Relatório Trabalho Aed 1

Aluno: Luiz Henrique Rodrigues Ferreira

1- Introdução:

O problema consistia em desenvolver um programa em linguagem C para gerenciar uma lista de filmes, onde cada filme contém uma lista de atores. A ideia central era permitir ao usuário cadastrar filmes, associar atores, remover itens, buscar elementos e realizar operações adicionais como listar favoritos e imprimir relatórios, a partir de um arquivo de filmes e atores.

Para resolver o problema, utilizei listas duplamente encadeadas com nó descritor.

A lista principal contém os filmes e cada filme por sua vez contém uma lista de atores.

O descritor permite armazenar ponteiros para o início e fim da lista, além da quantidade de elementos, facilitando a inserção e remoção.

Um campo extra (favorito) foi adicionado à struct Filme para permitir funcionalidades adicionais.

Assim, a estrutura ficou organizada e flexível, possibilitando percorrer as listas em ambos os sentidos.

Também foram implementadas funções de inicialização das listas, criar filme e ator, inserir filme e ator no final das listas, busca de filme ou ator, remoção de filme ou ator, imprimir filmes ou atores, imprimir as listas invertidas (o que permite demonstrar a propriedade das listas duplamente encadeadas de percorrer a lista começando do fim e chegando

ao início), imprimir a quantidade de atores e de filmes, liberar as listas, favoritar algum filme e imprimir a lista de filmes favoritos. Além das funções de carregar o arquivo e o menu de opções.

A main foi implementada como um menu que permite ao usuário escolher a opção do que ele deseja fazer com o programa.

2- Documentação do código:

Principais funções implementadas:

- inicializarListaFilmes / inicializarListaAtores: preparam listas vazias. Essas funções foram construídas deixando o descritor nulo e a quantidade de filmes e de atores nulos tambem

- criarFilme / criarAtor: alocam memória para novos elementos. Essas funções foram construídas alocando memória para os ponteiros de filme e ator, passando o titulo e o nome para as respectivas variáveis e deixando os ponteiros prox e ant nulos.

- inserirFilme / inserirAtor: inserem no final da lista. Essas funções foram construídas criando um novo filme ou ator. Confere se a lista está vazia. Se estiver, tanto o início quanto o fim recebem o novo filme ou ator. Se a lista já tiver filme ou ator, inserimos eles no final, fazendo o ponteiro do último elemento apontar para o novo, o anterior do novo apontar para o ultimo e atualizando o ponteiro do fim da lista, para que o fim seja o novo.

- buscarFilme / buscarAtor: percorrem a lista para

localizar itens. Essas funções foram utilizadas usando um ponteiro auxiliar tanto para filme, quanto para ator. Esse ponteiro percorre as listas até achar o filme ou ator procurado e retorna eles, caso acho. Se chegar ao final da lista e não achar, retorna NULL.

- removerFilme / removerAtor: removem itens, ajustando encadeamentos. Essas funções utilizam as funções de busca para tentar encontrar o filme ou ator a ser removido. Se encontra, analisa se é o primeiro ou último filme ou ator da lista. Caso seja, deve atualizar o ponteiro início ou fim do descritor, para que ele aponte para o segundo elemento (caso seja o inicio) ou para o penúltimo elemento (caso seja o fim). No final, libera o filme ou ator removido e retira um do número de filmes ou atores das listas.

- imprimirFilmes / imprimirAtores: exibem todos os elementos da lista. Utiliza um ponteiro auxiliar que percorre toda a lista, partindo do início e indo até o fim e imprimindo cada filme ou ator da lista.

- imprimirFilmesInvertido / imprimirAtoresInvertido: percorrem do fim para o início. Utilizam também um ponteiro auxiliar que percorre toda a lista, mas dessa vez partindo do fim e indo até o início e imprimindo cada filme ou ator da lista em ordem invertida

- imprimirQtdAtoresPorFilme: mostra a quantidade de atores em cada filme. A função percorre a lista de filmes, utilizando um ponteiro auxiliar que é atualizado até chegar ao fim e imprime quantos atores tem em cada filme cadastrado.

- imprimirQtdFilmes: mostra o total de filmes cadastrados. A função imprime a quantidade de filmes na lista de filmes.

- favoritarFilme / imprimirFavoritos: permitem favoritar

filmes e listá-los. A função favoritarFilme utiliza da função de buscar para encontrar o filme a ser favoritado e marca o filme como favorito. A função imprimirFavoritos utiliza um ponteiro auxiliar de filmes para percorrer a lista do inicio ao fim e imprimir os filmes favoritados, caso tenha encontrado.

- liberarListaAtores / liberarListaFilmes: liberam memória.

Essas funções utilizam ponteiros auxiliar de filmes e atores para percorrer as determinadas listas e utiliza o ponteiro temporário de filmes ou atores que é liberado a partir da função free para liberar as listas de filmes e atores.

- carregarArquivo: lê dados de filmes e atores de um

arquivo de texto. Essa função abre o arquivo para a leitura, lê a linha do título do filme, remove o \n do final, copia a linha para o titulo do filme, lê a próxima linha que é de quantidade de atores, transforma em um inteiro essa linha, insere o novo filme. Depois aponta para o último filme inserido. Por fim, lê cada ator a partir do loop for, remove o \n e insere o ator no filme

- menu: exibe opções de interação com o usuário.

3- Exemplos de uso:

Exemplo 1: Carregar os filmes do arquivo

Entrada (arquivo filme.txt):

Matrix

3

Keanu Reeves

Laurence Fishburne

Carrie-Anne Moss

Titanic

2

Leonardo DiCaprio

Kate Winslet

Saída:

Filme: Matrix (3 atores)

- Keanu Reeves

- Laurence Fishburne

- Carrie-Anne Moss

Filme: Titanic (2 atores)

- Leonardo DiCaprio

- Kate Winslet

Exemplo 2: Favoritar um filme

Entrada:

12

Digite o título do filme a favoritar: Matrix

13

Saída:

O filme 'Matrix' foi adicionado aos favoritos!

Matrix (3 atores)

Exemplo 3: Relatórios

Entrada:

9

10

Saída:

O filme 'Matrix' tem 3 atores cadastrados.

O filme 'Titanic' tem 2 atores cadastrados.

Quantidade de filmes cadastrados: 2

4- Conclusão:

Durante a implementação do programa, meus maiores desafios Foram pensar em como o programa seria estruturado, a fim de comportar uma lista duplamente encadeada de filmes dentro de uma lista duplamente encadeada de Atores. Também foi muito desafiador a implementação de inserções e remoções, onde os ponteiros teriam que ser modificados. Essa é uma parte complexa e que exige atenção redobrada. Para mim, com certeza a parte mais complicada foi criar a função que carregava o arquivo, porque é uma área que eu não tinha muita experiência e foi necessário muita

pesquisa para fazer isso. Por fim, concluo que foi um trabalho muito legal para aprimorar meus conhecimentos em estruturas de dados.

O programa final cumpre os objetivos: gerencia filmes e seus atores, suporta inserções, buscas, remoções, relatórios e recursos extras como favoritos.