

# Lista de exercícios Estrutura de Dados II

Nome: Pedro Henrique dos Santos Oliveira - RGM: 31274820

Curso: Ciência da Computação - Turma: 3A

[pedrohenrique\\_santos@live.com](mailto:pedrohenrique_santos@live.com)

Universidade da Cidade de São Paulo (UNICID) - Rua Cesário Galeno, 448/475  
São Paulo – SP – Brasil – CEP: 03071-000

**Abstract.** *In this project, a circular movie list was implemented in C language, inspired by a Data Structure II activity. The movie list has essential features such as dynamic memory allocation for the inserted movies, functions for searching for movies by name, counting the number of movies in the list, removing and adding movies. The last movie in the list points to the first, creating a continuous loop. This project demonstrates the application of knowledge acquired in Professor Juliano Ratusznei's classes and the ability to divide a problem into logical parts in order to achieve an effective solution.*

**Resumo.** Neste projeto, foi implementada uma lista circular de filmes em linguagem C, inspirada por uma atividade de Estrutura de Dados II. A lista de filmes apresenta características essenciais, como alocação dinâmica de memória para os filmes inseridos, funções para busca de filmes por nome, contagem da quantidade de filmes na lista, remoção e adição de filmes. O último filme na lista aponta para o primeiro, criando um loop contínuo. Este projeto demonstra a aplicação de conhecimentos adquiridos nas aulas do professor Juliano Ratusznei e a capacidade de dividir um problema em partes lógicas para atingir uma solução eficaz.

### Descritiva do problema a ser solucionado

- Implemente uma lista circular com nomes de filmes a qual tenha as seguintes características:
  - Alocação de memória dinâmica para os filmes inseridos;
  - Uma função de busca de filmes por nomes;
  - Contagem da quantidade de filmes presentes na lista;
  - Remoção de filmes;
  - Adição de filmes;
  - O último filme aponta para o primeiro filme da lista.

### Codificação:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>

// Definição da estrutura de dados Filme
struct Filme {
    char titulo[150];
    char sinopse[500];
    struct Filme* proximo;
};

// Criação de um tipo Filme
typedef struct Filme Filme;

// Declaração de variáveis globais
Filme* primeiro = NULL; // Ponteiro para o primeiro filme na lista
Filme* ultimo = NULL;   // Ponteiro para o último filme na lista

// Função para alocar memória para um novo filme
Filme* criarFilme() {
    Filme* novoFilme = (Filme*)calloc(1, sizeof(Filme)); // Usa calloc para
    inicializar as strings com valores nulos

    if (novoFilme == NULL) {
        printf("!!!! |Erro ao alocar memória.| !!!!\n");
        exit(1);
    }

    return novoFilme;
}

// Função para adicionar um filme à lista
void adicionarFilme() {
    Filme* novoFilme = criarFilme(); // Aloca memória para um novo filme

    printf("|Digite o título do filme: ");
    scanf("%[^\n]", novoFilme->titulo); // Lê o título do filme

    printf("|Digite a sinopse do filme: ");
    scanf("%[^\n]", novoFilme->sinopse); // Lê a sinopse do filme

    novoFilme->proximo = NULL; // Inicializa o próximo como nulo

    if (primeiro == NULL) { // Se a lista está vazia
        primeiro = novoFilme;
        ultimo = novoFilme;
        novoFilme->proximo = primeiro;
    }
}
```

```

    } else {
        ultimo->proximo = novoFilme; // Adiciona o novo filme no final da
lista
        novoFilme->proximo = primeiro;
        ultimo = novoFilme;
    }

    printf("|Filme adicionado com sucesso! :)\n");
}

// Função para contar o número de filmes na lista
int contarFilmes() {
    if (primeiro == NULL) {
        return 0;
    }

    Filme* atual = primeiro;
    int count = 0;

    do {
        count++;
        atual = atual->proximo;
    } while (atual != primeiro);

    return count;
}

// Função para remover filmes da lista
void removerFilmes() {
    int escolha, i;

    if (primeiro == NULL) {
        printf("|||A lista está vazia.|||\n");
        return;
    }

    printf("|Digite o número do filme a ser removido: ");
    scanf("%d", &escolha);

    int totalFilmes = contarFilmes();

    if (escolha < 1 || escolha > totalFilmes) {
        printf("!!!!|Escolha inválida: %d\n", escolha);
        return;
    }

    Filme* atual = primeiro;
    Filme* anterior = NULL;

    // Localiza o filme a ser removido
    for (i = 1; i < escolha; i++) {
        anterior = atual;
        atual = atual->proximo;
    }

    // Exibe o título do filme removido
    printf("!!!!|Filme removido: %s\n", atual->titulo);

    if (anterior != NULL) {
        anterior->proximo = atual->proximo;
    } else {

```

```

        primeiro = atual->proximo;
        ultimo->proximo = primeiro;
    }

    free(atual);

    printf("||Filme removido com sucesso.||\n");
}

// Função para listar filmes da lista
void listarFilmes() {
    if (primeiro == NULL) {
        printf("!!!!|Lista vazia.!!!!\n");
        return;
    }

    Filme* atual = primeiro;
    int numero = 1;

    do {
        printf("%d - %s\n", numero, atual->titulo);
        atual = atual->proximo;
        numero++;
    } while (atual != primeiro);

    int escolha;
    printf("|Selecione um filme pelo número: ");
    scanf("%d", &escolha);

    if (escolha >= 1 && escolha < numero) {
        atual = primeiro;
        int i;
        for (i = 1; i < escolha; i++) {
            atual = atual->proximo;
        }

        system("cls");
        printf(">>>|Título: %s\n", atual->titulo);
        printf(">>>|Sinopse: %s\n", atual->sinopse);
    }

    // Adicionar funcionalidade de navegação
    int opcao;
    int i;
    while (1) {
        printf("\n|1 - Próximo|\n|0 - Voltar ao menu anterior|\n");
        scanf("%d", &opcao);

        if (opcao == 0) {
            return; // Retorna ao menu anterior
        } else if (opcao == 1) {
            atual = atual->proximo;
        } else if (opcao == 2) {
            atual = atual->proximo;
            // Navegar para o filme anterior
            for (i = 1; i < escolha - 1; i++) {
                atual = atual->proximo;
            }
        }

        system("cls");
        printf(">>>|Título: %s\n", atual->titulo);
        printf(">>>|Sinopse: %s\n", atual->sinopse);
    }
}

```

```

    }
    printf(">>>|Título: %s\n", atual->titulo);
    printf(">>>|Sinopse: %s\n", atual->sinopse);
}

void buscarFilmePorTitulo() {
    if (primeiro == NULL) {
        printf(">>>>|Lista vazia.<<<<\n");
        return;
    }

    char tituloBusca[1050];
    printf("|Digite o título do filme a ser buscado: ");
    scanf(" %[^\\n]", tituloBusca);

    Filme* atual = primeiro;
    int encontrado = 0;

    do {
        if (strcmp(atual->titulo, tituloBusca) == 0) {
            printf(">>>|Título: %s\n", atual->titulo);
            printf(">>>|Sinopse: %s\n", atual->sinopse);
            encontrado = 1;
            break;
        }
        atual = atual->proximo;
    } while (atual != primeiro);

    if (!encontrado) {
        printf("!!! Filme não encontrado na lista. !!!\n");
    }
}

void liberarFilme(Filme* filme) {
    free(filme);
}

// Função para liberar a memória de todos os filmes
void liberarMemoria() {
    Filme* atual = primeiro;
    do {
        Filme* temp = atual;
        atual = atual->proximo;
        liberarFilme(temp); // Libera a memória de cada filme
    } while (atual != primeiro);

    if (primeiro != NULL) {
        liberarFilme(primeiro); // Libera o primeiro filme
    }
}

int main() {
    setlocale(LC_ALL, "Portuguese");
    int opcao;
    while (1) {
        printf("\n-----| Menu |-----\n");
        printf("| 1 - Adicionar Filme\n");
        printf("| 2 - Remover Filme\n");
        printf("| 3 - Lista de Filmes Adicionados\n");
        printf("| 4 - Buscar Filme por Título\n"); // Adiciona a opção de
        busca por título
        printf("| 5 - Encerrar\n");
    }
}

```

```

printf("_ _ _|Escolha uma opção|_ _ _ \n | ");
scanf("%d", &opcao);
system("cls"); // Limpa a tela

switch (opcao) {
    case 1:
        adicionarFilme();
        break;
    case 2:
        removerFilmes();
        break;
    case 3:
        listarFilmes();
        break;
    case 4:
        buscarFilmePorTitulo();
        break;
    case 5:
        liberarMemoria(); // Libera a memória alocada para a lista
de filmes
        exit(0);
    default:
        printf("!!! Opção inválida. Tente novamente. !!!\n");
        break;
}

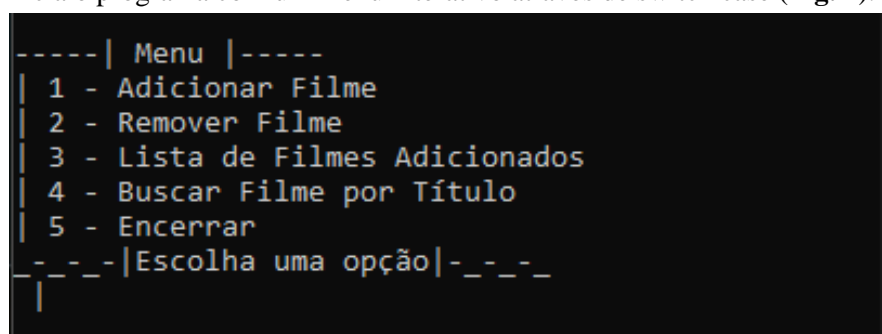
return 0;
}

```

A codificação do sistema foi desenvolvida em C e é composta por um conjunto de funções que realizam diferentes operações. Abaixo, fornecerei uma explicação detalhada do que cada função faz:

#### **main():**

- Inicia o programa com um menu interativo através de switch case (**Fig. 1**).



```

-----| Menu |-----
| 1 - Adicionar Filme
| 2 - Remover Filme
| 3 - Lista de Filmes Adicionados
| 4 - Buscar Filme por Título
| 5 - Encerrar
-----|Escolha uma opção|-----
|

```

**Figura 1** Menu inicial.

```

int main() {
    setlocale(LC_ALL, "Portuguese");
    int opcao;
    while (1) {
        printf("\n-----| Menu |-----\n");
        printf("| 1 - Adicionar Filme\n");
        printf("| 2 - Remover Filme\n");
        printf("| 3 - Lista de Filmes Adicionados\n");
    }
}

```

```

        printf("| 4 - Buscar Filme por Título\n"); // Adiciona a
opção de busca por título
        printf("| 5 - Encerrar\n");
        printf("_ _ _|Escolha uma opção|_ _ _ \n | ");
        scanf("%d", &opcao);
        system("cls"); // Limpa a tela

        switch (opcao) {
            case 1:
                adicionarFilme();
                break;
            case 2:
                removerFilmes();
                break;
            case 3:
                listarFilmes();
                break;
            case 4:
                buscarFilmePorTitulo();
                break;
            case 5:
                liberarMemoria(); // Libera a memória alocada para
a lista de filmes
                exit(0);
            default:
                printf("!!! Opção inválida. Tente novamente.
!!!\n");
                break;
        }
    }

    return 0;
}

```

#### **criarFilme():**

- Esta função é responsável por alocar memória dinamicamente para um novo filme.
- Utiliza a função *calloc* para alocar memória e inicializa as *strings* título e sinopse com valores nulos.
- Se a alocação de memória falhar, exibe uma mensagem de erro e encerra o programa.

```

Filme* criarFilme() {
    Filme* novoFilme = (Filme*)calloc(1, sizeof(Filme));
    // Usa calloc para inicializar as strings com valores nulos

    if (novoFilme == NULL) {
        printf("!!!! |Erro ao alocar memória.| !!!!\n");
        exit(1);
    }

    return novoFilme;
}

```

#### **adicionarFilme():**

- Permite ao usuário adicionar um novo filme à lista.
- Chama a função **criarFilme()** para alocar memória para o novo filme.
- Solicita ao usuário que insira o título e a sinopse do filme (**Fig. 2**).

- Inicializa o próximo filme como nulo.
- Se a lista estiver vazia, define o primeiro filme como o novo filme e o último filme também como o novo filme, estabelecendo a propriedade da lista circular.
- Caso contrário, ajusta os ponteiros para adicionar o novo filme no final da lista e atualiza o último filme.

```
|Digite o título do filme: A freira
|Digite a sinopse do filme: Presa em um convento na
a Romênia, uma freira comete suicídio. Para investigar
o caso, o Vaticano envia um padre assombrado e uma
noviça prestes a se tornar freira. Arriscando suas
vidas, a fé e até suas almas, os dois descobrem um
segredo profano e se confrontam com uma força do
mal que toma a forma de uma freira demoníaca e
transforma o convento em um campo de batalha.
|Filme adicionado com sucesso! :)
```

*Figura 2 Execução da opção 1 – Adicionar Filme*

```
void adicionarFilme() {
    Filme* novoFilme = criarFilme(); // Aloca memória para um novo
    filme

    printf("|Digite o título do filme: ");
    scanf(" %[^\\n]", novoFilme->titulo); // Lê o título do filme

    printf("|Digite a sinopse do filme: ");
    scanf(" %[^\\n]", novoFilme->sinopse); // Lê a sinopse do filme

    novoFilme->proximo = NULL; // Inicializa o próximo como nulo

    if (primeiro == NULL) { // Se a lista está vazia
        primeiro = novoFilme;
        ultimo = novoFilme;
        novoFilme->proximo = primeiro;
    } else {
        ultimo->proximo = novoFilme; // Adiciona o novo filme no
        final da lista
        novoFilme->proximo = primeiro;
        ultimo = novoFilme;
    }

    printf("|Filme adicionado com sucesso! :)\\n");
}
```

#### **contarFilmes():**

- Conta e retorna o número de filmes presentes na lista.
- Percorre a lista circular e incrementa um contador a cada iteração até retornar ao primeiro filme.
- Se a lista estiver vazia, retorna 0.

```
int contarFilmes() {
    if (primeiro == NULL) {
        return 0;
    }
}
```



```

    Filme* atual = primeiro;
    int count = 0;

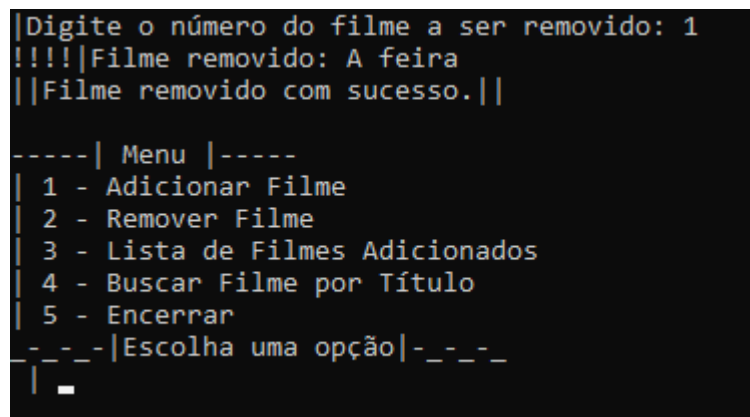
    do {
        count++;
        atual = atual->proximo;
    } while (atual != primeiro);

    return count;
}

```

### removerFilmes():

- Permite ao usuário remover um filme da lista.
- Solicita ao usuário o número do filme a ser removido (**Fig. 3**).
- Verifica se a lista está vazia e se a escolha do usuário está dentro dos limites.
- Localiza o filme a ser removido, ajusta os ponteiros do filme anterior para pular o filme a ser removido e libera a memória alocada para o filme.
- Se o filme a ser removido for o primeiro da lista, atualiza o ponteiro do primeiro filme.
- Certifica-se de que o último filme ainda aponta para o primeiro para manter a lista circular.



```

|Digite o número do filme a ser removido: 1
|!!!|Filme removido: A feira
|Filme removido com sucesso.||
|
|-----| Menu |-----
| 1 - Adicionar Filme
| 2 - Remover Filme
| 3 - Lista de Filmes Adicionados
| 4 - Buscar Filme por Título
| 5 - Encerrar
|_ _ _|Escolha uma opção|_ _ _
|
|

```

**Figura 3** Execução da opção 2 - Remover Filme

```

void removerFilmes() {
    int escolha, i;

    if (primeiro == NULL) {
        printf("!!!A lista está vazia.!!!\n");
        return;
    }

    printf("|Digite o número do filme a ser removido: ");
    scanf("%d", &escolha);

    int totalFilmes = contarFilmes();

    if (escolha < 1 || escolha > totalFilmes) {
        printf("!!!Escolha inválida: %d\n", escolha);
        return;
    }

    Filme* atual = primeiro;

```

```

Filme* anterior = NULL;

// Localiza o filme a ser removido
for (i = 1; i < escolha; i++) {
    anterior = atual;
    atual = atual->proximo;
}

// Exibe o título do filme removido
printf("!!!!|Filme removido: %s\n", atual->titulo);

if (anterior != NULL) {
    anterior->proximo = atual->proximo;
} else {
    primeiro = atual->proximo;
    ultimo->proximo = primeiro;
}

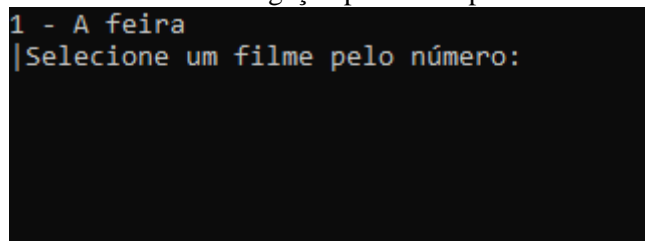
free(atual);

printf("||Filme removido com sucesso.||\n");
}

```

#### listarFilmes():

- Exibe a lista de filmes na tela, permitindo ao usuário selecionar um filme para visualizar detalhes.
- Percorre a lista circular e exibe o número e o título de cada filme.
- Solicita ao usuário que selecione um filme pelo número (**Fig. 4**).
- Exibe o título e a sinopse do filme selecionado.
- Oferece funcionalidade de navegação para ver o próximo filme (**Fig. 5**).

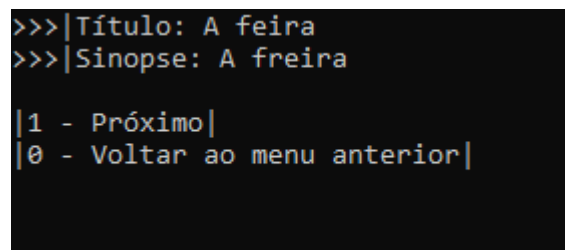


```

1 - A feira
|Selecione um filme pelo número:

```

**Figura 4** Execução da opção 3 – Lista de Filmes Adicionados



```

>>>|Título: A feira
>>>|Sinopse: A freira

|1 - Próximo|
|0 - Voltar ao menu anterior|

```

**Figura 5** Execução da opção 3 – Passar para o próximo filme

```

void listarFilmes() {
    if (primeiro == NULL) {
        printf("!!!!|Lista vazia.!!!!\n");
        return;
    }
}

```

```

Filme* atual = primeiro;
int numero = 1;

do {
    printf("%d - %s\n", numero, atual->titulo);
    atual = atual->proximo;
    numero++;
} while (atual != primeiro);

int escolha;
printf("|Selecione um filme pelo número: ");
scanf("%d", &escolha);

if (escolha >= 1 && escolha < numero) {
    atual = primeiro;
    int i;
    for (i = 1; i < escolha; i++) {
        atual = atual->proximo;
    }
    system("cls");
    printf(">>>|Título: %s\n", atual->titulo);
    printf(">>>|Sinopse: %s\n", atual->sinopse);
}

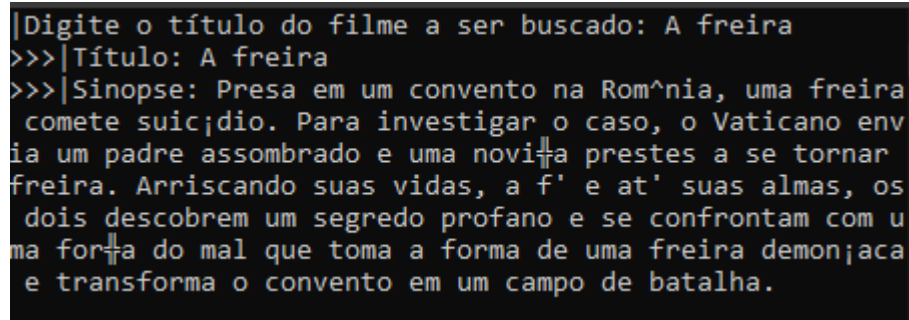
// Adicionar funcionalidade de navegação
int opcao;
int i;
while (1) {
    printf("\n|1 - Próximo|\n|0 - Voltar ao menu anterior|\n");
    scanf("%d", &opcao);

    if (opcao == 0) {
        return; // Retorna ao menu anterior
    } else if (opcao == 1) {
        atual = atual->proximo;
    } else if (opcao == 2) {
        atual = atual->proximo;
        // Navegar para o filme anterior
        for (i = 1; i < escolha - 1; i++) {
            atual = atual->proximo;
        }
    }
    system("cls");
    printf(">>>|Título: %s\n", atual->titulo);
    printf(">>>|Sinopse: %s\n", atual->sinopse);
}
printf(">>>|Título: %s\n", atual->titulo);
printf(">>>|Sinopse: %s\n", atual->sinopse);
}

```

### buscarFilmePorTitulo():

- Permite ao usuário buscar um filme na lista pelo título (**Fig. 6**).
- Solicita ao usuário que insira o título do filme a ser buscado.
- Percorre a lista circular e compara o título de cada filme com o título de busca.
- Se um filme com o título desejado for encontrado, exibe o título e a sinopse do filme.



```
|Digite o título do filme a ser buscado: A freira
>>>|Título: A freira
>>>|Sinopse: Presa em um convento na Rom^nia, uma freira
comete suicídio. Para investigar o caso, o Vaticano env
ia um padre assombrado e uma noviça prestes a se tornar
freira. Arriscando suas vidas, a f' e at' suas almas, os
dois descobrem um segredo profano e se confrontam com u
ma força do mal que toma a forma de uma freira demoníaca
e transforma o convento em um campo de batalha.
```

*Figura 6 Execução da opção 4 - Buscar filme por título*

```
void buscarFilmePorTitulo() {
    if (primeiro == NULL) {
        printf(">>>>|Lista vazia.|<<<<\n");
        return;
    }

    char tituloBusca[1050];
    printf("|Digite o título do filme a ser buscado: ");
    scanf(" %[^\\n]", tituloBusca);

    Filme* atual = primeiro;
    int encontrado = 0;

    do {
        if (strcmp(atual->titulo, tituloBusca) == 0) {
            printf(">>>|Título: %s\\n", atual->titulo);
            printf(">>>|Sinopse: %s\\n", atual->sinopse);
            encontrado = 1;
            break;
        }
        atual = atual->proximo;
    } while (atual != primeiro);

    if (!encontrado) {
        printf("!!! Filme não encontrado na lista. !!!\\n");
    }
}
```

### **liberarFilme(Filme\* filme):**

- Esta função é uma função auxiliar para liberar a memória alocada para um filme específico.
- Recebe um ponteiro para um filme e libera a memória associada a esse filme.

```
void liberarFilme(Filme* filme) {  
    free(filme);  
}
```

### **liberarMemoria():**

- Libera a memória alocada para todos os filmes na lista.
- Percorre a lista circular e, para cada filme, chama a função **liberarFilme()** para liberar a memória associada a esse filme.
- Após liberar a memória de todos os filmes, libera a memória alocada para o primeiro filme.

```
void liberarMemoria() {  
    Filme* atual = primeiro;  
    do {  
        Filme* temp = atual;  
        atual = atual->proximo;  
        liberarFilme(temp); // Libera a memória de cada  
filme  
    } while (atual != primeiro);  
  
    if (primeiro != NULL) {  
        liberarFilme(primeiro); // Libera o primeiro  
filme  
    }  
}
```

### **Aprendizagem Obtida:**

Este código em linguagem C cria e gerencia uma lista circular de filmes, utilizando alocação de memória dinâmica para criar estruturas de dados que representam os filmes. As principais funções incluem a adição de filmes à lista, a remoção de filmes, a listagem de filmes e a busca por título. A lista é circular, onde o último filme aponta de volta para o primeiro, permitindo iterações contínuas. O programa é interativo, fornecendo um menu para o usuário, e também demonstra boas práticas, como liberação de memória. Ao explorar esse código, é possível aprender sobre alocação dinâmica de memória, estruturas de dados personalizadas, manipulação de listas, interatividade com o usuário e resolução de problemas de programação.

## Referências:

PROGRAME SEU FUTURO. Curso de Programação C | Como implementar uma LISTA CIRCULAR? Lista Encadeada Circular | aula 258. [S.l.: s.n.], 2021. 1 vídeo (34:28 min). Publicado em 02/06/2021. Disponível em: <[https://www.youtube.com/watch?v= dWDe6JnMzg](https://www.youtube.com/watch?v=dWDe6JnMzg)>. Acesso em: 19 de setembro de 2023.

ALGORITMOS. Alocação dinâmica: introdução a malloc e calloc. YouTube, 17 de agosto de 2020. Disponível em: <[https://www.youtube.com/watch?v= reV9kQVLt0](https://www.youtube.com/watch?v=reV9kQVLt0)>. Acesso em: 30/09/2023

UNIVERSIDADE FEDERAL DO PARANÁ. 32 Alocação dinâmica de memória. Disponível em:<[https://www.inf.ufpr.br/nicolui/Docs/Livros/C/ArmandoDelgado/notas-32 Aloca\\_c\\_cao\\_dinamica\\_mem.html](https://www.inf.ufpr.br/nicolui/Docs/Livros/C/ArmandoDelgado/notas-32_Aloca_c_cao_dinamica_mem.html)>. Acesso em: 30/09/2023.

OPENAI. ChatGPT. 2021. Disponível em:<<https://chat.openai.com/>>. Acesso em: 28 de setembro de 2023.