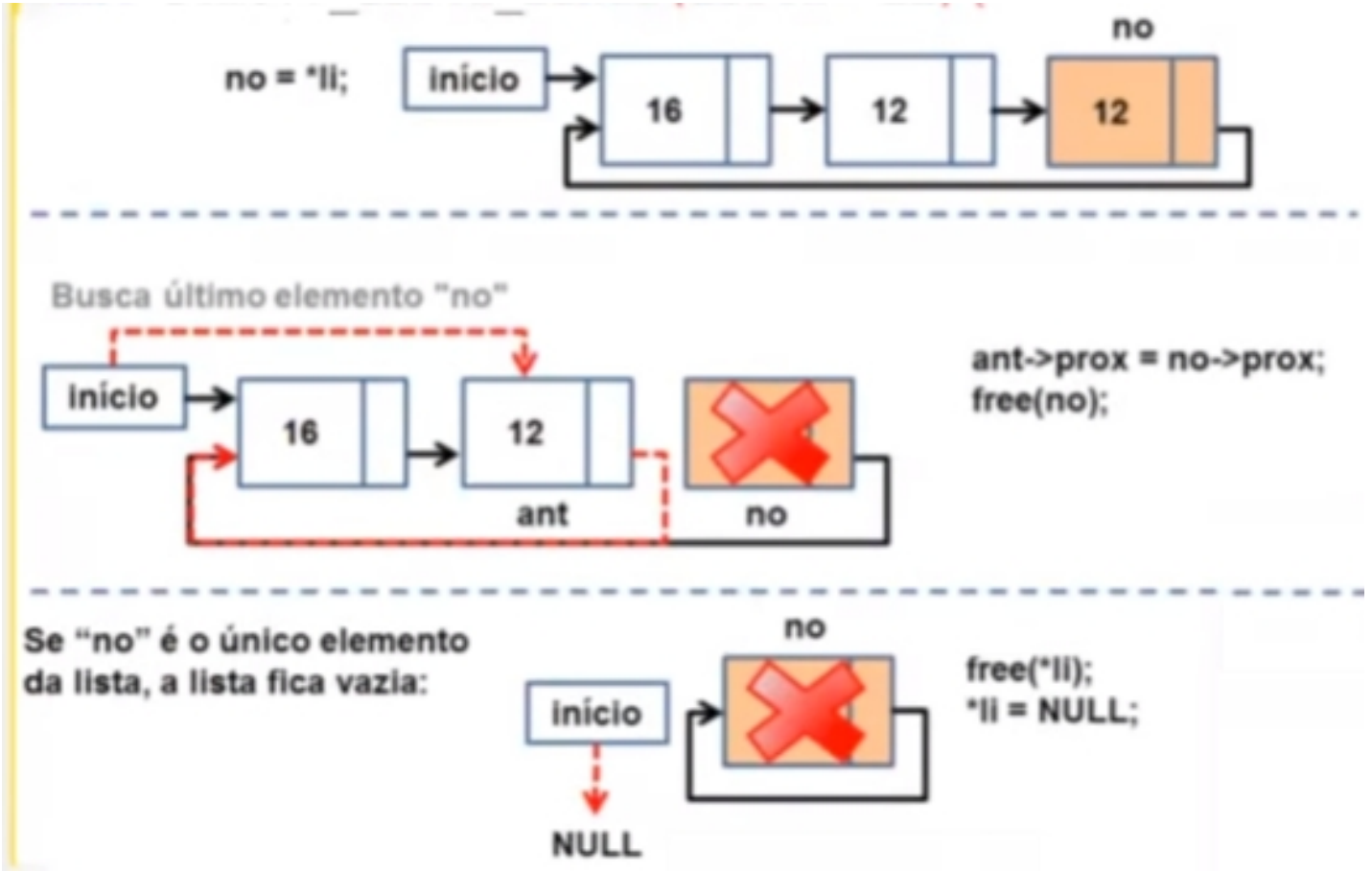
```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
int remove_lista_final(Lista* li);
```

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    struct aluno dados_aluno;
    li = cria_lista();
    x = lista_cheia(li);
    x = lista_vazia(li);
    x = insere_lista_ordenada(li, dados_aluno);
    imprime_lista(li);
    int x = tamanho_lista(li);
    remove_lista_final(li);
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protó
//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//**********************************
int remove_lista_final(Lista* li){
    if(li == NULL)
        return 0;
    if((*li) == NULL)//lista vazia
        return 0;

    if((*li) == (*li)->prox){//lista fica vaza
        free(*li);
        *li = NULL;
        return 1;
    }
    Elem *ant, *no = *li;
    while(no->prox != (*li)){//procura o último
        ant = no;
        no = no->prox;
    }
    ant->prox = no->prox;
    free(no);
    return 1;
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “remove_lista_final”;



3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “remove_lista”. Remove um elemento específico da lista.

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    struct aluno dados_aluno;
    int matricula_aluno;
    li = cria_lista();
    x = lista_cheia(li);
    x = lista_vazia(li);
    x = insere_lista_ordenada(li, dados_aluno);
    imprime_lista(li);
    int x = tamanho_lista(li);
    remove_lista(li, matricula_aluno);
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
int remove_lista(Lista* li, int mat);
```

3. Lista
3.5. Dinâmica Encadeada Circular

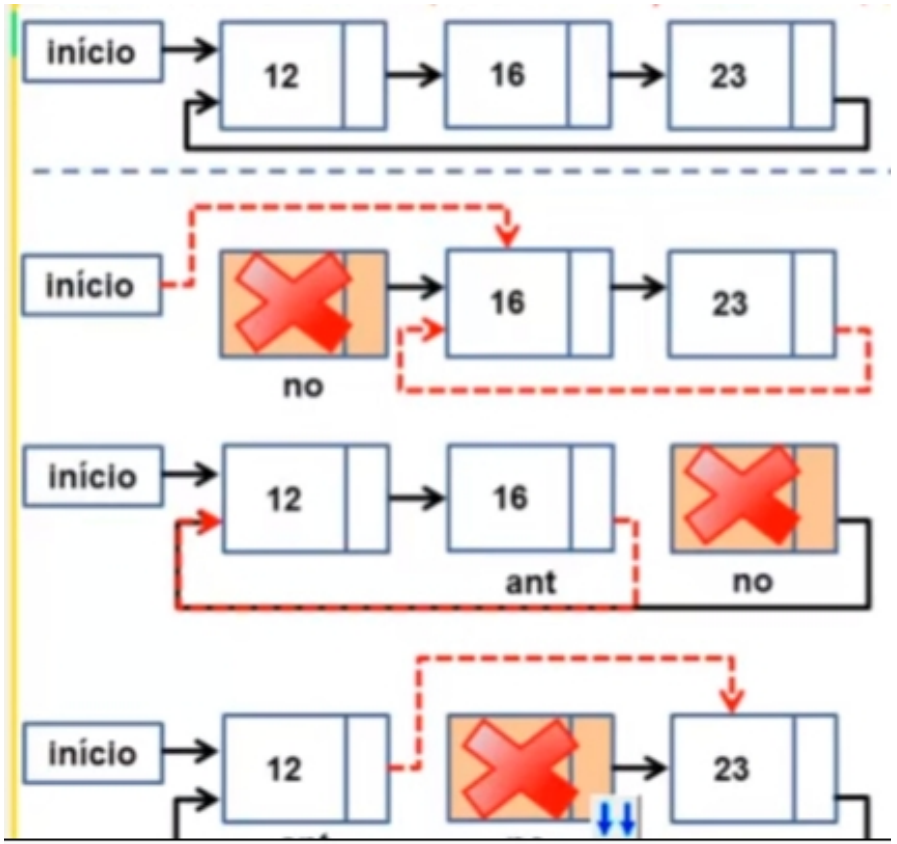
- Implementando a Função "remove_lista".

Remove um elemento específico da lista.

```
int remove_lista(Lista* li, int mat){
    if(li == NULL)
        return 0;
    if((*li) == NULL)//lista vazia
        return 0;
    Elem *no = *li;
    if(no->dados.matricula == mat){//remover do início
        if(no == no->prox){//lista fica vaza
            free(no);
            *li = NULL;
            return 1;
        }else{
            Elem *ult = *li;
            while(ult->prox != (*li))//procura o último
                ult = ult->prox;
            ult->prox = (*li)->prox;
            *li = (*li)->prox;
            free(no);
            return 1;
        }
    }
    Elem *ant = no;
    no = no->prox;
    while(no != (*li) && no->dados.matricula != mat){
        ant = no;
        no = no->prox;
    }
    if(no == *li)//não encontrado
        return 0;

    ant->prox = no->prox;
    free(no);
    return 1;
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

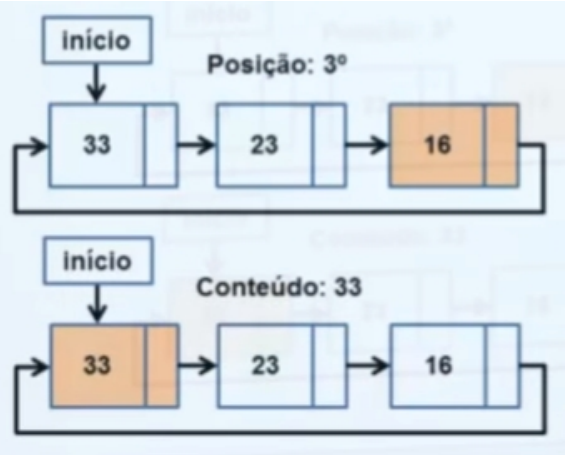
- Implementando a Função “remove_lista”. Remove um elemento específico da lista.



3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando as funções de CONSULTA. Podemos CONSULTAR um elemento de uma lista de duas formas:

- Pela posição;
- Pelo Conteúdo;
- Ambos dependem de busca sendo necessário percorrer a lista até encontrar o elemento desejado;



3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “consulta_lista_pos”;

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    struct aluno dados_aluno;
    int matricula_aluno, posicao;
    li = cria_lista();
    x = lista_cheia(li);
    x = lista_vazia(li);
    x = insere_lista_ordenada(li, dados_aluno);
    imprime_lista(li);
    int x = tamanho_lista(li);
    x = consulta_lista_pos(li, posicao, &dados_aluno);
    remove_lista(li, matricula_aluno);
    libera_lista(li);
}
```

```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
int consulta_lista_pos(Lista* li, int pos, struct aluno *al);
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “consulta_lista_pos”;

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protótipos
//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//*************************************
int consulta_lista_pos(Lista* li, int pos, struct aluno *al){
    if(li == NULL || (*li) == NULL || pos <= 0)
        return 0;
    Elem *no = *li;
    int i = 1;
    while(no->prox != (*li) && i < pos){
        no = no->prox;
        i++;
    }
    if(i != pos)
        return 0;
    else{
        *al = no->dados;
        return 1;
    }
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “consulta_lista_mat”;

```c
//Arquivo ListaDinEncadCirc.h

struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct elemento* Lista;
//**********************
int consulta_lista_mat(Lista* li, int mat, struct aluno *al);
```

```c
//Arquivo main.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h"
int main(){
    Lista *li;
    struct aluno dados_aluno;
    int matricula_aluno, posicao;
    li = cria_lista();
    x = lista_cheia(li);
    x = lista_vazia(li);
    x = insere_lista_ordenada(li, dados_aluno);
    imprime_lista(li);
    int x = tamanho_lista(li);
    x = consulta_lista_mat(li, matricula_aluno, &dados_aluno);
    remove_lista(li, matricula_aluno);
    libera_lista(li);
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

- Implementando a Função “consulta_lista_mat”;

```c
//Arquivo ListaDinEncadCirc.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaDinEncadCirc.h" //inclui os Protótipos
//Definição do tipo lista
struct elemento{
    struct aluno dados;
    struct elemento *prox;
};
typedef struct elemento Elem;
//*************************************
int consulta_lista_mat(Lista* li, int mat, struct aluno *al){
    if(li == NULL || (*li) == NULL)
        return 0;
    Elem *no = *li;
    while(no->prox != (*li) && no->dados.matricula != mat)
        no = no->prox;
    if(no->dados.matricula != mat)
        return 0;
    else{
        *al = no->dados;
        return 1;
    }
}
```

3. Lista
3.5. Dinâmica Encadeada Circular

EXERCÍCIO:

- Ao consultar pela posição ou matrícula, exibir as informações do elemento;

- Quando não encontrar um elemento exibir uma mensagem para o usuário;

- Quando não for possível realizar a operação, exibir a mensagem de lista vazia ou lista cheia conforme o caso.

- Quando inserir ou remover um elemento informar ao usuário se a operação ocorreu com sucesso ou não.

Referências

EDELWEISS, Nina; GALANTE, Renata. Estruturas de Dados. Porto Alegre, BOOKMAN, 2009.

HEINZLE, Roberto. Estruturas de Dados: implementações com C e Pascal. Blumenau, DIRETIVA, 2006.

TENENBAUM, Aron M. Estrutura de Dados usando C. São Paulo, Makron Books, 1995.

FORBELLONE, André Luiz Villar: EBERSPÄCHER, Henri Frederico. Lógica de Programação: a construção de algoritmos e estruturas de dados. 3. ed. São Paulo, PRENTICE HALL, 2005.

KOFFMAN, Elliot B.; WOLFGANG, Paul A. T. Objetos, abstração, estruturas de dados e projeto usando C++. Rio de Janeiro, LTC, 2008.

PEREIRA, Silvio do lago. Estruturas de dados fundamentais: conceitos e aplicações. São Paulo, Érica, 1996.

VILLAS, Marcos Viana et al. Estruturas de dados – Conceitos e técnicas de implementação. Rio de Janeiro, Campus, 1993.

VELOSO, Paulo et al. Estrutura de dados. Rio de Janeiro, Campus, 1996.

Canal do Youtube: Linguagem C Programação Descomplicada

Obrigado!