Estrutura de Dados

Aula 7 : Listas Circulares

Prof. MSc. Fausto Sampaio

https://github.com/Fausto14/estrutura_de_dados

Centro Universitário UniFanor - Wyden

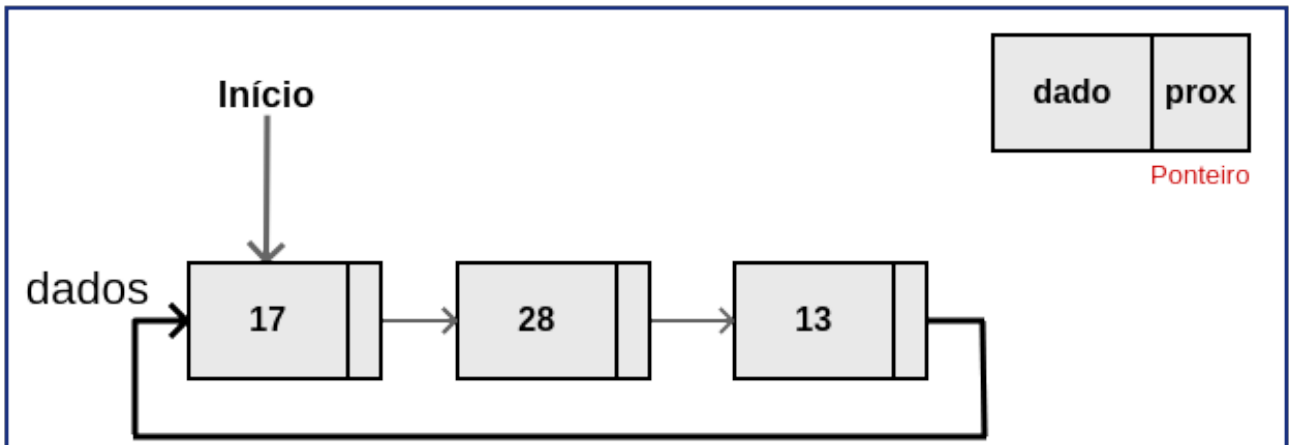
19 de novembro de 2019

Sumário

Lista Dinâmica Circular

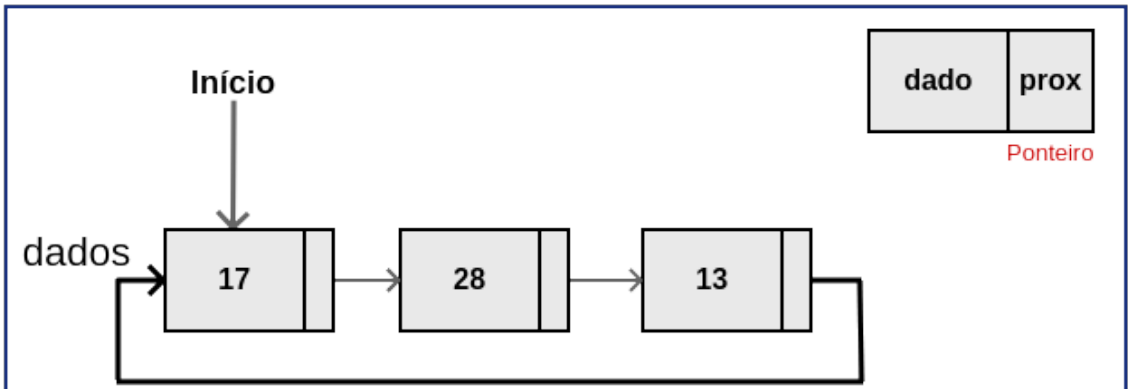
Definição

- **Lista Dinâmica Circular:** Tipo de lista onde cada elemento aponta para o seu sucessor (prox) 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 de lista;



Definição

- 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 **conteudo** do **ponteiro para ponteiro**.



Vantagens

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

Desvantagens

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

Quando utilizar

- Não há ncessidade de garantir um espaço mínimo para a execução do aplicativo;
- Inserção/Remoção em lista ordenada são as operações mais frequentes;
- Quando existe a necessidade de voltar ao primeiro elemento da lista depois de pecorrê-la.

Implementação em C

ListaDinEncadCirc.h

- os protótipos das funções;
- o tipo de dado armazenado na lista;
- o ponteiro "lista".

ListaDinEncadCirc.c

- o tipo de dado "lista";
- implementar as suas funções.

Protótipos

ListaDinEncadCirc.h

```
 3  struct aluno{
 4      int matricula;
 5      char nome[30];
 6      float n1,n2,n3;
 7  };
 8
 9  typedef struct elemento* Lista;
10
11  Lista* cria_lista();
12  void libera_lista(Lista* li);
13  int consulta_lista_pos(Lista* li, int pos, struct aluno *al);
14  int consulta_lista_mat(Lista* li, int mat, struct aluno *al);
15  int insere_lista_final(Lista* li, struct aluno al);
16  int insere_lista_inicio(Lista* li, struct aluno al);
17  int insere_lista_ordenada(Lista* li, struct aluno al);
18  int remove_lista(Lista* li, int mat);
19  int remove_lista_inicio(Lista* li);
20  int remove_lista_final(Lista* li);
21  int tamanho_lista(Lista* li);
22  int lista_vazia(Lista* li);
23  void imprime_lista(Lista* li);
```

Definição

ListaDinEncadCirc.c

```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;
```

Criar Lista

Procedimentos

- Alocar memória para um ponteiro tipo Lista;
- Fazer o ponteiro que indica o início da lista, apontar para NULL.

ListaDinEncadCirc.c

```
12  Lista* cria_lista(){
13      Lista* li = (Lista*) malloc(sizeof(Lista));
14      if(li != NULL)
15          *li = NULL;
16      return li;
17  }
```

Liberar Lista

Procedimentos

- Percorrer toda a lista até voltar novamente ao início da lista;
- Liberar memória referente a cada elemento da lista;
- Liberar memória referente à estrutura geral da lista;

ListaDinEncadCirc.c

```
19 void libera_lista(Lista* li){
20     if(li != NULL && (*li) != NULL){
21         Elem *aux, *no = *li;
22         while((*li) != no->prox){
23             aux = no;
24             no = no->prox;
25             free(aux);
26         }
27         free(no);
28         free(li);
29     }
30 }
```

Tamanho da Lista

Procedimentos

- Percorrer toda a lista enquanto o elemento corrente for diferente do início da lista;
- Para cada elemento acessado incrementar o contador em 1;
- retornar o valor do contador;

ListaDinEncadCirc.c

```
217 int tamanho_lista(Lista* li){
218     if(li == NULL || (*li) == NULL)
219         return 0;
220     int cont = 0;
221     Elem* no = *li;
222     do{
223         cont++;
224         no = no->prox;
225     }while(no != (*li));
226     return cont;
227 }
```

Lista Cheia

Procedimentos

- Retornar false (0);

Observação

- Na implementação dinâmica, a lista nunca será cheia, pois o limite da lista é a quantidade de memória do computador;

ListaDinEncadCirc.c

```
211 int lista_cheia(Lista* li){
212     return 0;
213 }
```

Lista Vazia

Procedimentos

- verdadeiro: se a estrutura da lista for NULL ou se o ponteiro de início apontar para NULL;
- caso contrário: falso;

ListaDinEncadCirc.c

```
215 int lista_vazia(Lista* li){
216     if(li == NULL)
217         return 1;
218     if(*li == NULL)
219         return 1;
220     return 0;
221 }
```

Inserir - Início da Lista

Procedimentos

- Se a lista for vazia:
  - Fazer o ponteiro que indica o início da lista, apontar para o novo elemento;
  - Fazer o ponteiro **prox** do novo elemento apontar para si mesmo (circular).
- Se a lista NÃO for vazia:
  - Percorrer na lista até encontrar o último elemento;
  - Fazer o ponteiro **prox** do último elemento apontar para o novo elemento;
  - Fazer o ponteiro **prox** do novo elemento apontar para o início da lista;
  - Fazer o ponteiro que indica o início da lista, apontar para o novo elemento;

Inserir - Início da Lista

ListaDinEncadCirc.c

```
84  int insere_lista_inicio(Lista* li, struct aluno al){
85      if(li == NULL)
86          return 0;
87      Elem *no = (Elem*) malloc(sizeof(Elem));
88      if(no == NULL)
89          return 0;
90      no->dados = al;
91      if((*li) == NULL){//lista vazia: insere início
92          *li = no;
93          no->prox = no;
94      }else{
95          Elem *aux = *li;
96          while(aux->prox != (*li)){
97              aux = aux->prox;
98          }
99          aux->prox = no;
100         no->prox = *li;
101         *li = no;
102     }
103     return 1;
104 }
```

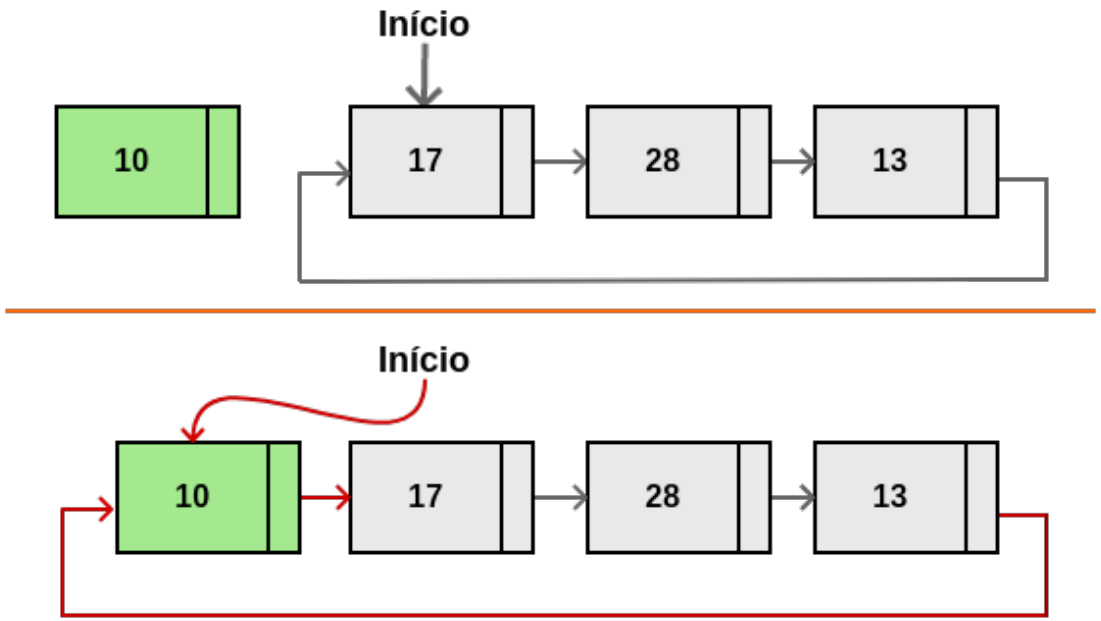
Inserir - Início da Lista

Lista vazia

- Como é feita a inserção em uma lista circular vazia?

Atenção

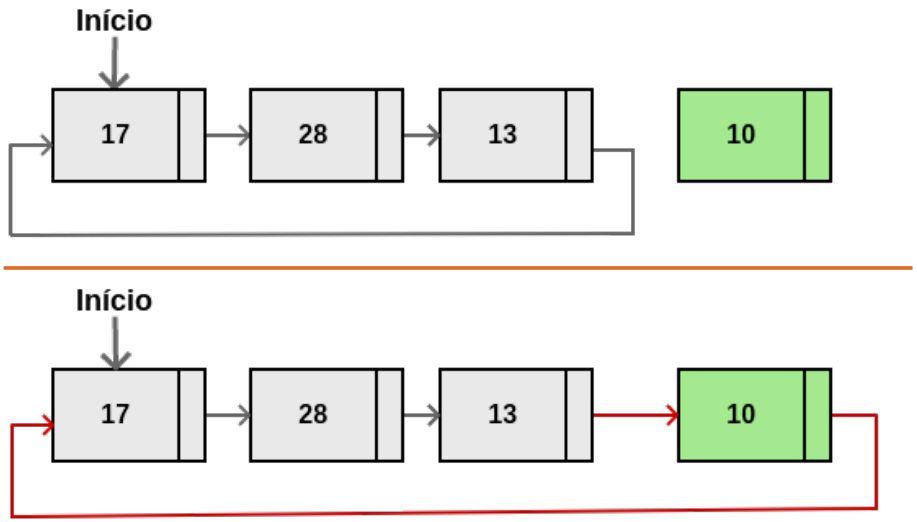
- Fazer o ponteiro **prox** do último elemento, apontar para o novo elemento;



Inserir - Final da Lista

Procedimentos

- Se a lista for vazia:
  - Fazer o início da lista, apontar para o novo elemento;
  - Fazer o **prox** do novo elemento apontar para si mesmo (circular).
- Se a lista NÃO for vazia:
  - Percorrer na lista até encontrar o último elemento;
  - Fazer o **prox** do último elemento apontar para o novo elemento;
  - Fazer o **prox** do novo elemento apontar para o início da lista;



Inserir - Final da Lista

ListaDinEncadCirc.c

```
63 int insere_lista_final(Lista* li, struct aluno al){
64     if(li == NULL)
65         return 0;
66     Elem *no = (Elem*) malloc(sizeof(Elem));
67     if(no == NULL)
68         return 0;
69     no->dados = al;
70     if((*li) == NULL){//lista vazia: insere início
71         *li = no;
72         no->prox = no;
73     }else{
74         Elem *aux = *li;
75         while(aux->prox != (*li)){
76             aux = aux->prox;
77         }
78         aux->prox = no;
79         no->prox = *li;
80     }
81     return 1;
82 }
```

Remover - Início da Lista

Procedimentos

- Se o elemento a ser removido for o único da lista: liberar o elemento do início (free), depois fazer o início da lista apontar para NULL;
- Se o elemento a ser removido NÃO for o único da lista: procurar o último elemento da lista;
- Fazer o **prox** do último elemento apontar para o **prox** do inicio da lista;
- Fazer o inicio da lista apontar para o **prox** do inicio da lista;
- Liberar memória do elemento que está sendo removido: free();

Cuidado

- Não se pode remover de uma lista vazia;
- Removendo o último nó, a lista fica vazia;

Remover - Inicio da Lista

ListaDinEncadCirc.c

```
139 int remove_lista_inicio(Lista* li){
140     if(li == NULL)
141         return 0;
142     if((*li) == NULL)//lista vazia
143         return 0;
144 
145     if((*li) == (*li)->prox){//lista fica vaza
146         free(*li);
147         *li = NULL;
148         return 1;
149     }
150     Elem *atual = *li;
151     while(atual->prox != (*li))//procura o último
152         atual = atual->prox;
153 
154     Elem *no = *li;
155     atual->prox = no->prox;
156     *li = no->prox;
157     free(no);
158     return 1;
159 }
```

Remover - Fim da Lista

Questão

- Quais os procedimentos para remover um elemento no fim de uma Lista Dinâmica Circular?

Imprimir Lista

Procedimentos

- Se a lista NÃO for vazia:
  - A partir do início, percorrer toda a lista enquanto o **prox** do elemento corrente for diferente do início da lista;
  - Para cada elemento acessado, imprimir seus dados.

```
241 void imprime_lista(Lista* li){
242     if(li == NULL || (*li) == NULL)
243         return;
244     Elem* no = *li;
245     do{
246         printf("Matricula: %d\n",no->dados.matricula);
247         printf("Nome: %s\n",no->dados.nome);
248         printf("Notas: %f %f %f\n",no->dados.n1,
249                                    no->dados.n2,
250                                    no->dados.n3);
251         printf("-------------------------------\n");
252         no = no->prox;
253     }while(no != (*li));
254 }
```

Referências

Referências

- André Ricardo Backes, CAPÍTULO 5 - Listas, Editor(s): André Ricardo Backes, **Estrutura de Dados Descomplicada em Linguagem C**, Elsevier Editora Ltda., 2016, Pages 77-191, ISBN 9788535285239.