INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAIANO

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Geovanni Martins de Souza

Jeovana Miranda Souza

Lívia Alkimim dos Santos

RELATÓRIO TÉCNICO SOBRE LISTAS DINÂMICAS

CAMPUS GUANAMBI

2025

Geovanni Martins de Souza
Jeovana Miranda Souza
Lívia Alkimim dos Santos

RELATÓRIO TÉCNICO SOBRE LISTAS DINÂMICAS

Trabalho apresentado como requisito parcial de avaliação do componente curricular Estrutura de Dados do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, 2 ° período, do Instituto Federal de Educação, Ciências e Tecnologia - Campus Guanambi.

Orientador: Prof. Reinaldo Monteiro Cotrim

CAMPUS GUANAMBI

2025

Ìndice de figuras

SUMÁRIO

1. Introdução ao Tema

Uma lista linear é uma sequência de n ≥ 0 nós X[1], X[2], ..., X[n], cujas propriedades estruturais essenciais envolvem apenas as posições relativas entre os itens, conforme aparecem em uma linha.Quando n = 0 a lista é chamada nula ou vazia. (KNUTH, 1997, p. 239, tradução nossa).

Conceitos Básicos:

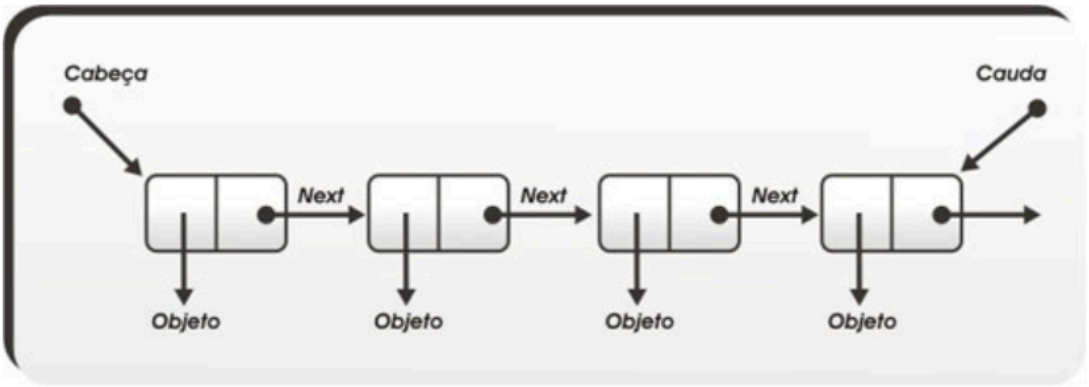
- **Nós (Nodes):** É a unidade básica de uma lista, responsável por armazenar um valor e, quando aplicável, um ponteiro para outro nó.
- **Ponteiro (Pointer):** É uma referência que indica a posição de outro elemento na memória, permitindo que os nós fiquem conectados.
- **Inserção (Insertion):** adiciona um novo nó em uma posição específica.
- **Remoção (Remove):**Faz a desalocação do nó existente.
- **Percurso (Traversal)**: Acessa sequencialmente todos os nós.
- **Busca (Search)**: Localiza nós com valores específicos.

Ela armazena uma sequência finita de elementos (geralmente do mesmo tipo) que é manipulada por operações fundamentais como: acessar elementos, inserir, excluir e percorrer a sequência é chamada de fila. São discutidas implementações concretas (listas estáticas em vetores/arrays e listas dinâmicas encadeadas), questões de organização de memória e análise do custo das operações (complexidade).

2. Explicação teórica

Lista Simplesmente Encadeada: Estrutura de dados linear em que cada nó contém dois componentes, sendo eles um campo de dado e um ponteiro/referência para o próximo nó na sequência.O primeiro nó é chamado de head; o último nó aponta para NULL (ou equivalente), indicando o fim da lista. A travessia (percurso) é possível somente em um sentido, do head até o nó final. Operações típicas: inserção, remoção, pesquisa, percurso.

Figura 1: Exemplo didático.



Fonte: Exemplo de funcionamento de uma lista encadeada STACK OVERFLOW EM PORTUGUÊS (2025).

A estrutura é simples, possuindo um campo do tipo inteiro chamado valor (poderia ser qualquer tipo, inclusive outro tipo mais complexo) e um ponteiro para um próximo nó.

Figura 2: Estrutura básica da lista simplesmente encadeada.

```
typedef struct No {
    int valor;
    struct No *proximo;
} No;
```

Fonte: Martins, Geovanni et al, outubro de 2025.

Na sequência, faremos a inserção de elementos na lista encadeada. A seguir, é apresentado o procedimento para inserir um novo elemento no início da lista.

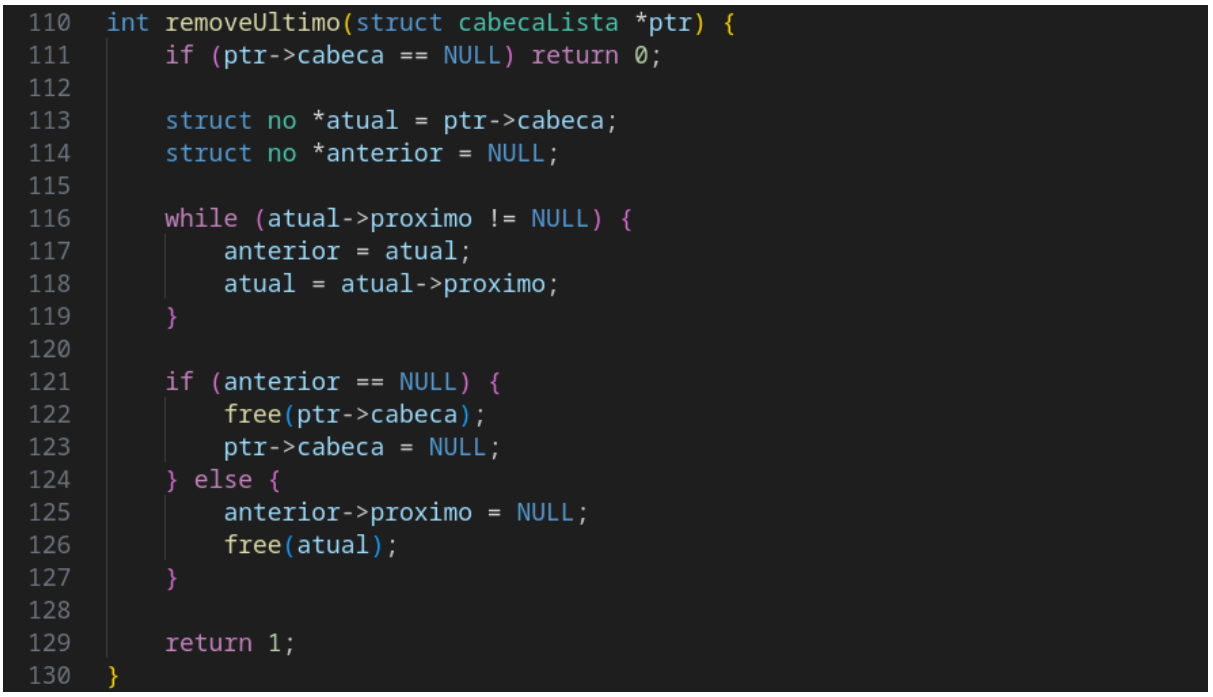
Figura 3: Inserção de elementos na lista.

```
70  void iniciarLista(struct cabecaLista *ptr) {
71      struct no *novo = malloc(sizeof *novo);
72      printf("Insira o valor inicial: ");
73      scanf("%d", &novo->valor);
74      novo->proximo = NULL;
75      ptr->cabeca = novo;
76  }
```

Fonte: Martins, Geovanni et al, outubro de 2025.

Após, o próximo passo é a remoção do último elemento da lista

Figura 4: Remoção do último elemento da lista.

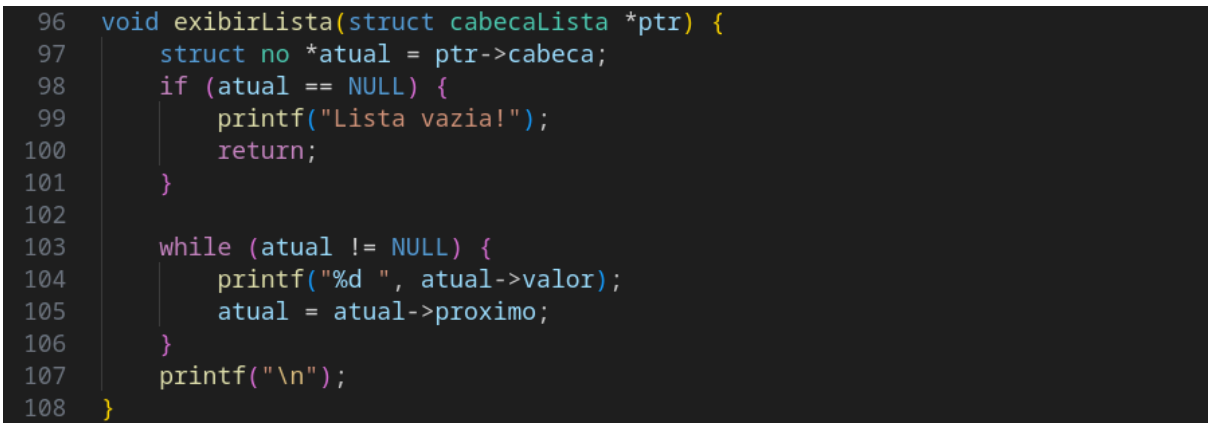


```
110 int removeUltimo(struct cabecaLista *ptr) {
111     if (ptr->cabeca == NULL) return 0;
112 
113     struct no *atual = ptr->cabeca;
114     struct no *anterior = NULL;
115 
116     while (atual->proximo != NULL) {
117         anterior = atual;
118         atual = atual->proximo;
119     }
120 
121     if (anterior == NULL) {
122         free(ptr->cabeca);
123         ptr->cabeca = NULL;
124     } else {
125         anterior->proximo = NULL;
126         free(atual);
127     }
128 
129     return 1;
130 }
```

Fonte: Martins, Geovanni et al, outubro de 2025.

Para finalizar, exibimos a lista na tela

Figura 5: Exibição da lista.

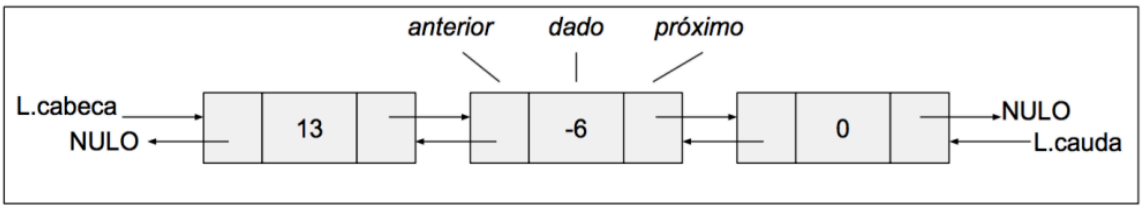


```
 96 void exibirLista(struct cabecaLista *ptr) {
 97     struct no *atual = ptr->cabeca;
 98     if (atual == NULL) {
 99         printf("Lista vazia!");
100         return;
101     }
102 
103     while (atual != NULL) {
104         printf("%d ", atual->valor);
105         atual = atual->proximo;
106     }
107     printf("\n");
108 }
```

Fonte: Martins, Geovanni et al, outubro de 2025.

Lista Duplamente Encadeada: A lista duplamente encadeada é formada por nós similares ao da lista encadeada simples, porém com um atributo adicional: um apontador para o nó anterior. Dessa forma, é possível percorrer esse tipo de lista em ambas as direções.

Figura 6: Exemplo didático de lista duplamente encadeada.



Fonte: Estrutura de uma lista duplamente encadeada GRÉGIO (2025).

O nó representado na figura acima pode ser criado da seguinte forma:

Figura 7: Estrutura básica da lista duplamente encadeada.

```
2.      Estrutura nó para a lista duplamente encadeada
3.  */
4.  typedef struct no{
5.      int valor;
6.      struct no *proximo;
7.      struct no *anterior;
8.  }No;
```

Fonte: Martins, Geovanni et al, outubro de 2025.

Como mencionado, na lista duplamente encadeada cada alteração na lista precisa atualizar os dois ponteiros de cada nó envolvido na operação. A seguir é apresentado um procedimento para inserir um novo nó no início da lista dupla. Pode-se analisar que os dois ponteiros são atualizados de acordo com a operação realizada, neste caso uma inserção no início.

Figura 8: Inserção de elemento na lista duplamente encadeada

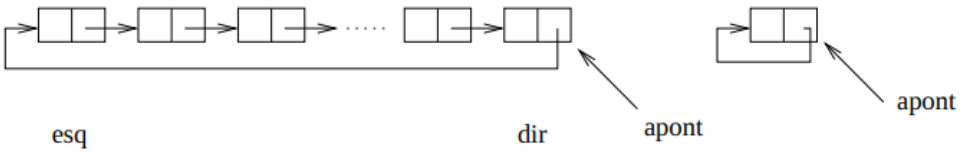
```c
/*
        procedimento para inserir um novo nó no inicio da lista
*/
void inserir_no_inicio(No **lista, int num){
    No *novo = malloc(sizeof(No));

    if(novo){
        novo->valor = num;
        // próximo do novo nó aponta para o inicio da lista
        novo->proximo = *lista;
        // o anterior é nulo pois é o primeiro nó
        novo->anterior = NULL;
        // se a lista não estiver vazia, o anterior do primeiro nó aponta para o novo nó
        if(*lista)
            (*lista)->anterior = novo;
        // o novo nó passa a ser o inicio da lista (o primeiro nó da lista)
        *lista = novo;
    }
    else
        printf("Erro ao alocar memoria!\n");
}
```

Fonte: Martins, Geovanni et al, outubro de 2025.

Lista Circular Simples: Uma lista simples tem a propriedade que o último nó aponta sempre para o primeiro nó. Então, de qualquer nó da lista pode-se atingir qualquer outro nó da lista.

Figura 9: Exemplo didático de Lista Circular



Fonte: Estrutura de uma lista circular GOLD (2001).

A principal complexidade da lista circular está no fato de que, em qualquer operação na lista, precisamos manipular vários ponteiros, mantendo sempre o último nó apontando para o primeiro nó. Nos trechos de código seguintes, são apresentadas as funções para inserção no início da lista e busca. Ao inserir no início, o primeiro nó é alterado fazendo com que o ponteiro do último nó seja alterado consequentemente visto que, o último ponteiro aponta para o primeiro nó da lista. Já na função de busca em uma lista circular consiste em percorrer os elementos da lista até encontrar o valor desejado ou retornar ao início, caso o elemento não exista. Diferente de uma lista linear, a lista circular não possui um “fim” explícito já

que o último nó aponta novamente para o primeiro, formando um ciclo. Por isso, a busca é implementada com um laço do-while, que garante que o primeiro nó seja verificado antes de comparar com o início. Se o valor buscado for encontrado, a função retorna o nó correspondente; caso contrário, retorna “NULL”.

Figura 10: Função para iniciar a lista circular

```
4.  void inserir_no_inicio(Lista *lista, int num){
5.      No *novo = malloc(sizeof(No));
6.
7.      if(novo){
8.          novo->valor = num;
9.          // o próximo aponta para o início da lista
10.         novo->proximo = lista->inicio;
11.         // novo se torna o início da lista
12.         lista->inicio = novo;
13.         // se fim for nulo (lista vazia), fim aponta para novo nó
14.         if(lista->fim == NULL)
15.             lista->fim = novo;
16.         // fim aponta para início
17.         lista->fim->proximo = lista->inicio;
18.         lista->tam++;
19.     }
20.     else
21.         printf("Erro ao alocar memoria!\n");
22. }
```

Fonte: Martins, Geovanni et al, outubro de 2025.

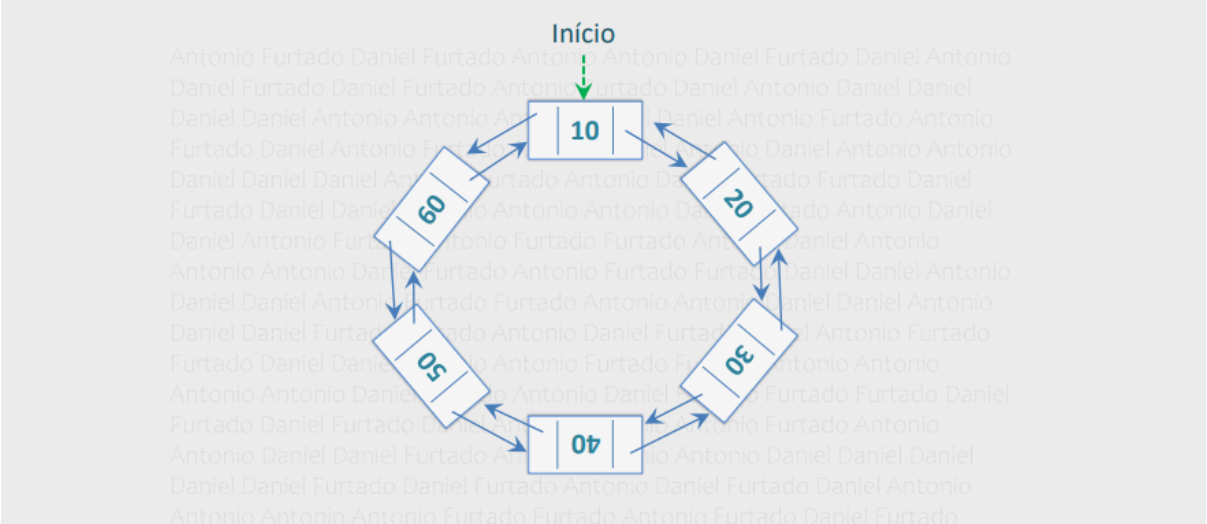
Figura 11: Função de busca em uma lista circular

```
136.  // função para buscar um valor
137.  No* buscar(Lista *lista, int num){
138.      No *aux = lista->inicio;
139.
140.      if(aux){
141.          do{
142.              if(aux->valor == num)
143.                  return aux;
144.              aux = aux->proximo;
145.          }while(aux != lista->inicio);
146.      }
147.      return NULL;
148.  }
```

Fonte: Martins, Geovanni et al, outubro de 2025.

Lista Duplamente Circular: Uma lista duplamente circular é composta por nós que possuem dois ponteiros: next, apontando para o próximo nó, e o ponteiro anterior (chamado de prev) apontando para o nó anterior.

Figura 12: Exemplo didático de uma lista duplamente ligada circular



Fonte: FURTADO, Daniel A. Listas Duplamente Ligadas Circulares.

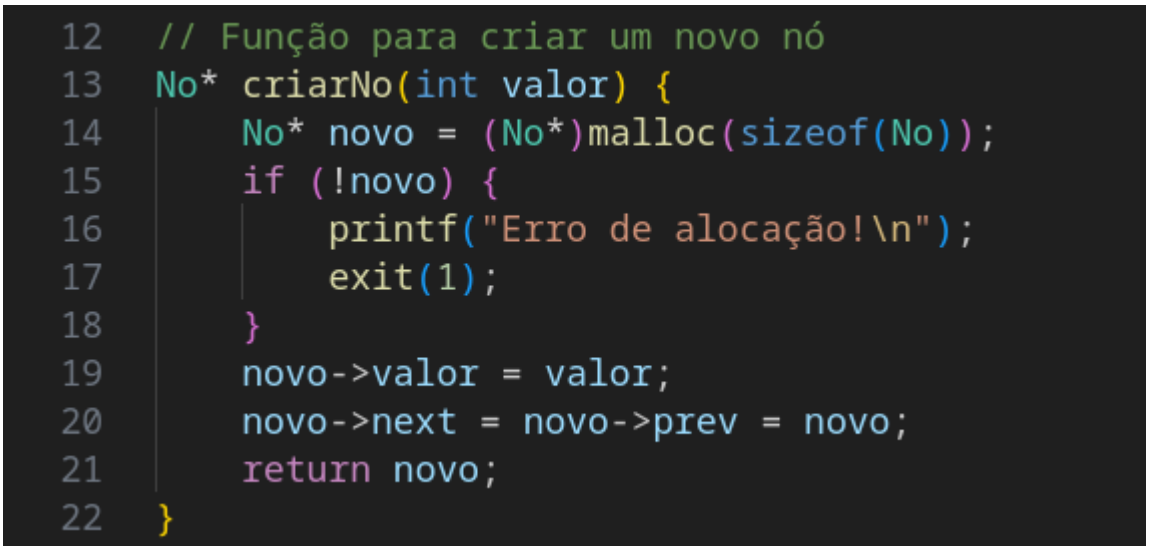
O último nó se liga ao primeiro pelo next, e o primeiro se liga ao último pelo prev, formando um ciclo. Existe apenas uma referência externa para o primeiro nó, mas a partir de qualquer nó é possível percorrer a lista tanto para frente quanto para trás. As inserções e remoções podem ser feitas de forma eficiente no início ou no final da lista, sem depender da referência externa para o nó anterior.

Figura 12: Estrutura do nó da lista duplamente circular

```
5  // Estrutura do nó da lista duplamente circular
6  typedef struct no {
7      int valor;
8      struct no *next;
9      struct no *prev;
10 } No;
```

Fonte: Martins, Geovanni et al, outubro de 2025.

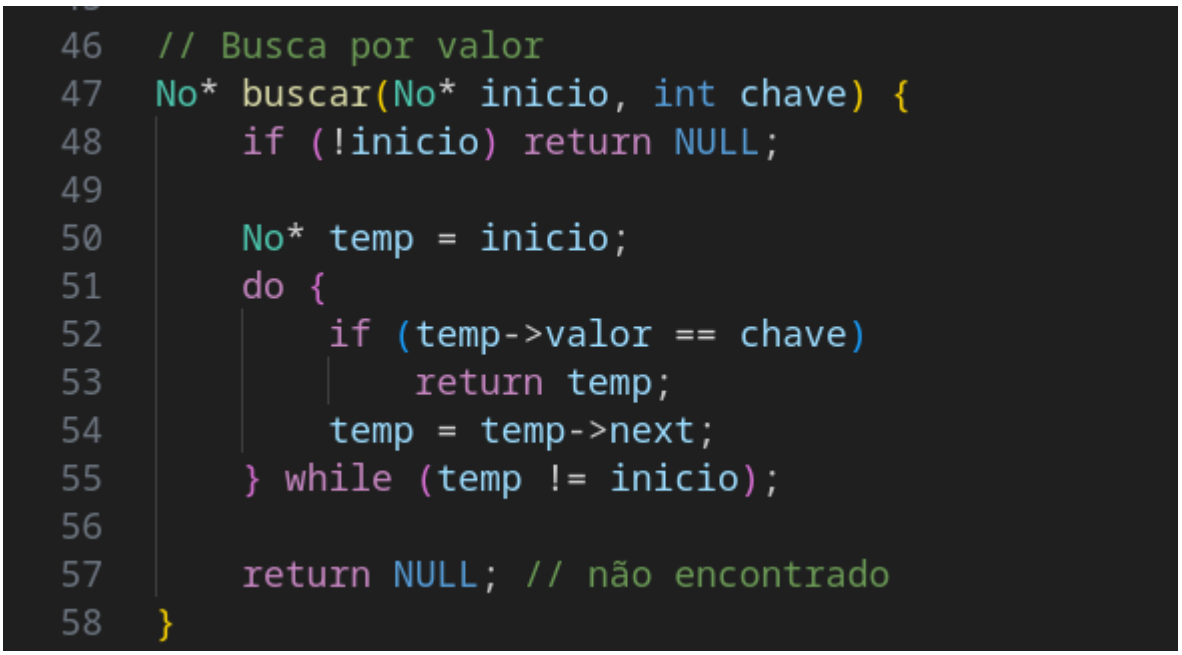
Figura 13: Inserção de novo nó



```
12  // Função para criar um novo nó
13  No* criarNo(int valor) {
14      No* novo = (No*)malloc(sizeof(No));
15      if (!novo) {
16          printf("Erro de alocação!\n");
17          exit(1);
18      }
19      novo->valor = valor;
20      novo->next = novo->prev = novo;
21      return novo;
22  }
```

Fonte: Martins, Geovanni et al, outubro de 2025.

Figura 14: Busca por um valor



```
46  // Busca por valor
47  No* buscar(No* inicio, int chave) {
48      if (!inicio) return NULL;
49
50      No* temp = inicio;
51      do {
52          if (temp->valor == chave)
53              return temp;
54          temp = temp->next;
55      } while (temp != inicio);
56
57      return NULL; // não encontrado
58  }
```

Fonte: Martins, Geovanni et al, outubro de 2025.

Para a remoção de um nó específico, seja por posição ou valor, é necessário o apontamento de um nó para outro e atualização do ponteiro para manter a integridade da lista circular. Ou seja, O nó anterior (prev) do nó removido deve apontar para o próximo nó (prox) do nó removido. O nó seguinte (prox) deve apontar de volta para o nó anterior (prev) do nó removido. Se a cabeça da lista for removida, é necessário atualizar o ponteiro da cabeça.

Figura 15: Remoção de um nó específico em lista duplamente circular

```c
// Remoção de um nó por valor
No* remover(No* inicio, int chave) {
    if (!inicio) return NULL;

    No* temp = inicio;
    do {
        if (temp->valor == chave) {
            if (temp->next == temp) { // único nó
                free(temp);
                return NULL;
            }
            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;
            if (temp == inicio)
                inicio = temp->next; // atualiza início se necessário
            free(temp);
            return inicio;
        }
        temp = temp->next;
    } while (temp != inicio);

    return inicio; // valor não encontrado
}
```

Fonte: Martins, Geovanni et al, outubro de 2025.

2.1.Comparação entre os tipos de lista

Como já discutido, listas encadeadas são estruturas de dados lineares cujos elementos (nós) não ocupam posições contíguas na memória. Cada nó contém um ponteiro que referencia o próximo elemento e, em alguns casos, também o nó anterior (KNUTH, 1999). Essa característica proporciona maior flexibilidade para inserções e remoções em tempo de execução, diferentemente dos arrays, que

exigem realocação de memória. A seguir, apresenta-se uma comparação entre os principais tipos de listas encadeadas, destacando suas características, vantagens e aplicações.

Figura 16: Comparação entre os tipos de lista

Tipo de lista	Ponteiros	Navegação	Aplicações comuns
Simplesmente encadeada	1 (próximo)	Apenas frente	Pilhas, listas lineares simples
Duplamente encadeada	2 (próximo e anterior)	Frente e trás	Editores de texto, histórico
Circular	1 (próximo)	Ciclo contínuo	Filas circulares, escalonamento
Duplamente circular	2 (próximo e anterior)	Ciclo contínuo bidirecional	Buffers, jogos, filas de prioridade

Fonte: Martins, Geovanni et al, outubro de 2025.

Bibliografia

FURTADO, Daniel A. *Listas Duplamente Ligadas Circulares*. 2025. 10 p. Universidade Federal de Uberlândia. Disponível em: https://furtado.prof.ufu.br/site/teaching/ED1/ED1-06-TAD-Listas-Dupla-Circular.pdf. Acesso em: 17 out. 2025.

GOLD, R. *Lista circular*. São Paulo: Instituto de Matemática e Estatística da Universidade de São Paulo, 2001. Disponível em: https://www.ime.usp.br/~gold/cursos/2001/mac2301/lista-circular.pdf. Acesso em: 17 out. 2025. Imagem extraída do documento.

GRÉGIO, André. *Listas Duplamente Encadeadas*. Disponível em: https://www.inf.ufpr.br/gregio/CI1001/ListaDuplamenteEncadeada.pdf. Acesso em: 15 out. 2025.

KNUTH, Donald E. The Art of Computer Programming: Volume 1 – Fundamental Algorithms. 3. ed. Reading, MA: Addison-Wesley, 1997. acesso em 14 de outubro de 2025.

KNUTH, Donald E. *The Art of Computer Programming: Volume 1 – Fundamental Algorithms*. 3. ed. Reading: Addison-Wesley, 1999. Acesso em 17 de outubro de 2025.

STACK OVERFLOW EM PORTUGUÊS. Qual o funcionamento de uma lista encadeada em C? Disponível em:https://pt.stackoverflow.com/questions/222303/qual-o-funcionamento-de-uma-lista-encadeada-em-c. Acesso em 15 outubro de 2025.