UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

CAMILA AYA SAITO (15635649)
VICTORIA FÁVERO NUNES (15698302)
VINICIUS MORAES DE CARVALHO (15642432)

Trabalho 2 de Estruturas de Dados 1

SÃO CARLOS
2024

INTRODUÇÃO

De acordo com a proposta e exigências do segundo trabalho da disciplina de Estrutura de Dados 1, o grupo desenvolveu um sistema de busca de alunos que gira em torno da aplicação de uma Árvore Balanceada de Busca (AVL), em que cada aluno tem uma lista de filmes favoritos. Nele, o usuário poderá utilizar as seguintes funcionalidades implementadas: cadastrar um aluno, listar os alunos cadastrados, buscar um aluno no sistema, listar todos os filmes citados, verificar se um filme foi citado, dar uma sugestão de colega que tenha preferências semelhantes ou distintas, gerar um arquivo texto (txt), imprimir os dados técnicos do sistema/árvore, remover um aluno, dizer qual o filme mais popular e sair do programa. Para a implementação do código, foi utilizado um Tipo Abstrato de Dados (TAD) de uma árvore de listas, com tudo encadeado dinamicamente, um arquivo principal (main) e um Makefile para compilar todos os códigos e rodar o programa.

ESTRUTURAS DE DADOS

```
typedef struct no_lista_arvore{
    char nome_filme[50];
    struct no_lista_arvore *prox;
}No_lista;
```

Na estrutura *no_lista_arvore*, nomeada como *No_lista*, que servirá para fazer a lista de filmes favoritos de cada aluno, há uma string (vetor de até 50 caracteres) *nome_filme*, que armazenará o nome do filme; e um ponteiro *prox* que apontará para uma *struct no_lista_arvore*, ou seja, para o próximo filme da lista.

```
typedef struct no_arvore{
    int numero_USP;
    char nome[50];
    struct no_arvore *esquerda, *direita;
    No_lista *lista_inicio, *lista_fim;
    int altura;
}No_arvore;
```

Na estrutura *no_arvore*, nomeada como *No_arvore*, que armazenará todas as informações de um nó da AVL de alunos, há uma variável do tipo inteiro *numero_USP* que guardará o número USP de um aluno; uma string de até 50 caracteres *nome* que armazenará o nome do aluno; dois ponteiros do tipo *struct no_arvore*, *esquerda* e *direita*, em que o primeiro apontará para o próximo nó à esquerda e o segundo para o próximo à direita; dois ponteiros do tipo *No_lista*, *lista_inicio*, que apontará para o início da lista de filmes do aluno cadastrado, e *lista_fim*, que apontará para o fim de tal lista; e uma variável *altura* do tipo inteiro, que armazenará o altura daquele nó.

```c
typedef struct no_lista_filme{
    char nome_filme[50];
    int contador;
    struct no_lista_filme *prox;
}No_lista_geral;
```

A estrutura *no_lista_filme*, nomeada como *No_lista_geral*, será a utilizada para representar os nós de uma lista apenas de filmes, separada da árvore. Ela contém uma string de até 50 caracteres *nome_filme*, que armazenará o nome de um filme; uma variável *contador* do tipo inteiro, que guardará quantas pessoas colocaram tal filme em suas listas de filmes favoritos na árvore; e um ponteiro *prox* do tipo *struct no_lista_filme,* que apontará para o próximo filme dessa lista.

```c
typedef struct l{
    No_lista_geral *inicio, *fim;
}lista;
```

A estrutura *lista* será a representação da lista de filmes que está separada da árvore para o usuário. Nela há dois ponteiros do tipo *No_lista_geral*, um *inicio*, que apontará para o início dessa lista, e um *fim*, que apontará para o fim.

```c
typedef struct Recomendacao{
    No_arvore *aluno;
    int count;
    struct Recomendacao *prox;
}Recomendacao;
```

Na estrutura *Recomendacao*, que servirá para construir uma lista de colegas que poderão ser recomendados, há um ponteiro *aluno* do tipo *No_arvore*, que apontará para o nó do aluno que poderá ser recomendado; uma variável do tipo inteiro *count*, que guardará quantos filmes iguais entre o aluno que pediu e o colega; e um ponteiro *prox* para outra *struct Recomendacao*, que apontará para o próximo colega da lista.

RECOMENDACAODIF ???

FUNÇÕES DO TAD

No_arvore* inserir: insere um nó (aluno) na árvore;
No_arvore* remover_aluno: remove e libera a memória de um nó (aluno) da árvore;

int altura: retorna a altura de um dado nó;
int quantidade_no: retorna a quantidade de nós existentes na árvore;
int maior_diferenca: retorna a maior diferença entre as alturas dos nós;
int criar_lista: cria uma lista de filmes L;
int inserir_no_lista_geral: insere um nó (filme) na lista geral de filmes;
int inserir_no_lista_arvore: insere um nó (filme) na lista de filmes acoplada à um aluno da árvore;
No_lista_geral* buscar_lista: busca um determinado filme na lista geral (separada da árvore) de filmes a partir do nome do filme;
Recomendacao* obter_recomendacao_similar: dado um aluno, através do número USP, encontra-se um colega que tenha preferências similares ao que pediu, ou seja, retorna-se o aluno que tenha a maior quantidade de filmes favoritos iguais aos do que pediu em sua lista;
Recomendacao* obter_proxima_recomendacao: retorna-se o próximo aluno da lista de recomendações;
void liberar_arvore: esvazia e libera toda a memória da árvore;
void arquivo: escreve todos as informações contidas na árvore em um dado arquivo;
int obter_numero_USP: retorna o número USP de um aluno;
void obter_nome_aluno: retorna o nome de um aluno;
void obter_nome_filme: retorna o nome de um filme;
void filmes_mais_populares: retorna os filmes mais populares até o momento, ou seja, os filmes que mais apareceram nas listas de favoritos dos alunos;

FUNCIONALIDADES DO PROGRAMA

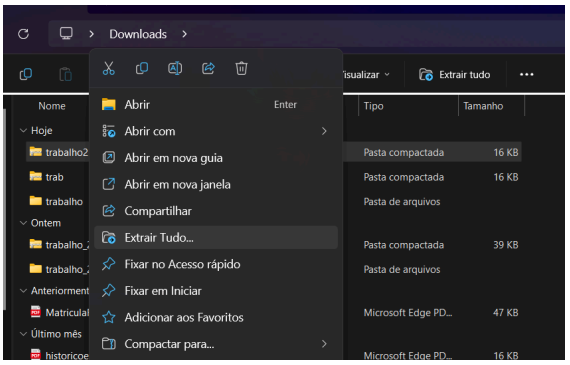
1) **Cadastrar um aluno:** cadastra um aluno no sistema, com todas as informações necessárias, assim como sua lista de filmes favoritos.
2) **Listar alunos:** lista todos os alunos cadastrados no sistema.
3) **Buscar aluno:** informa se um aluno está ou não cadastrado no sistema.
4) **Listar filmes:** lista todos os filmes já citados pelos alunos.
5) **Buscar filme:** informa se um filme foi citado ou não por algum aluno.
6) **Recomendação:** recomenda um colega com gostos semelhantes.
7) **Gerar arquivo texto:** gera um arquivo texto (.txt), que terá todas as informações da árvore.
8) **Dados Técnicos:** revela todos os dados técnicos da árvore implementada, como altura da árvore, maior diferença de alturas, e a quantidade de nós existentes.
9) **Remover aluno:** remove um aluno do sistema, assim como sua lista de filmes favoritos.
10) **Listar filme mais popular:** informa o filme mais popular do momento, ou seja, o filme que mais foi marcado como favorito entre os alunos.

0) **Sair:** sai do programa e gera um arquivo texto (.txt) com todas as informações que foram registradas.

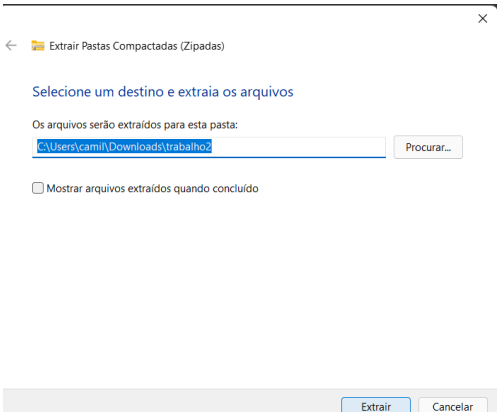
COMPILAÇÃO DO PROGRAMA

Para compilar o programa, foi utilizado o MinGW (compilador para sistemas Windows conseguirem usar o GCC) versão 6.3.0, no sistema operacional Windows. No programa, foram utilizadas as bibliotecas <stdio.h>, <stdlib.h> e <string.h>, que são bibliotecas padrões do C, além do TAD <*trabalho.h*> implementado.

Primeiramente, é necessário extrair os arquivos da pasta compactada. Para isso, clique com o botão direito sobre a pasta e selecione a opção *Extrair tudo*.

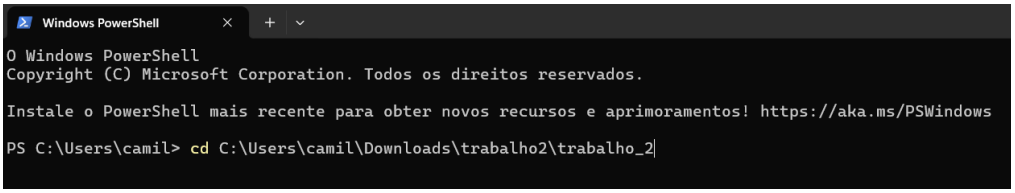


Após isso, selecione o local para onde deseja instalar a pasta com os arquivos extraídos e depois clique em *Extrair*.

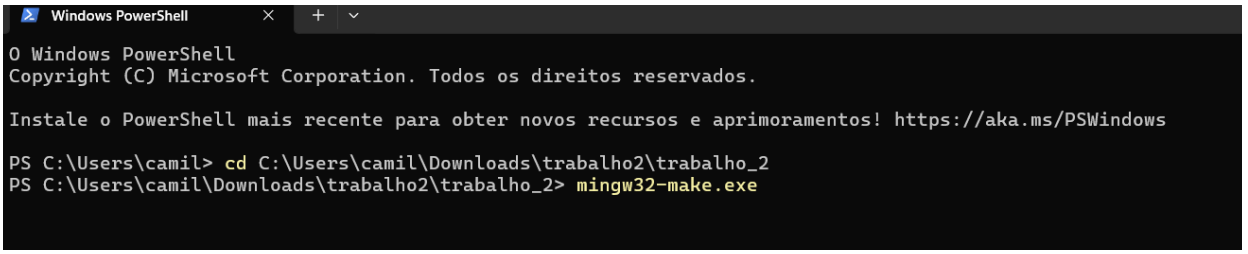


Com isso, os arquivos estarão extraídos e prontos para serem utilizados.

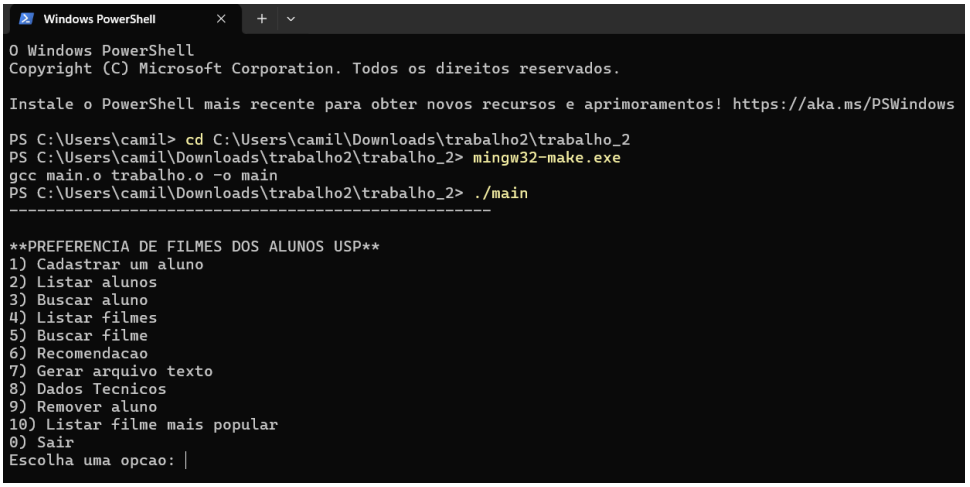
Para executar o programa, é necessário ter o MinGW instalado no sistema, abrir o prompt de comando e acessar, por linha de comando, a pasta (com os arquivos já extraídos) em que os arquivos estão (*cd 'caminho da pasta'*). No exemplo, os arquivos estão na pasta *C:\Users\camil\Downloads\ED1_leilao*, então o comando será *cd C:\Users\camil\Downloads\ED1_leilao*.



Depois, é necessário compilar os arquivos. Para isso, utilize o comando *mingw32-make.exe*, que, através do *Makefile*, irá compilar todos os arquivos necessários se uma só vez.



Por fim, para executar o programa, digite o comando *./main*, que irá fazer o programa rodar.



E caso se queira limpar o prompt de comando, digita-se o comando *cls*.