

Curso: Ciência da Computação

Disciplina: Estrutura de Dados 1

Professor: Clayton Zambon

3. Lista

3.5.Dinâmica Encadeada Circular;

3. Lista

3.5. Dinâmica Encadeada Circular

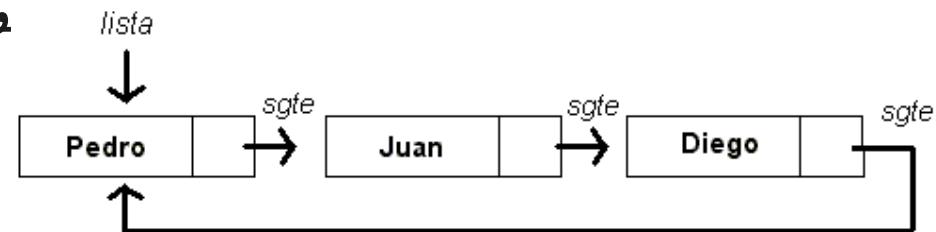
- Tipo de lista onde cada elemento aponta para o seu sucessor e o último elemento aponta para o primeiro da Lista;
- Usa um ponteiro especial para o primeiro elemento da lista;
- Não existe uma indicação de final da Lista.



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;

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float n1,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
10 //*****
11
```

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protótipos
5
6 //Definição do tipo lista
7 struct elemento{
8     struct aluno dados;
9     struct elemento *prox;
10 };
11 typedef struct elemento Elem;
...
```

```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5 int main(){
6     Lista *li;
7
8 }
9
```

3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função de “criar_lista”;

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float n1,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
//*****
10 Lista* cria_lista();
11
12
```

```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5
6 int main(){
7     Lista *li;
8     li = cria_lista();
9 }
```

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protótipos
5
6 //Definição do tipo lista
7 struct elemento{
8     struct aluno dados;
9     struct elemento *prox;
10};
11 typedef struct elemento Elem;
//*****
12
13 Lista* cria_lista(){
14     Lista* li = (Lista*) malloc(sizeof(Lista));
15     if(li != NULL)
16         *li = NULL;
17     return li;
18 }
```



3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função de “liberar_lista”;

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float n1,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
//*****
10 void libera_lista(Lista* li);
11
12
```

```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5 int main(){
6     Lista *li;
7     li = cria_lista();
8     libera_lista(li);
9 }
10
```

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protótipos
5
6 //Definição do tipo lista
7 struct elemento{
8     struct aluno dados;
9     struct elemento *prox;
10};
11 typedef struct elemento Elem;
//*****
12 void libera_lista(Lista* li){
13     if(li != NULL && (*li) != NULL){
14         Elem *aux, *no = *li;
15         while((*li) != no->prox){
16             aux = no;
17             no = no->prox;
18             free(aux);
19         }
20         free(no);
21         free(li);
22     }
23 }
24 }
```

3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função “tamanho_lista”;

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float n1,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
//*****
10 int tamanho_lista(Lista* li);
11
12
```

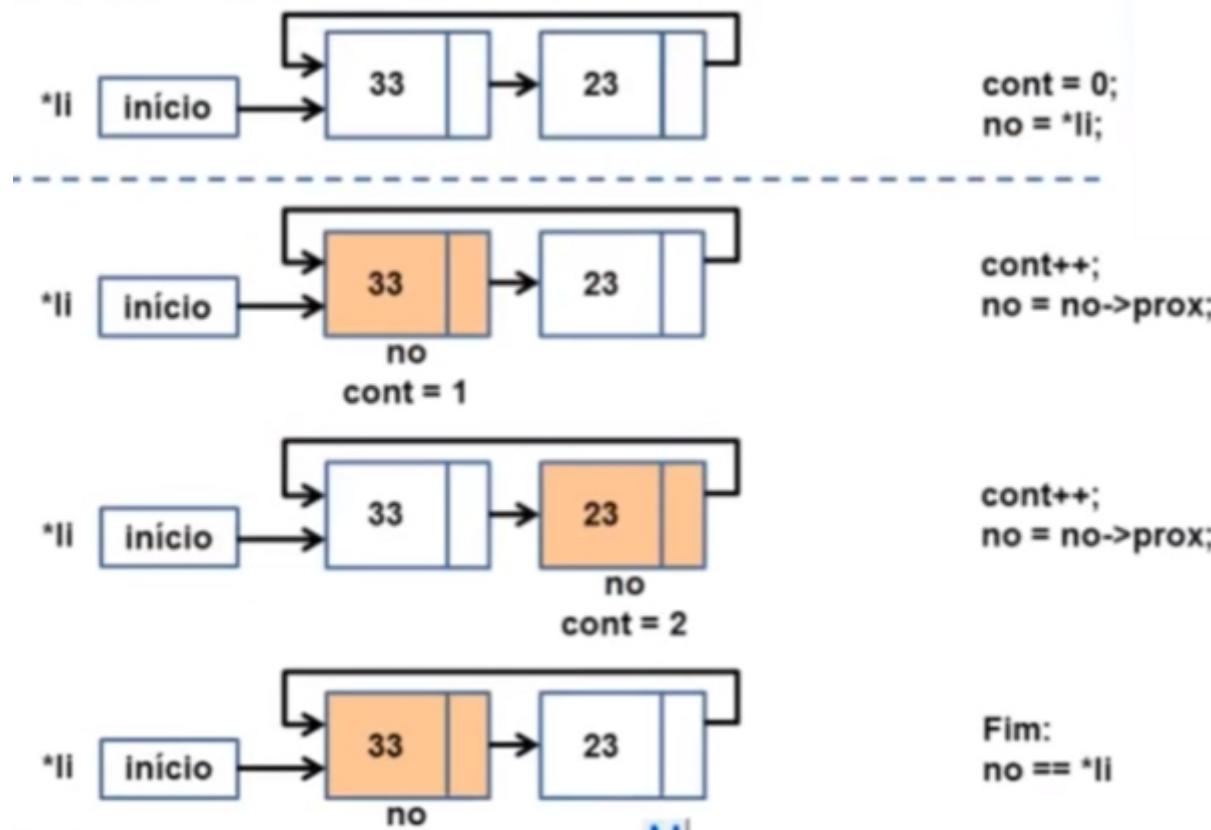
```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5
6 int main(){
7     Lista *li;
8     li = cria_lista();
9     int x = tamanho_lista(li);
10    libera_lista(li);
11 }
```

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protótipos
5
6 //Definição do tipo lista
7 struct elemento{
8     struct aluno dados;
9     struct elemento *prox;
10 };
11
12 typedef struct elemento Elem;
//*****
13 int tamanho_lista(Lista* li) {
14     if(li == NULL || (*li) == NULL)
15         return 0;
16     int cont = 0;
17     Elem* no = *li;
18     do{
19         cont++;
20         no = no->prox;
21     }while(no != (*li));
22     return cont;
23 }
```

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;

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float n1,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
10 //*****
11 int lista_cheia(Lista* li);
```

```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5
6 int main(){
7     Lista *li;
8     li = cria_lista();
9     x = lista_cheia(li);
10    int x = tamanho_lista(li);
11    libera_lista(li);
12 }
```

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Pr
5
6 //Definição do tipo lista
7 struct elemento{
8     struct aluno dados;
9     struct elemento *prox;
10 };
11
12 typedef struct elemento Elemt;
13 //*****
14 int lista_cheia(Lista* li){
15     return 0;
16 }
```

3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função “lista_vazia”;

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float nl,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
10 //*****
11 int lista_vazia(Lista* li);
```

```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5 int main(){
6     Lista *li;
7     li = cria_lista();
8     x = lista_cheia(li);
9     x = lista_vazia(li);
10    int x = tamanho_lista(li);
11    libera_lista(li);
12 }
```

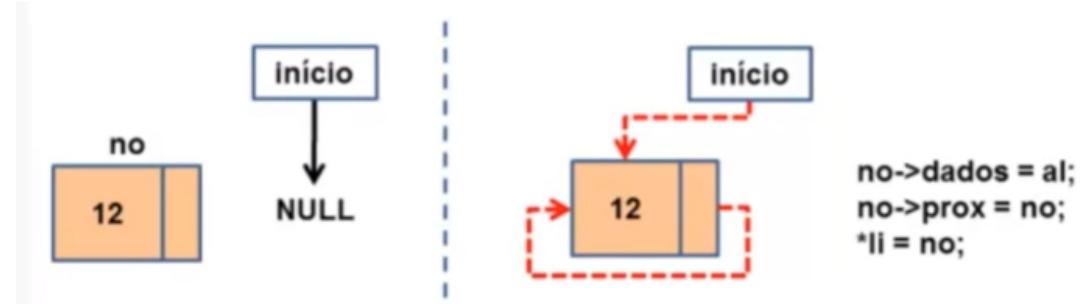
```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Prot
5
6 //Definição do tipo lista
7 struct elemento{
8     struct aluno dados;
9     struct elemento *prox;
10 };
11
12 typedef struct elemento Elemt;
13 //*****
14 int lista_vazia(Lista* li){
15     if(li == NULL)
16         return 1;
17     if(*li == NULL)
18         return 1;
19     return 0;
20 }
```

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”;

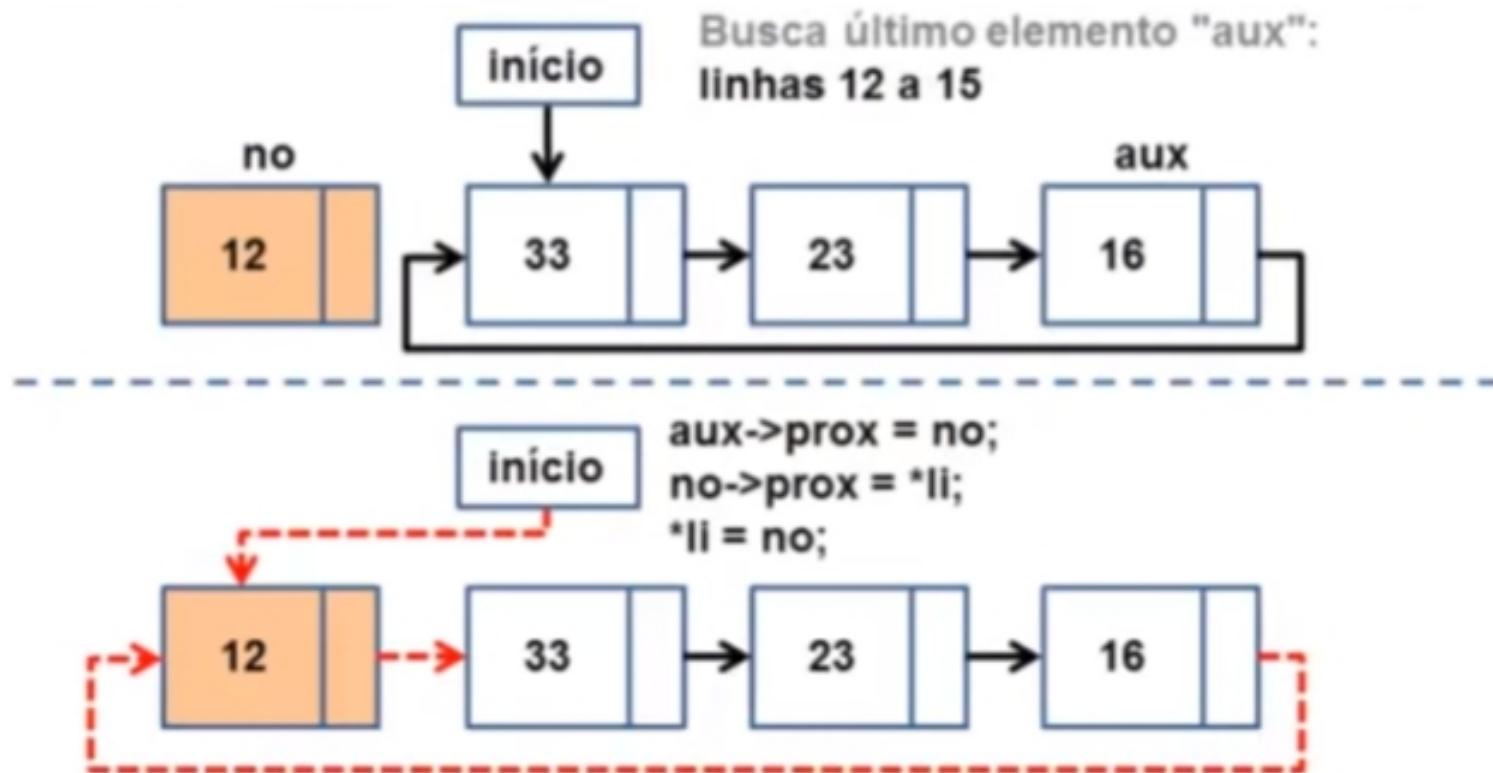
```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float nl,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
//*****
10 int insere_lista_inicio(Lista* li, struct aluno al);
11
12
13 //Arquivo main.c
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include "ListaDinEncadCirc.h"
17
18 int main(){
19     Lista *li;
20     struct aluno dados_aluno;
21     li = cria_lista();
22     x = lista_cheia(li);
23     x = lista_vazia(li);
24     x = insere_lista_inicio(li, dados_aluno);
25     int x = tamanho_lista(li);
26     libera_lista(li);
27 }
```

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protótipos
5
6 //Definição do tipo lista
7 struct elemento{
8     struct aluno dados;
9     struct elemento *prox;
10 };
11
12 typedef struct elemento Elem;
13
14 int insere_lista_inicio(Lista* li, struct aluno al){
15     if(li == NULL)
16         return 0;
17     Elem *no = (Elem*) malloc(sizeof(Elem));
18     if(no == NULL)
19         return 0;
20     no->dados = al;
21     if((*li) == NULL){ //lista vazia: insere inicio
22         *li = no;
23         no->prox = no;
24     }else{
25         Elem *aux = *li;
26         while(aux->prox != (*li)){
27             aux = aux->prox;
28         }
29         aux->prox = no;
30         no->prox = *li;
31         *li = no;
32     }
33     return 1;
34 }
```

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”;

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float nl,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
10 //*****
11 int insere_lista_final(Lista* li, struct aluno al);
12
```

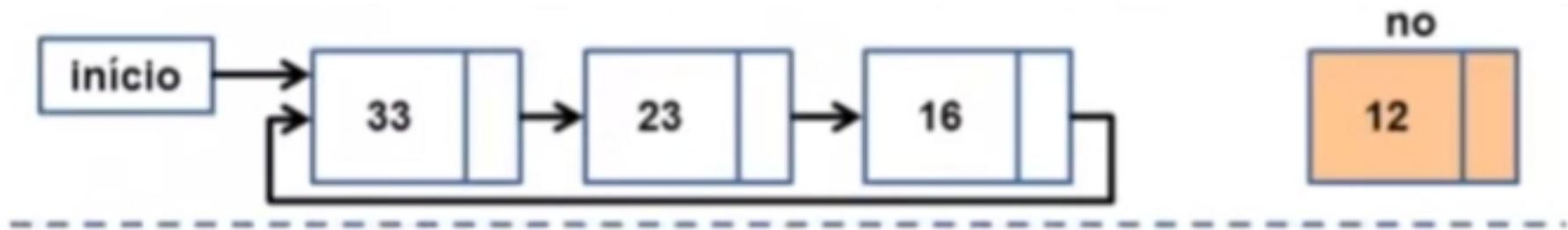
```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5 int main(){
6     Lista *li;
7     struct aluno dados_aluno;
8     li = cria_lista();
9     x = lista_cheia(li);
10    x = lista_vazia(li);
11    x = insere_lista_final(li, dados_aluno);
12    int x = tamanho_lista(li);
13    libera_lista(li);
14 }
```

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protótipos
5 //Definição do tipo lista
6 struct elemento{
7     struct aluno dados;
8     struct elemento *prox;
9 };
10 typedef struct elemento Elem;
11 //*****
12 int insere_lista_final(Lista* li, struct aluno al){
13     if(li == NULL)
14         return 0;
15     Elem *no = (Elem*) malloc(sizeof(Elem));
16     if(no == NULL)
17         return 0;
18     no->dados = al;
19     if((*li) == NULL){//lista vazia: insere inicio
20         *li = no;
21         no->prox = no;
22     }else{
23         Elem *aux = *li;
24         while(aux->prox != (*li)){
25             aux = aux->prox;
26         }
27         aux->prox = no;
28         no->prox = *li;
29     }
30     return 1;
31 }
```

3. Lista

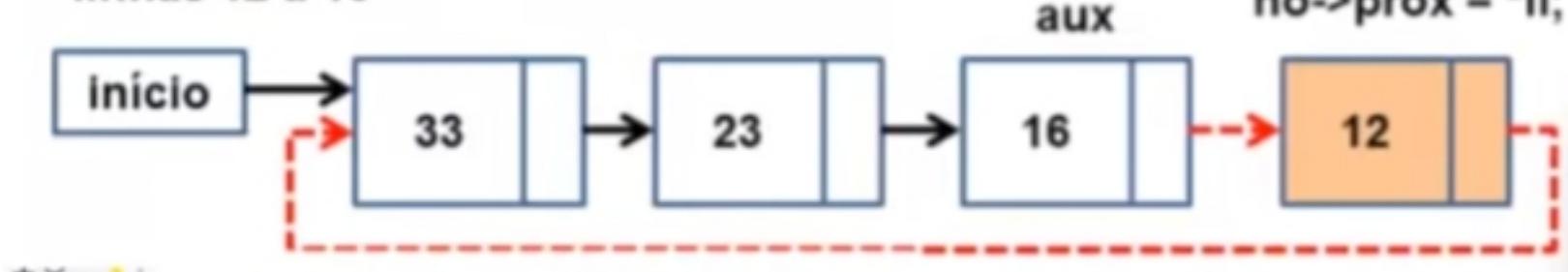
3.5. Dinâmica Encadeada Circular

- Implementando a Função “`insere_lista_final`”;



Busca último elemento “aux”:
linhas 12 a 15

Insere depois de “aux”:
`aux->prox = no;`
`no->prox = *li;`



3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função “insere_lista_ordenada”;

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float nl,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
//*****
10 int insere_lista_ordenada(Lista* li, struct aluno al);
11
12
```

```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5
6 int main(){
7     Lista *li;
8     struct aluno dados_aluno;
9     li = cria_lista();
10    x = lista_cheia(li);
11    x = lista_vazia(li);
12    x = insere_lista_ordenada(li, dados_aluno);
13    int x = tamanho_lista(li);
14    libera_lista(li);
15 }
```

```
12 int insere_lista_ordenada(Lista* li, struct aluno al){
13     if(li == NULL)
14         return 0;
15     Elemt *no = (Elemt*) malloc(sizeof(Elemt));
16     if(no == NULL)
17         return 0;
18     no->dados = al;
19     if((*li) == NULL){//insere inicio
20         *li = no;
21         no->prox = no;
22         return 1;
23     }
24     else{
25         if((*li)->dados.matricula > al.matricula){//insere inicio
26             Elemt *atual = *li;
27             while(atual->prox != (*li))//procura o ultimo
28                 atual = atual->prox;
29             no->prox = *li;
30             atual->prox = no;
31             *li = no;
32             return 1;
33         }
34         Elemt *ant = *li, *atual = (*li)->prox;
35         while(atual != (*li) && atual->dados.matricula < al.matricula){
36             ant = atual;
37             atual = atual->prox;
38         }
39         ant->prox = no;
40         no->prox = atual;
41         return 1;
42     }
43 }
```

3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função “imprime_lista”;

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float n1,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
//*****  
10 void imprime_lista(Lista* li);  

11
12 //Arquivo main.c
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include "ListaDinEncadCirc.h"
16
17 int main(){
18     Lista *li;
19     struct aluno dados_aluno;
20     li = cria_lista();
21     x = lista_cheia(li);
22     x = lista_vazia(li);
23     x = insere_lista_ordenada(li, dados_aluno);
24     imprime_lista(li); ←
25     int x = tamanho_lista(li);
26     libera_lista(li);
27 }
```

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protótipos
5
6 //Definição do tipo lista
7 struct elemento{
8     struct aluno dados;
9     struct elemento *prox;
10 };
11
12 typedef struct elemento Elem;
//*****  
13
14 void imprime_lista(Lista* li){
15     if(li == NULL || (*li) == NULL)
16         return;
17     Elem* no = *li;
18     do{
19         printf("Matricula: %d\n",no->dados.matricula);
20         printf("Nome: %s\n",no->dados.nome);
21         printf("Notas: %f %f %f\n",no->dados.n1,
22                                         no->dados.n2,
23                                         no->dados.n3);
24         printf("-----\n");
25         no = no->prox;
26     }while(no != (*li));
27 }
```

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”;

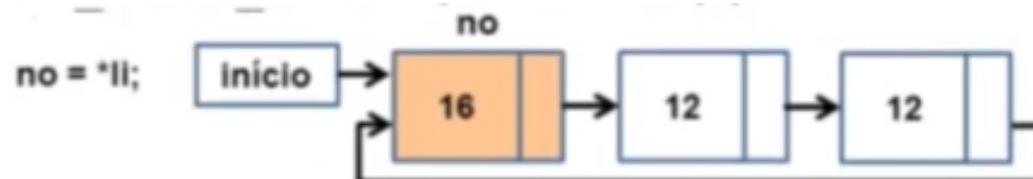
```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float nl,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
//*****
10 int remove_lista_inicio(Lista* li);
11
12 //Arquivo main.c
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include "ListaDinEncadCirc.h"
16 int main(){
17     Lista *li;
18     struct aluno dados_aluno;
19     li = cria_lista();
20     x = lista_cheia(li);
21     x = lista_vazia(li);
22     x = insere_lista_ordenada(li, dados_aluno);
23     imprime_lista(li);
24     int x = tamanho_lista(li);
25     remove_lista_inicio(li); ←
26     libera_lista(li);
27 }
```

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protótipos
5 //Definição do tipo lista
6 struct elemento{
7     struct aluno dados;
8     struct elemento *prox;
9 };
10 typedef struct elemento Elem;
//*****
11 int remove_lista_inicio(Lista* li){ ←
12     if(li == NULL)
13         return 0;
14     if((*li) == NULL)//lista vazia
15         return 0;
16
17     if((*li) == (*li)->prox) //lista fica vazia
18         free(*li);
19         *li = NULL;
20         return 1;
21     }
22     Elem *atual = *li;
23     while(atual->prox != (*li))//procura o último
24         atual = atual->prox;
25
26     Elem *no = *li;
27     atual->prox = no->prox;
28     *li = no->prox;
29     free(no);
30
31 }
```

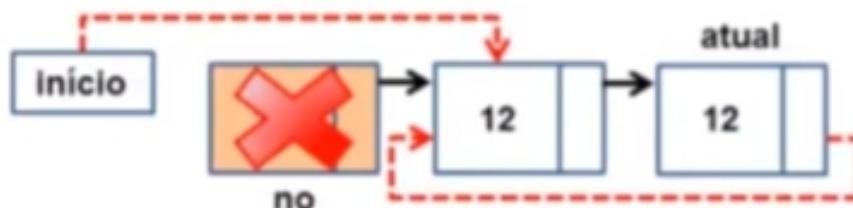
3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função “remove_lista_inicio”;

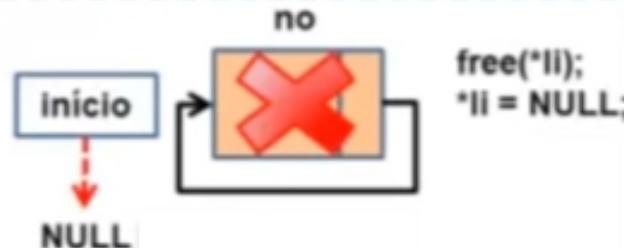


Busca último elemento “atual”



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

Se “no” é o único elemento da lista, a lista fica vazia:



```
free(*li);
*li = NULL;
```

3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função “remove_lista_final”;

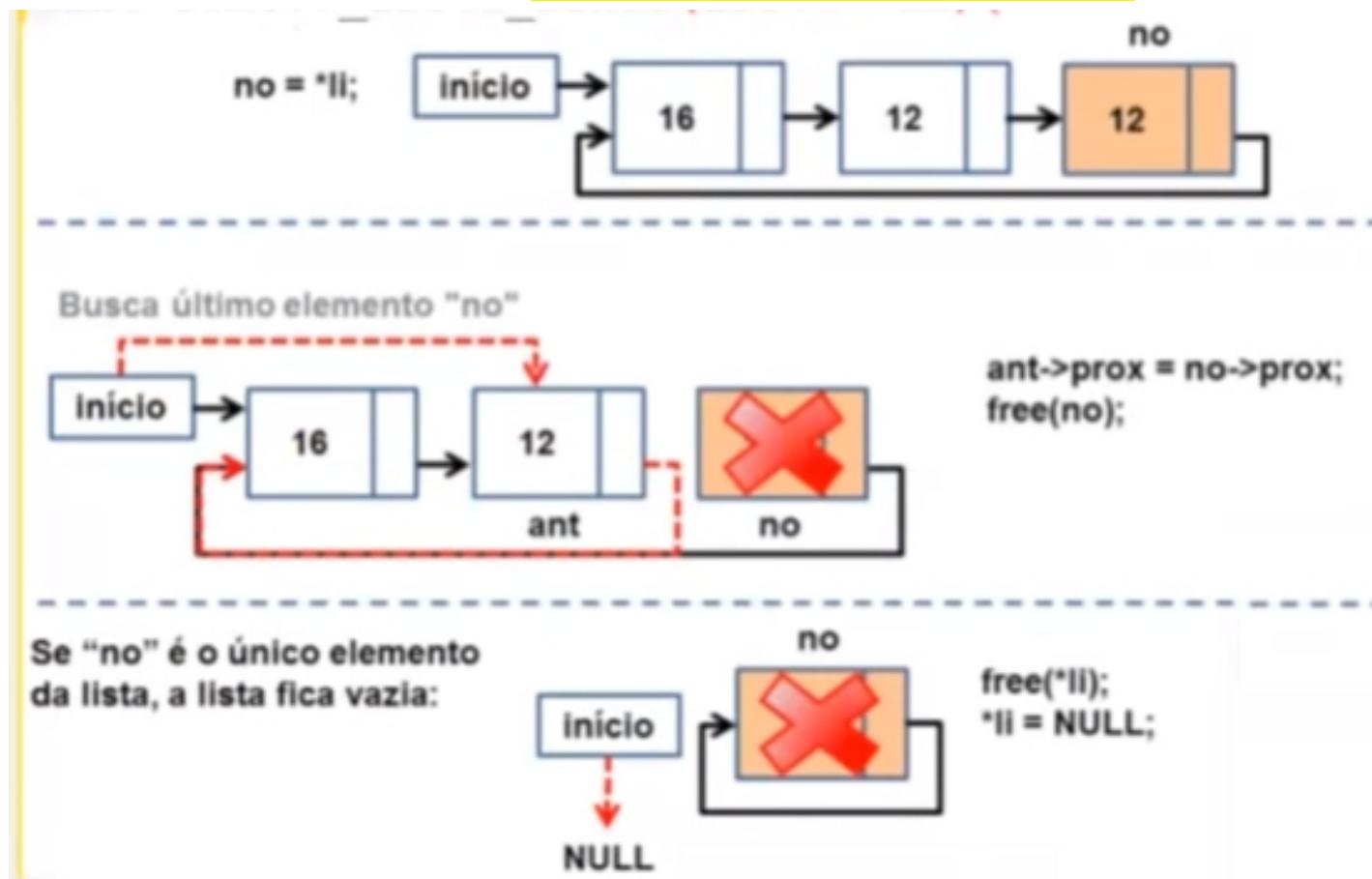
```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float nl,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
10 //*****
11 int remove_lista_final(Lista* li);
12
13 //Arquivo main.c
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include "ListaDinEncadCirc.h"
17
18 int main(){
19     Lista *li;
20     struct aluno dados_aluno;
21     li = cria_lista();
22     x = lista_cheia(li);
23     x = lista_vazia(li);
24     x = insere_lista_ordenada(li, dados_aluno);
25     imprime_lista(li);
26     int x = tamanho_lista(li);
27     remove_lista_final(li); ←
28     libera_lista(li);
29 }
```

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protó
5 //Definição do tipo lista
6 struct elemento{
7     struct aluno dados;
8     struct elemento *prox;
9 };
10 typedef struct elemento Elema;
11 //*****
12 int remove_lista_final(Lista* li){
13     if(li == NULL)
14         return 0;
15     if((*li) == NULL)//lista vazia
16         return 0;
17
18     if((*li) == (*li)->prox){//lista fica vazia
19         free(*li);
20         *li = NULL;
21         return 1;
22     }
23     Elema *ant, *no = *li;
24     while(no->prox != (*li)){//procura o último
25         ant = no;
26         no = no->prox;
27     }
28     ant->prox = no->prox;
29     free(no);
30     return 1;
31 }
```

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.

```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5 int main(){
6     Lista *li;
7     struct aluno dados_aluno;
8     int matricula_aluno; ←
9     li = cria_lista();
10    x = lista_cheia(li);
11    x = lista_vazia(li);
12    x = insere_lista_ordenada(li, dados_aluno);
13    imprime_lista(li);
14    int x = tamanho_lista(li);
15    remove_lista(li, matricula_aluno); ←
16    libera_lista(li);
17 }
```

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float nl,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
10 //*****
11 int remove_lista(Lista* li, int mat); ←
12
```

3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função
“remove_lista”.

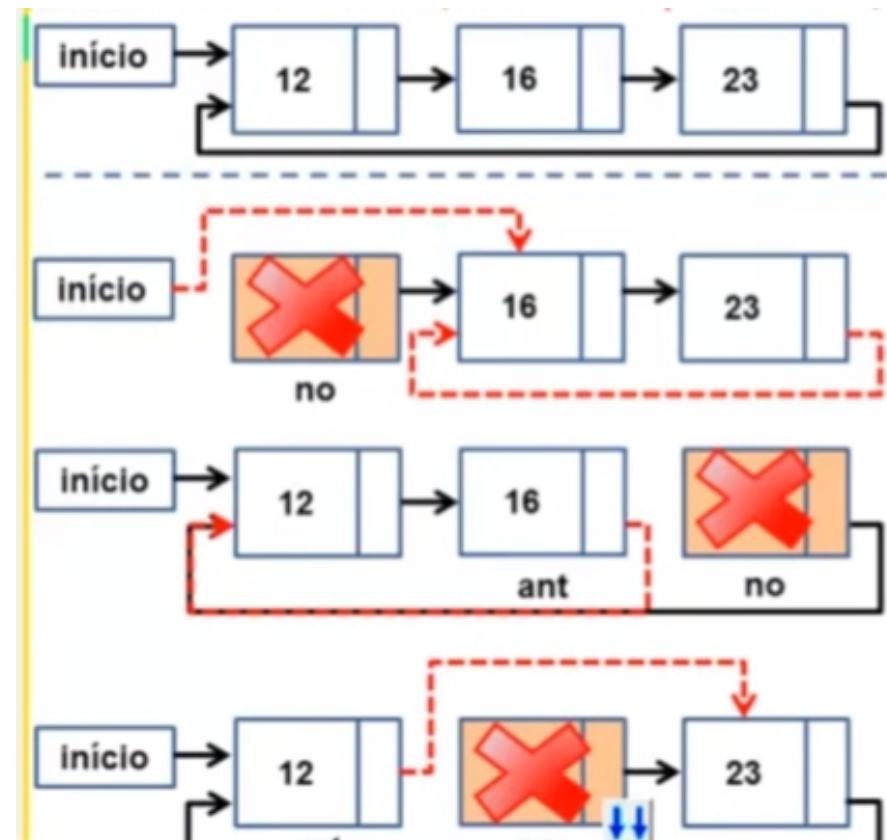
Remove um elemento
específico da lista.

```
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 inicio  
        if(no == no->prox){//lista fica vazia  
            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.

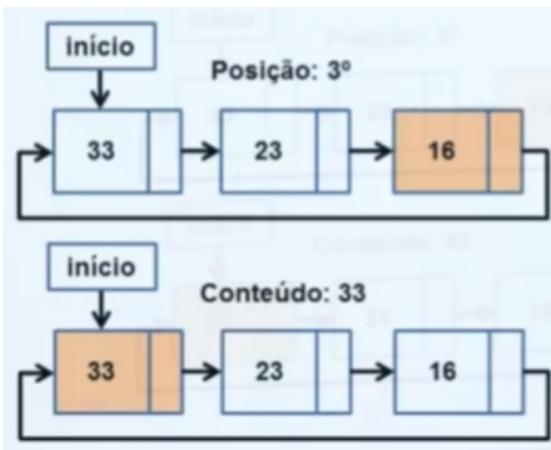


3. Lista

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”;

```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5
6 int main(){
7     Lista *li;
8     struct aluno dados_aluno;
9     int matricula_aluno, posicao;
10    li = cria_lista();
11    x = lista_cheia(li);
12    x = lista_vazia(li);
13    x = insere_lista_ordenada(li, dados_aluno);
14    imprime_lista(li);
15    int x = tamanho_lista(li);
16    x = consulta_lista_pos(li, posicao, &dados_aluno);
17    remove_lista(li, matricula_aluno);
18    libera_lista(li);
19 }
```

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float n1,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
10 //*****
11 int consulta_lista_pos(Lista* li, int pos, struct aluno *al);
```

3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função
“consulta_lista_pos”;

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protótipos
5 //Definição do tipo lista
6 struct elemento{
7     struct aluno dados;
8     struct elemento *prox;
9 };
10 typedef struct elemento Elem;
11 //*****
12 int consulta_lista_pos(Lista* li, int pos, struct aluno *al){
13     if(li == NULL || (*li) == NULL || pos <= 0)
14         return 0;
15     Elem *no = *li;
16     int i = 1;
17     while(no->prox != (*li) && i < pos){
18         no = no->prox;
19         i++;
20     }
21     if(i != pos)
22         return 0;
23     else{
24         *al = no->dados;
25         return 1;
26     }
27 }
```

3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função “consulta_lista_mat”;

```
1 //Arquivo main.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h"
5
6 int main(){
7     Lista *li;
8     struct aluno dados_aluno;
9     int matricula_aluno, posicao;
10    li = cria_lista();
11    x = lista_cheia(li);
12    x = lista_vazia(li);
13    x = insere_lista_ordenada(li, dados_aluno);
14    imprime_lista(li);
15    int x = tamanho_lista(li);
16    x = consulta_lista_mat(li, matricula_aluno, &dados_aluno);
17    remove_lista(li, matricula_aluno);
18    libera_lista(li);
```

```
1 //Arquivo ListaDinEncadCirc.h
2
3 struct aluno{
4     int matricula;
5     char nome[30];
6     float n1,n2,n3;
7 };
8
9 typedef struct elemento* Lista;
10 //*****
11 int consulta_lista_mat(Lista* li, int mat, struct aluno *al);
12
```

3. Lista

3.5. Dinâmica Encadeada Circular

- Implementando a Função “consulta_lista_mat”;

```
1 //Arquivo ListaDinEncadCirc.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ListaDinEncadCirc.h" //inclui os Protótipos
5 //Definição do tipo lista
6 struct elemento{
7     struct aluno dados;
8     struct elemento *prox;
9 };
10 typedef struct elemento Elem;
11 //*****int consulta_lista_mat(Lista* li, int mat, struct aluno *al){
12 int consulta_lista_mat(Lista* li, int mat, struct aluno *al){
13     if(li == NULL || (*li) == NULL)
14         return 0;
15     Elem *no = *li;
16     while(no->prox != (*li) && no->dados.matricula != mat)
17         no = no->prox;
18     if(no->dados.matricula != mat)
19         return 0;
20     else{
21         *al = no->dados;
22         return 1;
23     }
24 }
```

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 Iago. 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!