INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

Catálogo de Filmes

Lucas Miguel Jacobina Teixeira
Tales Ian Costa Barkema

JUSTIFICATIVA

- Gerenciamento de dados
- Flexibilidade
- Expansibilidade

OBJETIVO

- Gerenciar uma lista de filmes;
- Adicionar Filmes;
- Remover filmes;
- Exibir lista de filmes;

BIBLIOTECAS

- #include <stdio.h>: Funções de entrada e saída.
- #include <stdlib.h>: Alocação dinâmica de memória.
- #include <string.h>: Manipulação de strings.

DESENVOLVIMENTO

Struct Filme:

- **Estrutura ‘film’:**

```
typedef struct filme {
    char nome[50];
    int duration;
    int id;
    struct filme *next;
    struct filme *prev;
} film;
```

DESENVOLVIMENTO

Variável Global:

- ‘lista’ é um ponteiro global que aponta para o primeiro elemento da lista de filmes. Inicialmente, ele é NULL, indicando que a lista está vazia.

```
film *lista = NULL;
```

DESENVOLVIMENTO

Função Criar Lista:

- Aloca memória para um novo nó da lista, inicializa seus campos com os valores fornecidos e retorna o ponteiro para esse nó.

```c
film *criar_list(int id, char *nome, int duration) {
    film *new_node = (film*) malloc(sizeof(film));

    new_node->id = id;
    strcpy(new_node->nome, nome);
    new_node->duration = duration;
    new_node->next = NULL;
    new_node->prev = NULL;

    return new_node;
}
```

DESENVOLVIMENTO

Função de Inserção

- Insere um novo nó no final da lista.
- Se a lista estiver vazia (lista == NULL), o novo nó torna-se o primeiro elemento.
- Caso contrário, percorre a lista até o final e adiciona o novo nó lá, ajustando os ponteiros next e prev de forma adequada.

```c
void insertion(int id, char *nome, int duration) {
    film *new_node = criar_list(id, nome, duration);

    if (lista == NULL) {
        lista = new_node;
    } else {
        film *atual = lista;
        while (atual->next != NULL) {
            atual = atual->next;
        }

        atual->next = new_node;
        new_node->prev = atual;
    }
}
```

DESENVOLVIMENTO

Função Remover:

- Remove um nó da lista baseado no id fornecido.
- Percorre a lista em busca do nó com o id especificado.
- Ajusta os ponteiros next e prev dos nós adjacentes para excluir o nó da lista.
- Libera a memória alocada para o nó removido.

```c
void removing(int id) {
    film *ptr = lista;
    while (ptr != NULL) {
        if (ptr->id == id) {
            if (ptr->prev == NULL) {
                lista = ptr->next;
                if (lista != NULL) {
                    lista->prev = NULL;
                }
            } else if (ptr->next == NULL) {
                ptr->prev->next = NULL;
            } else {
                ptr->prev->next = ptr->next;
                ptr->next->prev = ptr->prev;
            }

            free(ptr);
            printf("Removido\n");
            return;
        }
        ptr = ptr->next;
    }
    printf("Filme nao encontrado\n");
}
```

DESENVOLVIMENTO

Função Exibir Lista:

- Percorre e exibe todos os nós da lista, mostrando o nome, duração e id de cada filme.

```c
void ver_list(film *lista) {
    film *ptr = lista;
    while (ptr != NULL) {
        printf("\n\nNome do filme: %s\n", ptr->nome);
        printf("Duracao: %d min\n", ptr->duration);
        printf("Posicao na lista: %d\n", ptr->id);
        printf("_____\n");
        ptr = ptr->next;
    }
}
```

RESULTADOS

Funções:

Complexidade Computacional

- `criar_list`: O(1)
- `insertion`: O(n)
- `removing`: O(n)
- `ver_list`: O(n)
- `main`: Depende das interações do usuário, com cada chamada relevante tendo complexidade O(n).

Demonstração do Sistema

Obrigado!