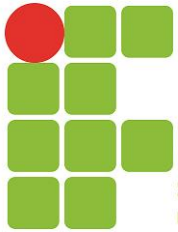


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Presidente Epitácio

Estruturas de Dados – ED1C3

Listas Ordenadas e Ponteiro para Ponteiro

PROF. MARCELO ROBERTO ZORZAN

DISCIPLINA: ESTRUTURAS DE DADOS I

AULA 09

Aula de Hoje

Lista encadeada ordenada

Ponteiro para Ponteiro

Lista Encadeada Ordenada

- Inserção Ordenada

- Insere um nó mantendo a lista ordenada

- Se a lista estiver vazia

→ Faz a lista apontar para o novo nó

- Senão encontra o primeiro nó maior ou igual ao novo nó

→ Faz o novo nó apontar para o nó atual

→ Faz o nó anterior apontar para o novo nó

Lista Ordenada - Implementação

```
CELULA* inserirListaOrdem(CELULA* lista, int x){
  CELULA* atual = lista;
  CELULA* anterior = NULL;
  CELULA* q;

  q = getnode();
  if(q != NULL)
  {
      q->info = x;
      q->next = NULL;

      /*procura posicao para insercao*/
      while(atual != NULL && atual->info < x)
      {
          anterior = atual;
          atual = atual->next;
      }
    ...
```

Insere nó mantendo a lista ordenada

Lista Ordenada - Implementação

```
        /*insere elemento*/
        if(atual == lista)  /*insere no começo*/
        {
            q->next = lista;
            lista = q;
        }
        else  /*insere no meio/fim*/
        {
            anterior->next = q;
            q->next = atual;
        }
        return lista;
    }// Fim do if(q != NULL)
    else {
        printf ("\nERRO na alocação do nó.\n");
        return NULL;
    }
}
```

Lista Ordenada - Implementação

```
int main(){
    CELULA *ptrlista;
    ptrlista = init(ptrlista);
    ptrlista = inserirListaOrdem(ptrlista, 7);
    exibe_lista(ptrlista);
    ptrlista = inserirListaOrdem(ptrlista, 3);
    exibe_lista(ptrlista);
    ptrlista = inserirListaOrdem(ptrlista, 5);
    exibe_lista(ptrlista);

    getch();
    return 0;
}
```

Ponteiro para Ponteiro

Um ponteiro para um ponteiro é como se você deixasse dentro do seu armário um papel com o endereço da casa de um amigo.

Notação:

```
tipo_da_variável **nome_da_variável;
```

onde

****nome_da_variável** é o conteúdo final da variável apontada

***nome_da_variável** é o conteúdo do ponteiro intermediário.

Ponteiro para Ponteiro

Exemplo:

```
int main()
{
    int a = 10;
    int *Ptra;
    int **ptrPtra;
    Ptra = &a;
    ptrPtra = &Ptra;
    printf("Valor de ptrPtra: %d", **ptrPtra);
    getch();
    return 0;
}
```

Lista Encadeada - Implementação

- Por que o tipo de retorno da função init tem que ser CELULA*?

```
CELULA* init (CELULA *lista)
{
    lista = NULL;
    return lista;
}
```

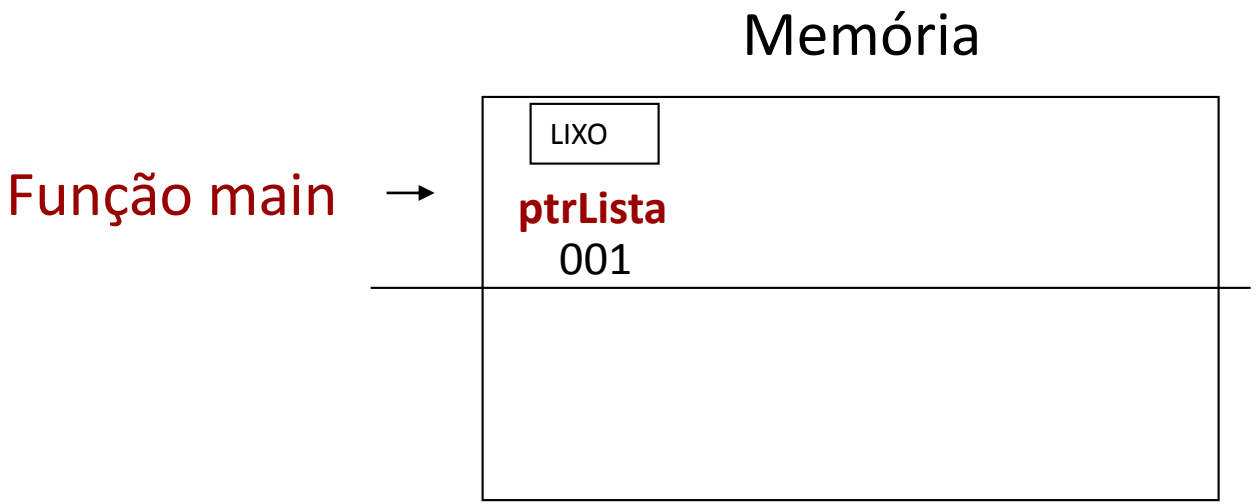
```
int main()
{
    CELULA *ptrLista;
    ptrLista = init(ptrlista);
}
```

Lista Encadeada - Implementação

- Por que o tipo de retorno da função init tem que ser CELULA*?

```
int main()
{
   CELULA *ptrLista;

}
```



Lista Encadeada - Implementação

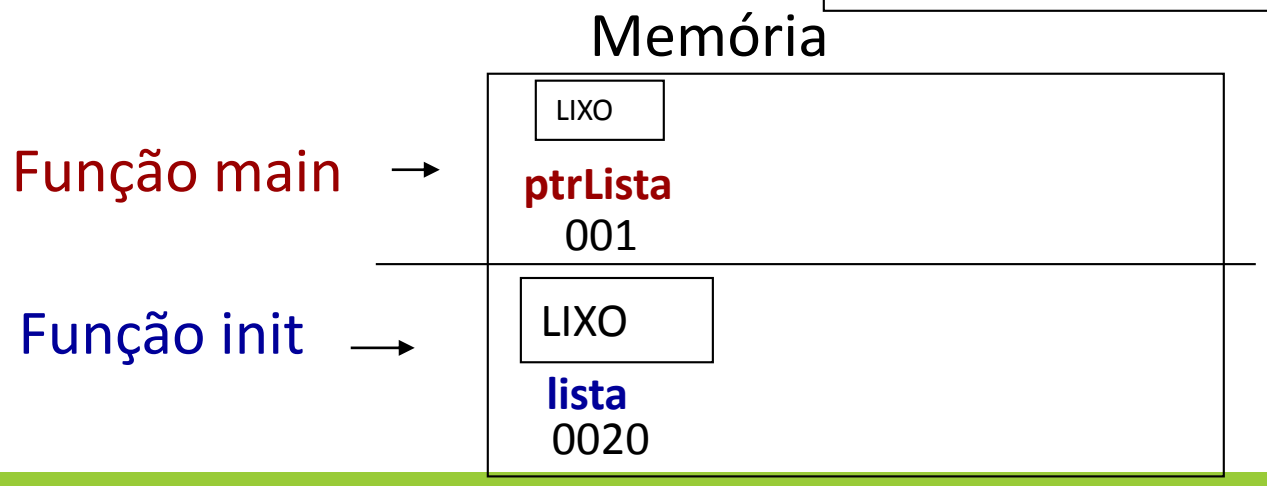
- Por que o tipo de retorno da função init tem que ser CELULA*?

```
int main()
{
   CELULA *ptrLista;
   ptrLista = init(ptrlista);
}
```

```
CELULA* init (CELULA *lista)
{



}
```



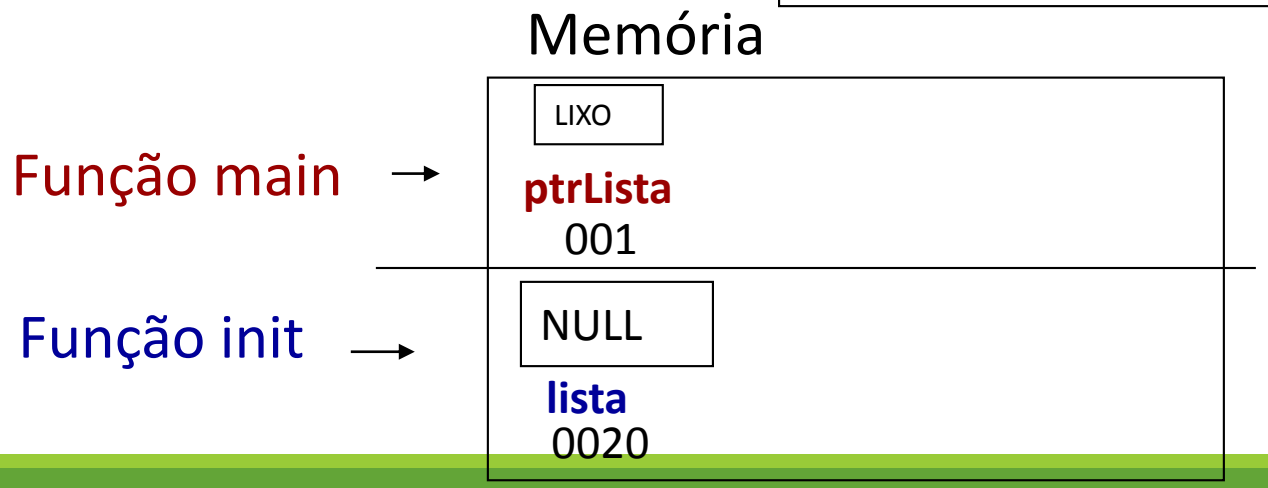
Lista Encadeada - Implementação

- Por que o tipo de retorno da função init tem que ser CELULA*?

```
int main()
{
   CELULA *ptrLista;
   ptrLista = init(ptrlista);
}
```

```
CELULA* init (CELULA *lista)
{
    lista = NULL;

}
```

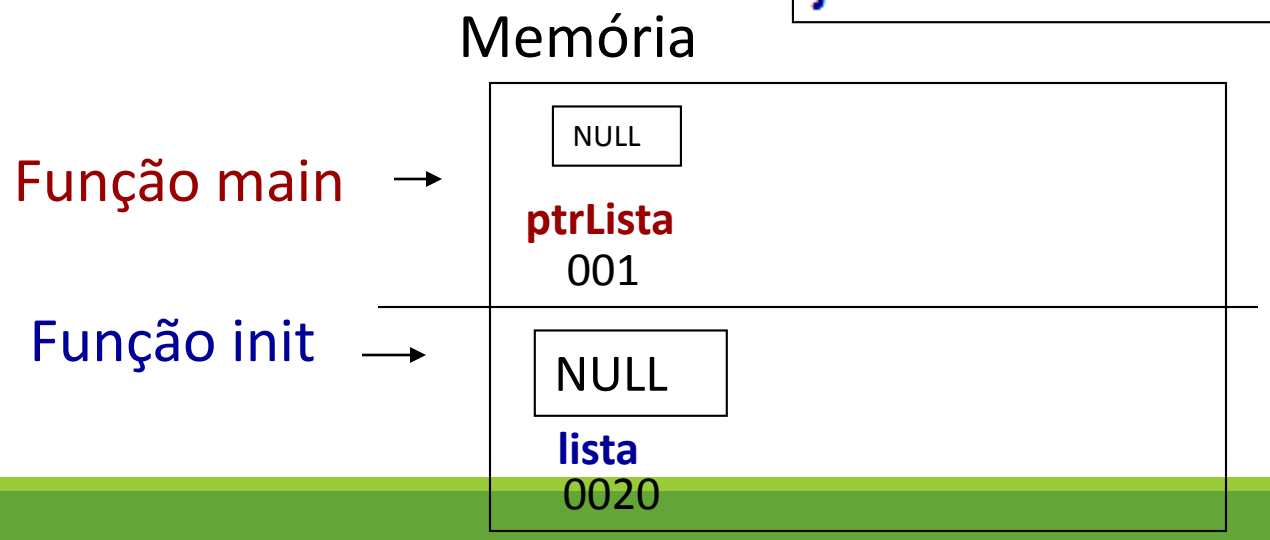


Lista Encadeada - Implementação

- Por que o tipo de retorno da função init tem que ser CELULA*?

```
int main()
{
   CELULA *ptrLista;
   ptrLista = init(ptrlista);
}
```

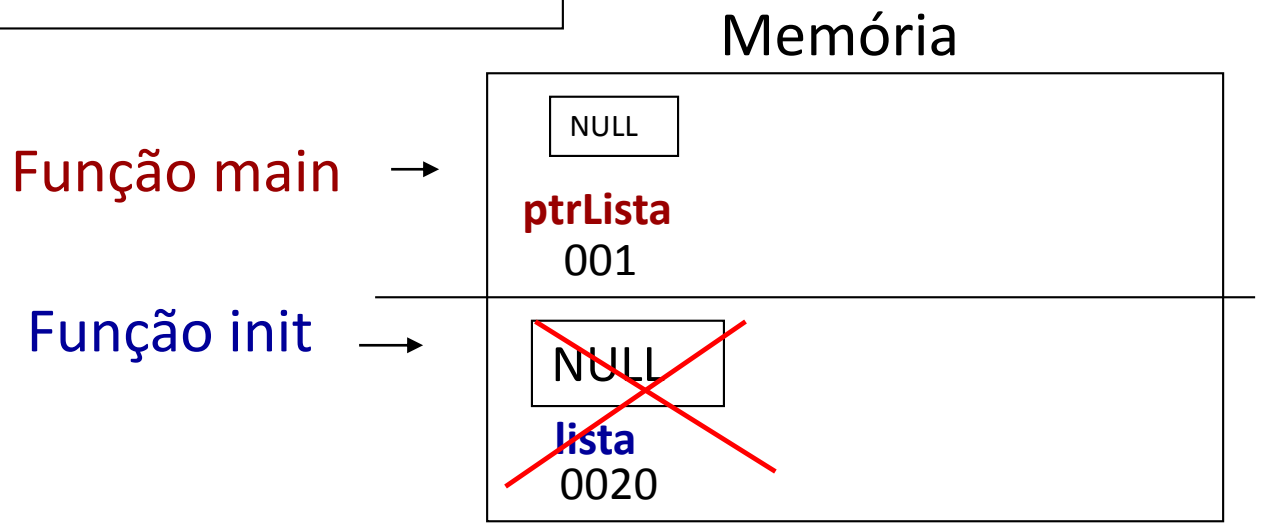
```
CELULA* init (CELULA *lista)
{
    lista = NULL;
    return lista;
}
```



Lista Encadeada - Implementação

- Por que o tipo de retorno da função init tem que ser CELULA*?

```
int main()
{
   CELULA *ptrLista;
   ptrLista = init(ptrlista);
}
```



Lista Encadeada - Implementação

- E se usarmos ponteiro para ponteiro?

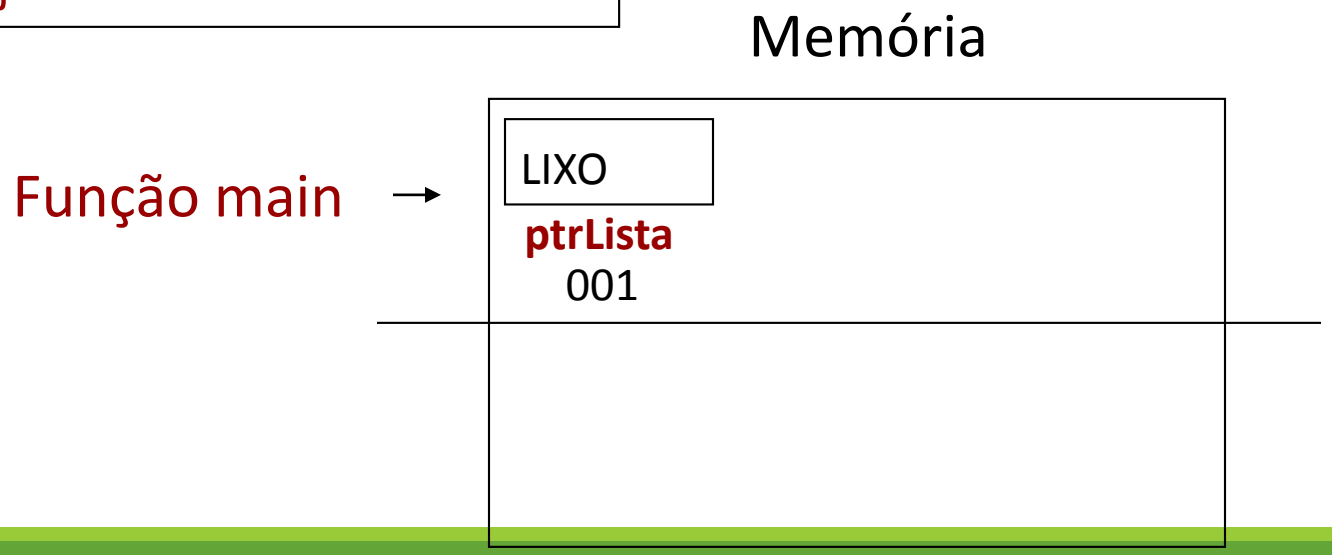
```
void init (CELULA **lista)
{
    *lista = NULL;
}
```

Lista Encadeada - Implementação

- E se usarmos ponteiro para ponteiro?

```
int main()
{
   CELULA *ptrLista;

}
```



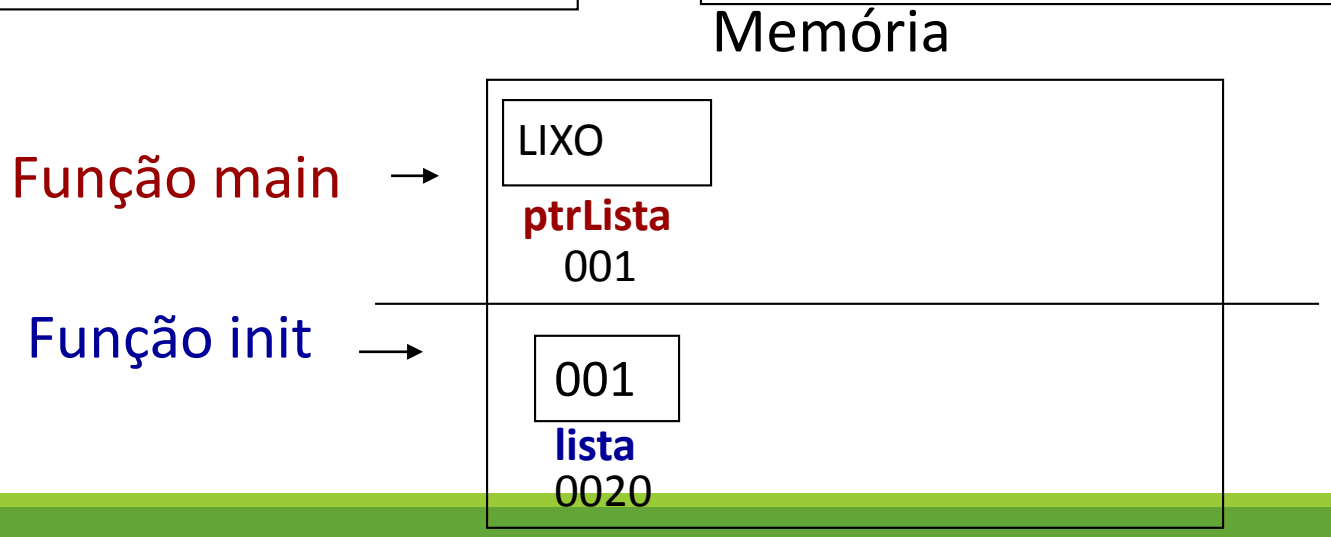
Lista Encadeada - Implementação

- E se usarmos ponteiro para ponteiro?

```
int main()
{
   CELULA *ptrLista;
   init(&ptrLista);
}
```

```
void init (CELULA **lista)
{


}
```

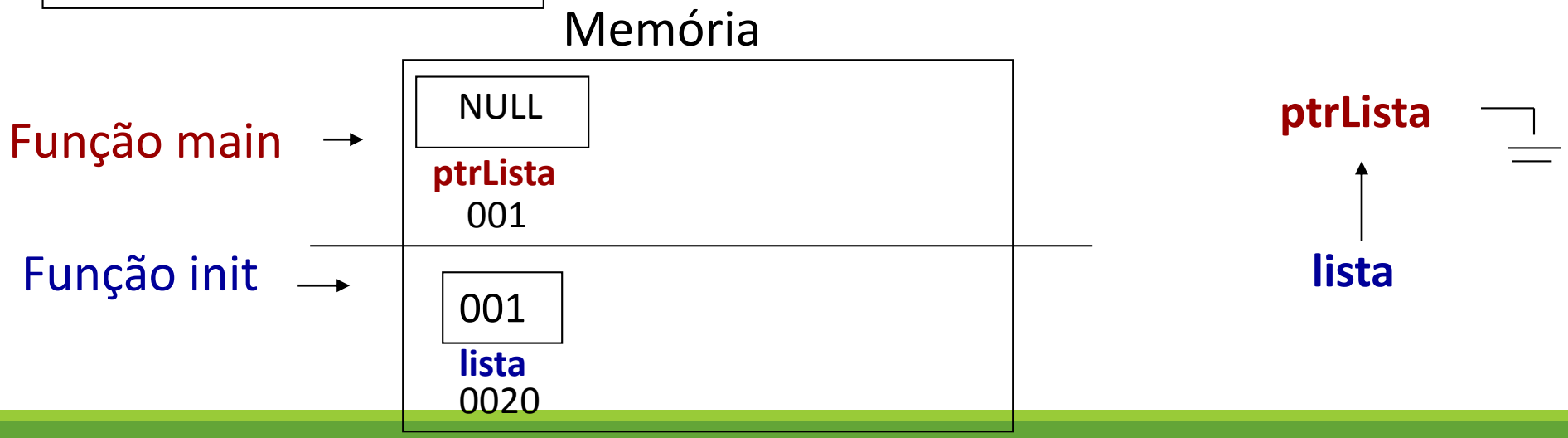


Lista Encadeada - Implementação

- E se usarmos ponteiro para ponteiro?

```
int main()
{
   CELULA *ptrLista;
   init(&ptrLista);
}
```

```
void init (CELULA **lista)
{
    *lista = NULL;
}
```



Lista Encadeada - Implementação

- Inicializar o ponteiro externo à lista encadeada linear com o valor NULL

```
void init (CELULA **lista)
{
    *lista = NULL;
}
```

→Inicia o ponteiro externo para uma lista encadeada

Lista Encadeada - Implementação

- Inserir novo nó no início da lista encadeada

```
void insere_inicio (CELULA **lista, int x) {
    CELULA *q;

    q = getnode ();
    if (q != NULL) {
        q->info = x;
        q->next = *lista;
        *lista = q;
    }
    else {
        printf ("\nERRO: falha na alocacao do noh.\n");
        exit(1);
    }
}
```

Lista Encadeada - Implementação

- Inserir novo nó no final da lista encadeada

```
void insere_fim (CELULA **lista, int x) {
    CELULA *q;
    CELULA *aux;

    q = getnode ();
    if (q != NULL) {
        q->info = x;
        q->next = NULL;

        if (empty(*lista))
            *lista = q;
```

Lista Encadeada - Implementação

- Inserir novo nó no final da lista encadeada (cont.)

```
        else{  //percorre lista até chegar ao ultimo nó
          aux = *lista;
          while (aux->next != NULL)
               aux = aux->next;

          aux->next = q;
      }
   }// Fim do if(q != NULL)
   else {
      printf ("\nERRO na alocação do nó.\n");
      exit(1);
    }
}
```

Lista Encadeada - Implementação

- Remover nó início da lista encadeada

```
void remove_inicio (CELULA **lista) {
    CELULA *q;

    q = *lista;
    if (!empty(*lista)) {  //há itens na lista
        *lista = q->next;
        freenode (q);
    }
    else {
        printf ("\nERRO: lista vazia.\n");
        exit(1);
    }
}
```

Lista Encadeada - Implementação

- Remover nó específico da lista encadeada

```
int remove_valor (CELULA **lista, int x) {
    CELULA *q;
    CELULA *aux;

    if ((q = pesquisa (*lista, x)) != NULL)
    {
        aux = *lista;
        if (aux == q)  //nó está no inicio da lista
            remove_inicio (lista);
```

Lista Encadeada - Implementação

- Remover nó da lista encadeada (cont.)

```
        else {
            while (aux->next != q)
                aux = aux->next;
            aux->next = q->next;
            freenode (q);
        }
        return 1; // removeu
    }
    return 0; //não removeu
}
```

Lista Encadeada - Implementação

```
int main(){
    CELULA *ptrlista;
    int del;
    init(&ptrlista);
    insere_inicio(&ptrlista, 7);
    exibe_lista(ptrlista);
    insere_fim(&ptrlista, 3);
    exibe_lista(ptrlista);
    del = remove_valor (&ptrlista, 3);
    if(del == 0)
       printf("\nErro!");
    exibe_lista(ptrlista);

    getch();
}
```

Leitura Recomendada

DROZDEK, Adam. Estrutura de Dados e Algoritmos em C++. Editora Pioneira Thomson Learning, 2005.

→ Pág 91, seção 3.5 (Listas Auto-Organizadas)- até pág. 94