Trabalho 2 - Estruturas de Dados - Sistema CINE USP

Integrantes:
Ana Paula de Abreu Batista (Nº USP 126884242)
Italo Carlos Martins Bresciani (Nº USP 15461782)
Luiz Gabriel Correia dos Santos (Nº USP 15639682)

1. Resumo do Trabalho Desenvolvido:

Para o desenvolvimento do sistema Cine USP, uma **Lista Circular Encadeada e Ordenada (TAD Lista)** foi utilizada para armazenar filmes e seus gêneros, permitindo inserções e remoções eficientes. Além disso, uma **Árvore Binária de Busca (TAD Árvore)** organiza os usuários e os filmes, facilitando buscas rápidas por meio de chaves (Nº USP). A Lista é composta por blocos que contêm o nome do filme, o gênero associado e referências, enquanto a Árvore utiliza nós que armazenam informações como chave de busca, dados do usuário e listas de filmes ordenados vinculadas.

Na função principal (main), implementamos funcionalidades como **`criar_cadastro`**, que insere novos usuários no sistema, **`listar_filmes`**, que exibe filmes cadastrados em ordem alfabética, **buscar_usuario**, que localiza usuários na árvore, **recomendar_colega**, que sugere amigos com interesses semelhantes calculados pela **distância euclidiana** entre as preferências de filmes por gênero, e **mais_mencionado**, que exibe o título mais mencionado. A recomendação é feita comparando os vetores de quantidade de filmes por gênero entre os usuários. A distância euclidiana é calculada pela fórmula:

$$d = \sum_{i=1}^{8} (vetor\ usuário\ 1[i] - vetor\ usuário\ 2[i])^2$$

O colega com d menor é considerado o mais similar. Incluímos também **`exportar_dados`**, que salva informações em arquivo texto, **`exibir_dados_arvore`**, que fornece dados sobre a estrutura da árvore e outras funções necessárias para o funcionamento correto do sistema.

Análise e Justificativa das Estruturas de Dados:

1.1. Lista Circular Encadeada e Ordenada (TAD Lista):

```c
typedef char elem_lista;
typedef int tipo_do_bloco;

#define QUANT 8

typedef struct listabloco
{
    elem_lista *dado;
    tipo_do_bloco tipo;
    tipo_do_bloco cont_ref;
    struct listabloco *prox, *ant;
} ListaBloco;

typedef struct lista
{
    ListaBloco *inicio;
    int vetor[QUANT];

} Lista;
```

O tipo de dado da lista, **`elem_lista`**, foi definido como **`char`**, o que permite armazenar o nome do filme em formato de texto. O tipo de dado **`tipo_do_bloco`** foi definido como **`int`**, permitindo armazenar o gênero do filme por meio de um código.

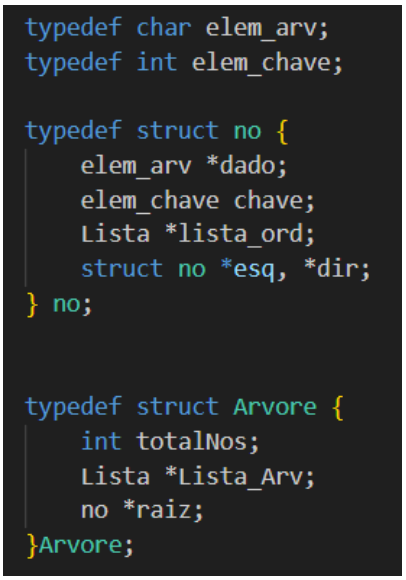
A estrutura **`ListaBloco`** representa um **nó individual na lista**, que contém os seguintes campos:

`elem_lista *dado:` Um ponteiro para o dado que armazena o nome do filme, com o armazenamento flexível para cada nome.
`tipo_do_bloco tipo:` Campo para armazenar o gênero do filme especificado.
`tipo_do_bloco cont_ref:` Campo para armazenar a quantidade de vezes que o filme foi mencionado (usado na Lista geral de filmes cadastrados no sistema, **`Lista_Arv`**).
`ListaBloco *proximo:` Um ponteiro para o próximo bloco (nó) na lista. Esse campo cria o encadeamento de blocos, permitindo navegação sequencial (lógica) na lista de produtos.
`ListaBloco *anterior:` Um ponteiro para o bloco anterior na lista.

A estrutura **`Lista`** contém os seguintes campos:

`ListaBloco *inicio:` Ponteiros para o primeiro bloco da lista.
`tipo_do_bloco vetor[QUANT]:` Um vetor que armazena a quantidade de filmes cadastrados pelo usuário, agrupados por gênero. Esse vetor é essencial para calcular a similaridade entre usuários, utilizando métricas como a distância euclidiana, ao comparar suas preferências de gênero.

1.2. Árvore Binária de Busca (TAD Arvore):



```c
typedef char elem_arv;
typedef int elem_chave;

typedef struct no {
    elem_arv *dado;
    elem_chave chave;
    Lista *lista_ord;
    struct no *esq, *dir;
} no;


typedef struct Arvore {
    int totalNos;
    Lista *Lista_Arv;
    no *raiz;
}Arvore;
```

Definimos o tipo de dado da árvore **elem_arv** como **char** para armazenar os nomes dos usuários e o **elem_arv** como **int** para armazenar o número USP do usuário .

A estrutura **no** representa **um nó individual da árvore**, que contém:

elem_arv *dado: Ponteiro que armazena dinamicamente o nome do usuário.
elem_chave chave: Tipo de dado que armazena o número USP do usuário, será o valor de ordenação e busca na ABB.
Lista *lista_ord: Esse ponteiro aponta para uma lista de filmes (do TAD Lista), que armazena os nomes dos filmes favoritos da pessoa, e seus respectivos gêneros em ordem alfabética.
no *esq,*dir: Ponteiros para os nós filhos esquerda e direita de um nó da árvore, possibilitando a ligação entre os blocos (nós) e permitindo a navegação e flexibilidade na manipulação da árvore.

A estrutura **Arvore** contém os seguintes campos:

no *raiz: Ponteiros para a raiz da árvore.
Lista *Lista_Arv: Esse ponteiro aponta para uma lista de filmes (do TAD Lista), que armazena os nomes de todos os filmes cadastrados no sistema (sem repetição), e seus respectivos gêneros em ordem alfabética.
int totalNos: Um contador que armazena o número total de usuários na árvore.

1.3. Tratamento de erros:

O tratamento de erros se deu pela estrutura **enum** que permite designar às constantes inteiras (0, 1, 2, 3, 4, 5) palavras-chave como:

- `SUCESSO`: indica que a operação foi executada com sucesso;
- `ERRO_ALLOC`: indica que houve erro de alocação de memória;
- `ERRO_NULL`: indica que houve a tentativa de acesso a um ponteiro ou dado nulo;
- `ERRO_ELEM_REPETIDO`: indique houve a tentativa de inserção de um elemento repetido a uma estrutura de dados;
- `ERRO_POP_VAZIO`: indica que houve a tentativa de retirar um elemento de uma estrutura de dados vazia;
- `ERRO_DEALLOC`: indique que houve erro de desalocação de memória.

2. Instruções de Compilação e Execução do Programa:

2.1. Ambiente de Desenvolvimento

Compilador: GCC (GNU Compiler Collection) versão 11.4.0

Sistema Operacional: Ubuntu 22.04.4 LTS

Bibliotecas Utilizadas: O código utiliza apenas bibliotecas padrão da linguagem C, como `stdio.h` e `stdlib.h`, o que facilita a portabilidade para diferentes sistemas operacionais e versões do GCC. Além das bibliotecas `locale.h, string.h, ctype.h, limits.h, math.h e stdbool.h.`

2.2. Passos para Compilação

- **Ambiente LINUX:**

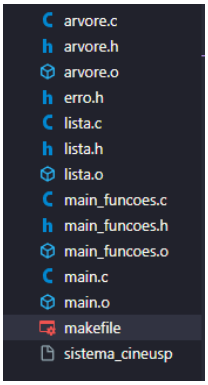
Acesse o Diretório do Código-Fonte: Em um terminal, navegue até o diretório onde o código-fonte está salvo.

Make: Caso o make não esteja instalado em seu ambiente, instale-o com o comando: sudo apt install make.

Compile o Código: Utilizando o Makefile, digite o comando `make` para gerar os arquivos objetos (.o´s). Conforme a imagem a seguir:

```
mugen_italo@DESKTOP-G32OBPE:/mnt/c/Users/Italo/Desktop/BCD/ED1/Trabalho2-ed$ make clean
rm -f arvore.o lista.o main_funcoes.o main.o sistema_cineusp
mugen_italo@DESKTOP-G32OBPE:/mnt/c/Users/Italo/Desktop/BCD/ED1/Trabalho2-ed$ make
gcc -Wall -Wextra -g -c arvore.c -o arvore.o
gcc -Wall -Wextra -g -c lista.c -o lista.o
gcc -Wall -Wextra -g -c main_funcoes.c -o main_funcoes.o
gcc -Wall -Wextra -g -c main.c -o main.o
gcc -Wall -Wextra -g -o sistema_cineusp arvore.o lista.o main_funcoes.o main.o
mugen_italo@DESKTOP-G32OBPE:/mnt/c/Users/Italo/Desktop/BCD/ED1/Trabalho2-ed$
```

Esse comando compila o arquivo `main.c` e gera um “executável” chamado `sistema_cineusp`. Para limpar os arquivos objetos e o “executável” use `make clean`



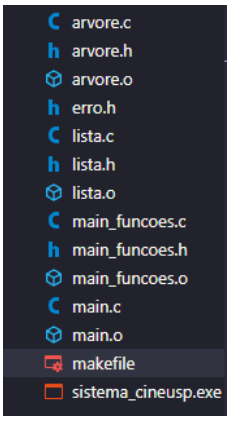
- **Ambiente Windows:**

Acesse o Diretório do Código-Fonte: Em um terminal, navegue até o diretório onde o código-fonte está salvo.

Compile o Código: Para compilar o código com o make é necessário ter o MinGW instalado, e então digite o comando `mingw32-make` para gerar os arquivos objetos (.o´s),

```
PS C:\Users\Italo\Desktop\BCD\ED1\Trabalho2-ed> mingw32-make
gcc -Wall -Wextra -g -c arvore.c -o arvore.o
gcc -Wall -Wextra -g -c lista.c -o lista.o
gcc -Wall -Wextra -g -c main_funcoes.c -o main_funcoes.o
gcc -Wall -Wextra -g -c main.c -o main.o
gcc -Wall -Wextra -g -o sistema_cineusp.exe arvore.o lista.o main_funcoes.o main.o
PS C:\Users\Italo\Desktop\BCD\ED1\Trabalho2-ed> mingw32-make clean
del /Q /F arvore.o lista.o main_funcoes.o main.o sistema_cineusp.exe
PS C:\Users\Italo\Desktop\BCD\ED1\Trabalho2-ed>
```

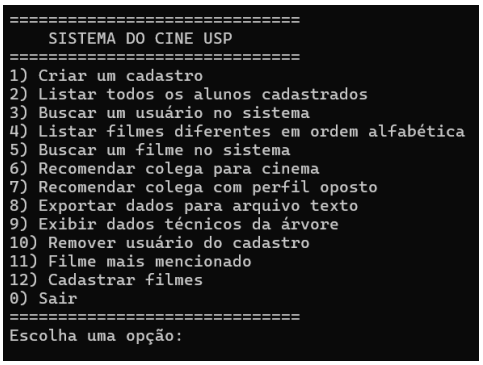
Esse comando compila o arquivo `main.c` e gera um executável chamado `sistema_cineusp.exe`. Para limpar os arquivos objetos e o executável use `mingw32-make clean`.



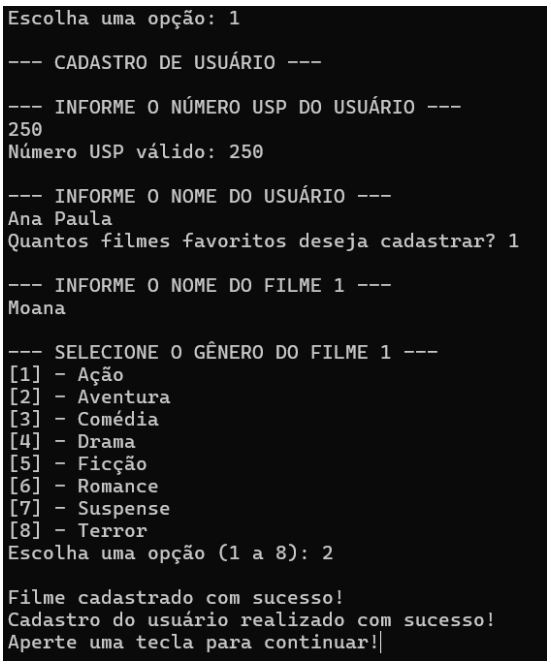
2.3. Execução do Programa

Executar o Programa:
Para iniciar o programa no Ubuntu (Linux), use o comando abaixo:
`./sistema_cineusp`
Para iniciar o programa no PowerShell (Windows), use o comando abaixo:
`./sistema_cineusp.exe`

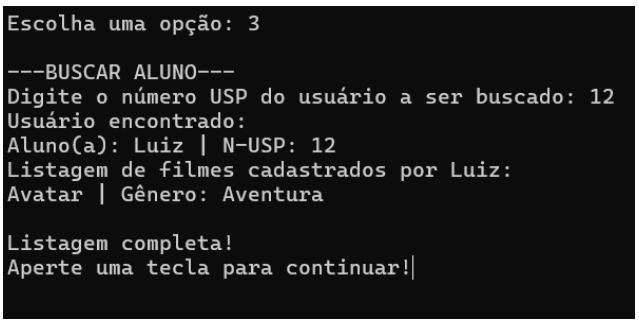
1. **Exemplo de Execução**: Após iniciar o programa, um menu principal é exibido na tela. Esse menu permite ao usuário selecionar opções como cadastrar um usuário, listar filmes ou alunos, cadastrar filmes, recomendar um colega mais similar e outros.

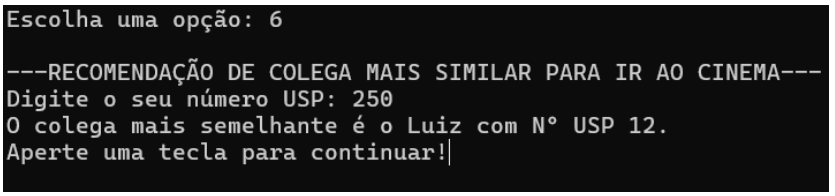


2. **Exemplo de Cadastro de Usuário**: Ao selecionar a opção "Criar um cadastro", o sistema solicitará o nome do aluno a ser inserido, como também o número USP, quantos filmes deseja cadastrar no ato do cadastro, nomes e gêneros dos filmes. Após inserir o dado, uma confirmação de cadastro é exibida.

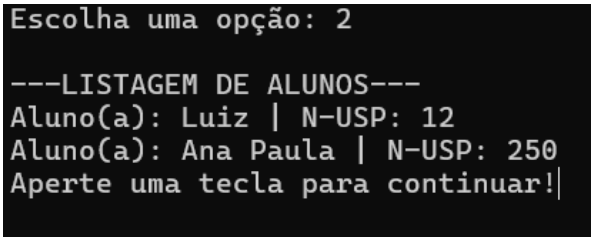


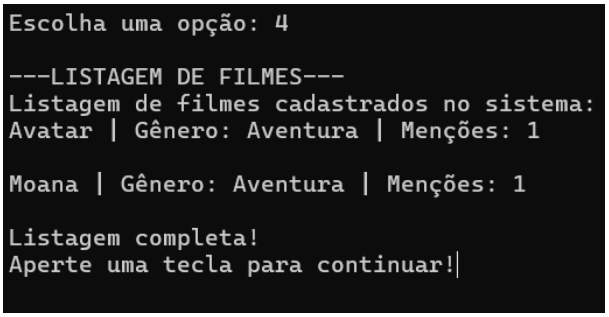
3. **Exemplo de Recomendar colega para cinema**: Essa funcionalidade permite que o usuário insira seu Nº USP e encontre o colega mais similar na árvore, com base na menor distância euclidiana. A distância é calculada utilizando o vetor que registra a quantidade de filmes cadastrados por gênero, comparando as preferências de ambos os usuários.

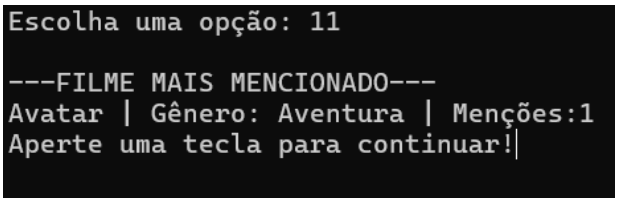




4. **Exemplo de Listagem de Alunos cadastrados, Filmes cadastrados, Filme mais mencionado e Exibir dados técnicos da árvore**: É possível listar todos os alunos cadastrados e seus respectivos Nº USP, Listar todos os filmes cadastrados no sistema em ordem alfabética, mostrar o Filme mais mencionado como favorito pelos usuários e exibir dados técnicos da árvore como Total de Nós, Altura da arvore, e diferença de altura entre subárvore direita e esquerda de algum nó específico.







OBS.: Quando todos os filmes têm a mesma quantidade de menções, o filme mais mencionado é o primeiro da lista ordenada de filmes cadastrados no sistema.

```
Escolha uma opção: 9

--- DADOS DA ÁRVORE ---

Total de Nós: 2
Altura da Árvore: 2
Digite o N-USP do nó que deseja verificar: 250
Maior diferença entre alturas que existe entre as sub-árvores do nó com N-USP 250: 1

---FIM EXIBIÇÃO DOS DADOS---
Aperte uma tecla para continuar!
```

5. **Exemplo de cadastro de filmes:** O sistema verifica inicialmente se o usuário existe por meio do Nº USP. Caso não exista, o usuário é orientado a realizar o cadastro. Se o usuário já estiver registrado, novos filmes podem ser adicionados à sua lista de favoritos. Para cada filme, o sistema verifica na lista geral de filmes da árvore: se o filme ainda não foi mencionado, ele é adicionado; caso já exista, apenas incrementa o contador que registra sua quantidade de menções.

```
---CADASTRO DE FILMES---

--- INFORME O NÚMERO USP DO USUÁRIO ---
250
Número USP válido: 250
Quantos filmes favoritos deseja cadastrar? 2

--- INFORME O NOME DO FILME 1 ---
Barbie

--- SELECIONE O GÊNERO DO FILME 1 ---
[1] - Ação
[2] - Aventura
[3] - Comédia
[4] - Drama
[5] - Ficção
[6] - Romance
[7] - Suspense
[8] - Terror
Escolha uma opção (1 a 8): 2

Filme cadastrado com sucesso!

--- INFORME O NOME DO FILME 2 ---
Verdade Nua e Crua

--- SELECIONE O GÊNERO DO FILME 2 ---
[1] - Ação
[2] - Aventura
[3] - Comédia
[4] - Drama
[5] - Ficção
[6] - Romance
[7] - Suspense
[8] - Terror
Escolha uma opção (1 a 8): 3

Filme cadastrado com sucesso!
Cadastro feito com sucesso.
Aperte uma tecla para continuar!
```

6. **Exemplo de encerramento de programa:** Encerra e sai do programa.