

UNIVERSIDADE FEDERAL DO PARANÁ

GUILHERME ARTHUR NUNES MENEGARI

LEONARDO CHICORA NETO

SOPHIA SOARES COSTA E SILVA

**RECOMENDAÇÃO DE FILMES**

CURITIBA

2024

GUILHERME ARTHUR NUNES MENEGARI  
LEONARDO CHICORA NETO  
SOPHIA SOARES COSTA E SILVA

### **RECOMENDAÇÃO DE FILMES**

TRABALHO APRESENTADO À DISCIPLINA:  
LINGUAGEM PROGRAMAÇÃO (DS123)  
SETOR DE EDUCAÇÃO PROFISSIONAL E  
TECNOLÓGICA, UNIVERSIDADE FEDERAL DO PARANÁ  
PROF. DR. ROBERSON CESAR ALVES DE ARAUJO

# SUMÁRIO

<b>SUMÁRIO.....</b>	<b>3</b>
<b>PREFÁCIO.....</b>	<b>4</b>
<b>1. INTRODUÇÃO.....</b>	<b>5</b>
1.1 TEMA.....	5
1.2 OBJETIVO DO PROJETO.....	5
1.3 DELIMITAÇÃO DO PROBLEMA.....	5
1.4 JUSTIFICATIVA PARA A ESCOLHA DO TEMA.....	6
1.5 GLOSSÁRIO.....	6
<b>2. DESCRIÇÃO GERAL DO SISTEMA.....</b>	<b>8</b>
2.1 DESCRIÇÃO DO PROBLEMA.....	8
2.2 PRINCIPAIS ENVOLVIDOS E SUAS CARACTERÍSTICAS.....	8
2.2.1 USUÁRIO DO SISTEMA.....	9
2.2.2 DESENVOLVEDORES DO SISTEMA.....	9
2.3 REGRAS DE NEGÓCIO.....	9
<b>3. REQUISITOS DO SISTEMA.....</b>	<b>10</b>
3.1 REQUISITOS FUNCIONAIS.....	10
3.2 REQUISITOS NÃO FUNCIONAIS.....	13
3.3 REQUISITOS DE IMPLEMENTAÇÃO.....	15
<b>4. ARQUITETURA DO SISTEMA.....</b>	<b>16</b>
4.1 DIAGRAMA DE CASO DE USO.....	18
4.2 AMBIENTE DE DESENVOLVIMENTO.....	18
<b>5. IMPLEMENTAÇÃO.....</b>	<b>20</b>
5.1 BIBLIOTECAS.....	20
5.2 CÓDIGO COMENTADO.....	20
<b>6. TESTES.....</b>	<b>31</b>
<b>7. MANUAL DO USUÁRIO.....</b>	<b>34</b>
<b>8. MAPA DOS MENUS.....</b>	<b>36</b>
<b>9. CONSIDERAÇÕES FINAIS.....</b>	<b>37</b>

## PREFÁCIO

Este trabalho é uma tentativa de aplicar os conceitos teóricos aprendidos em sala de aula em um cenário prático, permitindo uma compreensão mais profunda e intuitiva da linguagem de programação C. Através deste projeto, buscamos não apenas demonstrar a aplicação dos conceitos de programação, mas também enfatizar a importância do pensamento lógico, da resolução de problemas e da eficiência no código.

A linguagem C, sendo uma das linguagens de programação mais populares e amplamente utilizadas, oferece uma excelente plataforma para explorar esses conceitos. Com sua sintaxe direta e a capacidade de interagir diretamente com o hardware, a linguagem C nos permite entender melhor como os programas são executados em um nível mais baixo.

Esperamos que este documento sirva como um recurso valioso para todos aqueles que desejam aprofundar seus conhecimentos em programação e, especificamente, na linguagem C. Seja você um estudante iniciante em programação ou um desenvolvedor experiente procurando revisar e aprimorar suas habilidades, acreditamos que este trabalho tem algo a oferecer.

Finalmente, gostaríamos de expressar nossa gratidão aos professores e colegas que contribuíram para a realização deste projeto. Seu apoio e orientação foram inestimáveis. Esperamos que este trabalho atenda às suas expectativas e seja um testemunho do nosso aprendizado e crescimento nesta disciplina.

# **1. INTRODUÇÃO**

No mundo digital de hoje, a personalização é a chave para melhorar a experiência do usuário. Uma área onde a personalização tem um impacto significativo é a indústria do entretenimento, especificamente na recomendação de filmes. Este documento se concentra em um software que recomenda filmes aos usuários com base em seus gostos e escolhas anteriores, utilizando a linguagem de programação C.

Neste trabalho, vamos explorar o desenvolvimento de um sistema de recomendação de filmes. Começaremos com uma visão geral do sistema, explicando como ele funciona e por que a personalização é tão importante na recomendação de filmes. Em seguida, discutiremos os conceitos fundamentais por trás do sistema, como a coleta de dados do usuário, a análise desses dados e a geração de recomendações personalizadas.

Além disso, vamos mergulhar nos aspectos técnicos do desenvolvimento do software, incluindo a estrutura do código, o gerenciamento de dados e a implementação de algoritmos de recomendação. Cada seção será acompanhada por exemplos de código em C para ilustrar os conceitos discutidos.

## **1.1 TEMA**

O projeto consiste na criação de um programa, desenvolvido em linguagem C, que recomenda filmes ao usuário com base em suas preferências. O tema foi escolhido após um debate entre os alunos responsáveis pelo desenvolvimento do programa. Considerando a vasta gama de opções de entretenimento disponíveis nos tempos atuais, muitas vezes surge a dúvida sobre o que assistir. Portanto, o objetivo deste programa é auxiliar o usuário na escolha de filmes.

## **1.2 OBJETIVO DO PROJETO**

O projeto foi pensado em sanar a dúvida do que assistir na hora de escolher um filme, uma vez que na era digital existem tantas opções disponíveis e meios de se assistir a um filme, sendo assim é que foi criado um programa capaz de auxiliar o usuário nessa questão.

### 1.3 DELIMITAÇÃO DO PROBLEMA

A delimitação do problema deste projeto reside na necessidade de desenvolver um algoritmo eficiente que possa fornecer recomendações de filmes personalizadas para os usuários. O algoritmo deve ser capaz de analisar as preferências do usuário e, com base nisso, sugerir filmes que se alinhem a essas preferências.

No entanto, a delimitação do problema não termina aqui. O algoritmo deve ser capaz de aprender e adaptar-se às mudanças nas preferências do usuário ao longo do tempo. Além disso, deve ser capaz de lidar com uma grande quantidade de dados, pois o número de filmes disponíveis é vasto e está sempre crescendo.

Outra delimitação do problema é a necessidade de garantir que o programa seja fácil de usar. O usuário deve ser capaz de interagir facilmente com o programa e entender as recomendações fornecidas.

### 1.4 JUSTIFICATIVA PARA A ESCOLHA DO TEMA

Com base em temas debatidos com o grupo acadêmico que desenvolveu o projeto, foi optado pela linguagem C para se criar o código, por ser uma linguagem de baixo nível e ter diversas funcionalidades para aplicações se torna perfeito para o desenvolvimento do programa.

Optando por um tema que pudesse ajudar usuários comuns em suas rotinas cotidianas é pensando num programa de indicação de filmes. Nos dias atuais com os inúmeros aplicativos de filmes e opções disponíveis no cinema, tal programa visa poder ajudar a realizar a escolha do que assistir em um momento de lazer, além de poder organizar filmes assistidos e pretensões futuras de filmes.

### 1.5 GLOSSÁRIO

TERMOS GERAIS	DESCRIÇÃO
---------------	-----------

<b>LINGUAGEM</b>	É um conjunto de regras sintáticas que se comunicam com o computador e com isso podem montar um programa.
<b>SOFTWARE</b>	Conjunto de instruções escritas e interpretadas por um computador para que uma determinada tarefa seja executada.
<b>REQUISITOS</b>	São características ou funcionalidades que o sistema deve apresentar para ser considerado funcional.
<b>DESEMPENHO</b>	Velocidade com que o programa retorna aos comandos inseridos
<b>ESCALABILIDADE</b>	Capacidade de inserção de dados sem comprometer o desempenho do sistema.
<b>BIBLIOTECAS</b>	São arquivos que ajudam a organizar o código ou resolver problemas dentro dele.
<b>DIRETÓRIO</b>	É uma estrutura usada para organizar arquivos em um computador.
<b>ARQUITETURA DE SOFTWARE</b>	É a estrutura de um sistema de software, definindo seus componentes, suas relações e seus princípios de projeto e evolução.

## **2. DESCRIÇÃO GERAL DO SISTEMA**

O sistema consiste em um programa feito na linguagem C, dispõem de opções que indicam filmes ao usuário com base em critérios pré-selecionados anteriormente para realizar a indicação que seja compatível com o gosto pessoal de quem o usa. Além da função principal tem algumas outras funções secundárias onde pode favoritar os filmes que já assistiu, marcar filmes que tem pretensão de assistir futuramente dentre outras funcionalidades que são disponibilizadas.

### **2.1 DESCRIÇÃO DO PROBLEMA**

O desafio que buscamos resolver é o tempo na tomada de decisão para a escolha de um filme, considerando a vasta quantidade de opções disponíveis atualmente.

- Quem é afetado pelo sistema?

O sistema tem como alvo qualquer pessoa que enfrenta dificuldades para selecionar um filme dentre as inúmeras opções disponíveis.

- Qual o impacto do sistema?

O sistema visa reduzir significativamente o tempo gasto na escolha de filmes, apresentando opções que se alinham aos gostos do usuário. Além disso, o sistema oferece a funcionalidade de organizar o conteúdo assistido pelo usuário, permitindo a criação de uma lista personalizada de filmes que o usuário tem interesse em assistir e dos que já foram assistidos.

- Qual seria uma boa solução para o problema?

A solução mais viável para o problema seria o desenvolvimento de um programa que auxilie o usuário na escolha de filmes. Este programa, além de sua funcionalidade principal de sugerir filmes baseados nos gostos do usuário, também ofereceria recursos adicionais que ajudariam o usuário a organizar sua programação de filmes. Isso incluiria a capacidade de criar listas personalizadas de filmes para assistir no futuro e um histórico dos filmes já assistidos.



## **2.2 PRINCIPAIS ENVOLVIDOS E SUAS CARACTERÍSTICAS**

### **2.2.1 USUÁRIO DO SISTEMA**

O sistema se direciona a qualquer pessoa que tenha interesse em obter um auxílio para fazer a escolha de um filme. Não existe uma restrição de público para quem o sistema foi desenvolvido, a intenção é abranger o máximo de pessoas possíveis.

### **2.2.2 DESENVOLVEDORES DO SISTEMA**

Os desenvolvedores do sistema são estudantes do curso de Análise e Desenvolvimento de Sistemas, oferecido pela Universidade Federal do Paraná, tais alunos possuem habilidades e conhecimento da Linguagem C, assim como domínio de algumas bibliotecas para que fosse possível a criação do sistema.

Equipe:

Guilherme Arthur Nunes Menegari

Leonardo Chicora Neto

Sophia Silva

## **2.3 REGRAS DE NEGÓCIO**

- Como regra de negócio fora estabelecido que não pode existir dois usuários com o mesmo login, ou seja, nomes iguais o sistema não vai permitir a criação.

### 3. REQUISITOS DO SISTEMA

Requisitos são as condições ou capacidades necessárias que um sistema ou programa deve ter para satisfazer as necessidades do cliente, incluindo aspectos funcionais e não funcionais, e podem ser especificados em vários níveis de detalhe e classificados em diferentes tipos, dependendo do contexto do projeto.

Além de ter aspectos funcionais e não funcionais, existe uma separação de prioridade dos requisitos, são eles:

- **Alta:** são obrigatórios estarem no sistema para que atenda às demandas solicitadas pelo cliente.
- **Média:** não são em sua totalidade essenciais para o funcionamento do sistema, porém garante parâmetros de confiabilidade e melhorias.
- **Baixa:** são requisitos que não são necessários para que o sistema funcione, porém acrescentam funcionalidades que podem oferecer um conforto a mais na experiência para o cliente final.

#### 3.1 REQUISITOS FUNCIONAIS

Um requisito funcional é uma característica ou funcionalidade que o software deve atender às requisições do cliente. São tais requisições que modelam o sistema e criam suas funcionalidades.

Requisito: RF1 Cadastro do usuário	
Prioridade	Alta
Descrição	O sistema deve permitir que o usuário se cadastre fornecendo um nome de usuário e uma senha. O sistema deve solicitar que o usuário confirme a senha para garantir que não haja erros de digitação.

Requisito: RF2 Login do Usuário	
Prioridade	Alta

<b>Descrição</b>	O sistema deve fornecer uma opção para que o usuário faça login usando o nome de usuário e a senha fornecidos durante o cadastro. Isso permitirá que o usuário acesse suas recomendações personalizadas
------------------	---

<b>Requisito: RF3 Entrada do Usuário</b>	
<b>Prioridade</b>	Alta
<b>Descrição</b>	O sistema deve ser capaz de coletar as preferências do usuário, como filmes. Essas preferências serão usadas para personalizar as recomendações de filmes para o usuário.

<b>Requisito: RF4 Recomendação de Filmes</b>	
<b>Prioridade</b>	Alta
<b>Descrição</b>	Com base nas preferências do usuário, o sistema deve ser capaz de recomendar filmes que o usuário possa gostar. As recomendações devem ser personalizadas para cada usuário.

<b>Requisito: RF5 Atualização de preferências</b>	
<b>Prioridade</b>	Média
<b>Descrição</b>	O sistema deve ser capaz de atualizar as preferências do usuário com base em suas avaliações recentes. Isso permitirá que o sistema continue a fornecer recomendações relevantes à medida que os gostos do usuário mudam.

<b>Requisito: RF6 Não recomendar filmes já inseridos</b>	
<b>Prioridade</b>	Média

<b>Descrição</b>	O sistema deve manter um registro dos filmes que o usuário marcou como assistidos. Isso pode ser usado para evitar recomendar filmes que o usuário já viu.
------------------	--

<b>Requisito: RF7 Cálculo e Porcentagem</b>	
<b>Prioridade</b>	Alta
<b>Descrição</b>	O sistema deve ser capaz de calcular a porcentagem de correspondência entre as preferências do usuário e os atributos de um filme. Isso será usado para determinar quão bem um filme corresponde às preferências do usuário.

<b>Requisito: RF8 Ordenação de Recomendações</b>	
<b>Prioridade</b>	Alta
<b>Descrição</b>	O sistema deve ser capaz de ordenar as recomendações com base nas porcentagens de correspondência calculadas, apresentando primeiro os filmes com maior correspondência.

<b>Requisito: RF9 Pesquisar Filmes</b>	
<b>Prioridade</b>	Média
<b>Descrição</b>	O sistema deve permitir que os usuários pesquisem filmes na base de dados. Isso permitirá que os usuários encontrem filmes específicos que desejam assistir.

<b>Requisito: RF10 Lista de Desejos</b>	
<b>Prioridade</b>	Baixa

<b>Descrição</b>	O sistema deve permitir que os usuários criem uma lista de filmes que desejam assistir no futuro. Isso permitirá que os usuários mantenham um registro dos filmes que estão interessados em assistir.
------------------	---

<b>Requisito: RF11 Opção de “Onde Estou”</b>	
<b>Prioridade</b>	Baixa
<b>Descrição</b>	O sistema deve fornecer uma opção para que o usuário se localize entre os menus. Para impedir confusões.

### 3.2 REQUISITOS NÃO FUNCIONAIS

Requisitos não funcionais definem funcionalidades e propriedades do sistema desenvolvido, ou seja, pode-se dizer que são funções que não interferem no desenvolvimento mas ajudam na confiabilidade e segurança do sistema, sendo assim, um aumento da qualidade do sistema.

Requisito: RNF1 Desempenho	
Prioridade	Alta
Descrição	O sistema deve ser capaz de fornecer recomendações em um tempo aceitável. Isso garantirá que os usuários não tenham que esperar muito tempo para receber suas recomendações.

Requisito: RNF2 Escalabilidade	
Prioridade	Média
Descrição	O sistema deve ser capaz de lidar com um grande número de filmes. Isso garantirá que o sistema possa continuar a fornecer recomendações precisas à medida que a base de dados de filmes cresce.

Requisito: RNF3 Precisão	
Prioridade	Alta
Descrição	As porcentagens calculadas devem ser precisas para garantir que as recomendações sejam relevantes para o usuário. Isso garantirá que os usuários recebam recomendações que correspondam às suas preferências.

Requisito: RNF4 Usabilidade	
<b>Prioridade</b>	Alta
<b>Descrição</b>	O sistema deve ser fácil de usar. A interface do usuário deve ser intuitiva e agradável. Isso garantirá que os usuários possam navegar facilmente pelo sistema e encontrar o que estão procurando.

### 3.3 REQUISITOS DE IMPLEMENTAÇÃO

O requisito de implementação é como o sistema vai ser implementado, sendo assim é uma condição para o projeto. Ele deve ser seguido estritamente para que a taxa de sucesso da implementação seja alcançada.

Requisito: RI1 Linguagem de Programação	
Prioridade	Alta
Descrição	O sistema deve ser implementado na linguagem de programação C. Isso se deve ao fato de ser a linguagem requisitada para o projeto.

Requisito: RI2 Base de Dados	
Prioridade	Alta
Descrição	O sistema deve utilizar uma base de dados para armazenar informações sobre os usuários, suas preferências e os filmes. Isso permitirá que o sistema mantenha um registro persistente dessas informações e as recupere quando necessário.



## 4. ARQUITETURA DO SISTEMA

O sistema foi criado em criado com duas camadas, sendo ela a própria criação em C e a segunda camada em armazenamento. Optado por fazer apenas com duas camadas no intuito de ter um programa leve, capaz de rodar sem problemas de erro durante sua execução.

Dentro da arquitetura do código existem algumas funções que permitem que o código seja funcional.

### **Sistema de Login e Cadastro de Usuários:**

- Módulo de Cadastro (cadastrar()): Responsável por receber os dados de um novo usuário (login e senha), verificar se o usuário já existe no arquivo de usuários e, caso contrário, armazenar as informações em um arquivo binário.
- Módulo de Login (login()): Recebe os dados de login de um usuário, verifica se as credenciais correspondem aos registros existentes e autentica o usuário se as credenciais forem válidas.

### **Manipulação de Senha:**

- Funções getSenha(): Fornece métodos para receber a senha do usuário sem mostrar os caracteres digitados, garantindo assim maior segurança.

### **Sistema de Recomendação de Filmes:**

- Inserção de Filmes Favoritos (inserirFilmesFavoritos()): Permite ao usuário inserir seus filmes favoritos, consultando-os em um arquivo de filmes e armazenando-os em arrays de filmes favoritos e gêneros favoritos.
- Busca de Filmes (buscarFilmes()): Permite ao usuário buscar filmes por título ou gênero, consultando um arquivo de filmes e exibindo os resultados correspondentes.
- Listagem de Filmes (listarFilmes()): Lista todos os filmes disponíveis, consultando um arquivo de filmes e exibindo os títulos.
- Recomendação de Filmes (recomendarFilmes()): Utiliza os filmes favoritos do usuário e seus gêneros favoritos para recomendar outros filmes com base na reincidência de gêneros.

**Funções Auxiliares:**

- Limpar Tela (limparTela()): Fornece métodos para limpar a tela do console, melhorando a experiência do usuário.
- Onde Estou (ondeEstou()): Exibe ao usuário sua localização atual no sistema de menu, ajudando na navegação.

**Função Principal (main()):**

- Menu Principal: Exibe opções para o usuário escolher entre cadastrar, fazer login, buscar filmes, listar filmes, sair, entre outras.
- Controle de Fluxo: Gerencia o fluxo de execução do programa, direcionando o usuário para as funcionalidades selecionadas.

**Interação:**

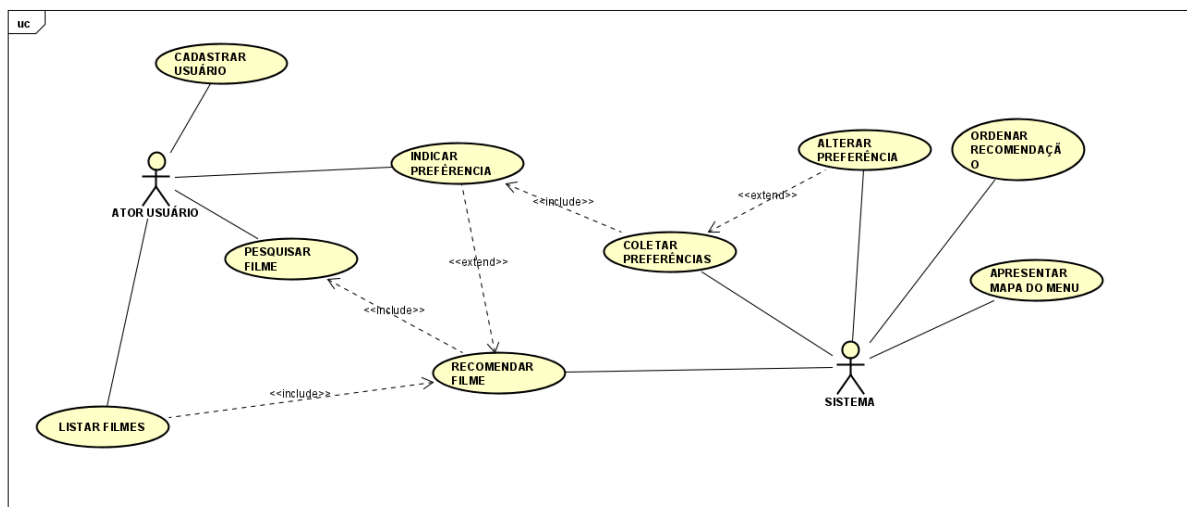
- O usuário interage com o sistema através do menu principal, escolhendo opções como cadastrar, fazer login, buscar filmes, etc.
- As funções de cadastro e login manipulam os arquivos de usuários para armazenar e verificar as credenciais.
- O sistema de recomendação de filmes consulta um arquivo de filmes para realizar operações como inserir filmes favoritos, buscar filmes por título ou gênero e recomendar filmes com base nos favoritos do usuário.

## 4.1 DIAGRAMA DE CASO DE USO

Os diagramas de caso de uso, uma ferramenta da Linguagem de Modelagem Unificada (UML), são projetados para definir as funcionalidades que um sistema deve possuir e como ele interage com os usuários.

Os componentes são:

- **Atores:** Simbolizam os usuários ou entidades externas que interagem com o sistema. Eles são ilustrados como figuras humanóides.
- **Casos de Uso:** Ilustram as funcionalidades do sistema do ponto de vista do usuário. Eles são representados por formas ovais ou retangulares.
- **Associações:** Indicam a relação entre atores e casos de uso. São representadas por linhas contínuas.



## 4.2 AMBIENTE DE DESENVOLVIMENTO

Para desenvolvimento do código usando a Linguagem em C foi usado o programa Visual Studio Code, criado pela Microsoft, tal programa tem por sua finalidade oferecer um ambiente de desenvolvimento para a criação de softwares e também frameworks. Por ter várias linguagens de programação e podendo adicionar extensões das quais auxiliam melhor no trabalho foi optado por desenvolver nesta plataforma.

Além disso, para que o programa possa salvar e armazenar as informações inseridas, foi optado por usar um arquivo de texto para que pudesse servir como banco de dados. Tal função se torna simples e demonstra a capacidade dos programadores envolvidos em relação a manipulação de dados e arquivos.

## 5. IMPLEMENTAÇÃO

A implementação é uma fase crucial em qualquer projeto de desenvolvimento de sistemas. É nesta etapa que as ideias e conceitos se transformam, e com isso é modelado e escrito o código de acordo com o que se faz necessário para o ser produto funcional. A implementação envolve várias atividades, incluindo a codificação do sistema, testes, integração de diferentes componentes e, finalmente, a implantação do sistema.

No contexto do nosso sistema de recomendação de filmes, a implementação envolverá o desenvolvimento de algoritmos que podem analisar os gostos dos usuários e sugerir filmes relevantes. Além disso, também precisaremos implementar funcionalidades que permitam aos usuários organizar o conteúdo assistido e criar listas personalizadas de filmes.

### 5.1 BIBLIOTECAS

Para que fosse possível o desenvolvimento do código, foi utilizado um total de 4 bibliotecas. As bibliotecas nada mais são que rotinas usadas para auxiliar na entrada e saída de dados.

Abaixo estão listadas as bibliotecas usadas no desenvolvimento e as suas funcionalidades:

- **stdio.h:** permite que o programa tenha entrada e saída de dados.
- **stdlib.h:** com essa biblioteca permite o gerenciamento de memória, operações aritméticas com ponto flutuante dentre outras funcionalidades.
- **string.h:** é comumente usada para a manipulação de string.
- **stdbool.h:** permite que se possa utilizar expressões booleanas, com isso se elimina a necessidade de usar números inteiros para representar verdadeiro ou falso dentro da linha de código.
- **ctype.h:** fornece a manipulação de caracteres
- **conio.h:** originalmente não é pertencente a biblioteca de C, mas do MS-DOS. É utilizada para que o programa tenha função de escrita e leitura.

- **termios.h**: uma biblioteca criada para que se tenha entrada e saída de dados em Linux.
- **unistd.h**: tem como função fornecer a API do sistema operacional POSIX.

## 5.2 CÓDIGO COMENTADO

Com as bibliotecas declaradas no código pode-se então explicar como o código foi escrito. Para melhor explicação cada parte é explicada, além de se ter comentários dentro do próprio código.

Como cabeçalho e início do código foi definido quais bibliotecas seriam necessárias para o funcionamento do código. Além disso, também foi definido as estruturas de login e de armazenamento de informações do usuário, como o seu nome e a sua senha.

```
#include <stdio.h> // Inclusão de biblioteca padrão de entrada e saída
#include <stdlib.h> // Inclusão de biblioteca padrão de funções
#include <string.h> // Inclusão de biblioteca para manipulação de strings
#include <stdbool.h> // Inclusão de biblioteca para uso de variáveis booleanas
#include <ctype.h> // Inclusão de biblioteca para manipulação de caracteres

// Definições e estruturas para o sistema de login
#define LOGIN_MAX 64
#define SENHA_MAX 32
#define MAX_TENTATIVAS 3

// Definição da estrutura para armazenar informações de usuário
typedef struct {
    char login[LOGIN_MAX];
    char senha[SENHA_MAX];
} Usuario;
```

```

// Funções para manipulação de senha, dependendo do sistema operacional
#if defined(_WIN32) || defined(_WIN64)
#include <conio.h>

// Função para receber a senha do usuário sem mostrar os caracteres digitados
void getSenha(char *senha) {
    int i = 0;
    char ch;
    while ((ch = getch()) != '\r' && i < SENHA_MAX - 1) {
        if (ch == '\b' && i > 0) { // Backspace
            printf("\b \b");
            i--;
        } else if (ch != '\b') {
            senha[i++] = ch;
            printf("*");
        }
    }
    senha[i] = '\0';
    printf("\n");
}

```

O programa foi construído pensando em poder contemplar o maior número de usuários possíveis, por esse motivo foi inserida duas bibliotecas para tornar viável sua execução em Linux.

```

// Unix
#else
#include <termios.h>
#include <unistd.h>

```

Para aumentar a segurança do programa foi implementado uma sequência de código para que o programa receba a entrada da senha sem mostrar ela, ou seja, vai ocultar a senha.

```
// Função para receber a senha do usuário sem mostrar os caracteres digitados
void getSenha(char *senha) {
    struct termios oldt, newt;
    int ch;
    int i = 0;

    tcgetattr(STDIN_FILENO, &oldt);
    newt = oldt;
    newt.c_lflag &= ~(ECHO);
    tcsetattr(STDIN_FILENO, TCSANOW, &newt);

    while ((ch = getchar()) != '\n' && i < SENHA_MAX - 1) {
        senha[i++] = ch;
    }
    senha[i] = '\0';

    tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
    printf("\n");
}
#endif
```

Feito uma linha de código para criação do usuário e a validação da existencia do mesmo dentro do arquivo do programa



```
// Função para cadastrar um novo usuário
void cadastrar() {

    FILE *arquivo;
    Usuario usuario, comp;
    bool existe = false;
    limparTela();
    printf("Cadastro de usuario:\n");
    printf("Usuario: ");
    scanf("%63s", usuario.login);
    printf("Senha: ");
    getSenha(usuario.senha);

    arquivo = fopen("usuarios.bin", "rb+");
    if (!arquivo) {
        arquivo = fopen("usuarios.bin", "wb");
    } else {
        // Verifica se o usuário já existe no arquivo
        while(fread(&comp, sizeof(Usuario), 1, arquivo)) {
            if(strcmp(usuario.login, comp.login) == 0) {
                printf("Este usuario ja esta cadastrado! Por favor, faca login.\n");
                existe = true;
                break;
            }
        }
    }
}
```

```
// Se o usuário não existe, ele é registrado no arquivo
if (!existe) {
    fseek(arquivo, 0, SEEK_END); // Move o ponteiro para o final do arquivo para adicionar o novo usuário
    fwrite(&usuario, sizeof(Usuario), 1, arquivo);
    printf("Usuario cadastrado com sucesso!\n");
}

fclose(arquivo);
}
```

Como o sistema depende da criação de um usuário, é necessário que após a criação do mesmo faça login dentro do sistema para fazer uso das suas funcionalidades.

```
// Função para realizar o login
bool login() {

    FILE *arquivo;
    Usuario usuario;
    char login[LOGIN_MAX];
    char senha[SENHA_MAX];
    int tentativas = 0;

    arquivo = fopen("usuarios.bin", "rb");
    if (!arquivo) {
        printf("Nenhum usuario cadastrado, por favor cadastre pelo menos um usuario.\n");
        return false;
    }
}
```

Pensando em otimizar a segurança do usuário é inserido um número de tentativas para que ele possa fazer o login, caso não consiga realizar o login após o número determinado de tentativas o sistema volta com uma mensagem para o usuário.

```
// Loop para permitir várias tentativas de login
while (tentativas < MAX_TENTATIVAS) {
    printf("Usuario: ");
    scanf("%63s", login);
    printf("Senha: ");
    getSenha(senha);

    rewind(arquivo);
    // Verifica se o login e senha correspondem aos registros
    while (fread(&usuario, sizeof(Usuario), 1, arquivo)) {
        if (!strcmp(login, usuario.login) && !strcmp(senha, usuario.senha)) {
            fclose(arquivo);
            printf("Bem-vindo %s\n", usuario.login);
            return true;
        }
    }

    printf("\nUsuario ou senha invalidos! Tentativas restantes: %d\n", MAX_TENTATIVAS - (tentativas + 1));
    tentativas++;
}
```

Como é declarada as variáveis dos filmes para que o sistema possa recomendar os filmes.

```
#define NUM_FILMES_RECOMENDADOS 15
#define MAX_LINHA 200
#define MAX_FILMES 100 // Definição de MAX_FILMES

// Definições e estruturas para o sistema de recomendação de filmes
typedef struct {
    char title[100];
    char genres[100];
} Filme;
```

Função criada para a recomendação de filmes.

```
// Funções para recomendação de filmes
void recomendarFilmes(const char filmesFavoritos[][100], const char genresFavoritos[][100], Filme filmes[], int numFilmes, int numFilmesFav);
float calcularReincidencia(const char* genresUsuario, const char* genresFilme);
bool filmeNoArquivo(const char* titleFilme, Filme filmes[], int numFilmes);
void inserirFilmesFavoritos(const Filme filmes[], int numFilmes, char filmesFavoritos[][100], char genresFavoritos[][100], int maxFilmesFav);
void buscarFilmes(const Filme filmes[], int numFilmes);
void listarFilmes(const Filme filmes[], int numFilmes);
void ondeEstou();
```

Principal função que apresenta a tela inicial para o usuário.

```
// Função principal
int main() {

    printf("\nBem-vindo ao Sistema de Indicação de Filmes\n");
    printf("=====\n\n");

    int opcao;
    bool logado = false;

    // Variável para armazenar o número de filmes favoritos
    int numFilmesFavoritos = 0;
```

Criada a função de loop para após determinadas ações ele volte para o “Menu Principal” ao invés de o programa encerrar.

```
// Loop principal do menu
do {
    // Limpar os dados de filmes favoritos a cada execução
    printf("1. Cadastrar\n2. Login\n3. Sair\n4. Onde Estou?\n\nEscolha uma opcao: ");
    scanf("%d", &opcao);
    switch (opcao) {
        case 1:
            cadastrar();
            break;
        case 2:
            logado = login();
            if (logado) {
                printf("Voce esta logado.\n");
            }
            break;
        case 3:
            printf("Saindo do sistema...\n\n");
            return 0;

        case 4:
            ondeEstou();
            break;
        default:
            printf("Opcao invalida! Tente novamente.\n");
    }
} while (!logado);
```

Nesta parte do código é responsável por ler o arquivo que contém os filmes que o sistema tem para realizar a indicação.

```

if (logado) {

    FILE *arquivo = fopen("movies.txt", "r");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        return 1;
    }

    Filme filmes[MAX_FILMES];
    int numFilmes = 0;
    char linha[MAX_LINHA];

    // Lê os filmes do arquivo de texto
    while (fgets(linha, sizeof(linha), arquivo) && numFilmes < MAX_FILMES) {
        char *titulo = strtok(linha, ",");
        char *generos = strtok(NULL, "\n");

        if (titulo != NULL && generos != NULL) {
            strcpy(filmes[numFilmes].title, titulo);
            strcpy(filmes[numFilmes].genres, generos);
            numFilmes++;
        }
    }

    fclose(arquivo);
}

```

Permite gerenciar a lista de filmes favoritos do usuário.

```

const int maxFilmesFavoritos = 5;
char filmesFavoritos[maxFilmesFavoritos][100];
char genresFavoritos[maxFilmesFavoritos][100];
int numFilmesFavoritos = 0;

int escolha;
do {
    numFilmesFavoritos = 0;
    memset(filmesFavoritos, 0, sizeof(filmesFavoritos)); // Reinicializa a array de filmes favoritos
    memset(genresFavoritos, 0, sizeof(genresFavoritos)); // Reinicializa a array de gêneros favoritos
    limparTela();
    printf("\nMenu:\n");
    printf("1. Escolher filmes favoritos\n");
    printf("2. Buscar filmes\n");
    printf("3. Listar filmes\n");
    printf("4. Sair\n");
    printf("5. Onde Estou?\n");
    printf("Digite sua escolha: ");
    scanf("%d", &escolha);
    getchar();
}

```

Este bloco dentro do código é responsável por executar uma ação com base na escolha do usuário.

```

switch(escolha) {
    case 1:
        inserirFilmesFavoritos(filmes, numFilmes, filmesFavoritos, genresFavoritos, maxFilmesFavoritos, &numFilmesFavoritos);
        break;
    case 2:
        buscarFilmes(filmes, numFilmes);
        break;
    case 3:
        listarFilmes(filmes, numFilmes);
        break;
    case 4:
        printf("Encerrando o programa.\n");
        break;
    case 5:
        ondeEstou();
        break;
    default:
        printf("Escolha invalida. Por favor, digite um numero do menu.\n");
}
} while (escolha != 4);
}

return 0;
}

```

Para que a cada opção selecionada a tela fosse sendo poluída com informações anteriores, é colocada essa função para que a tela fosse limpa, assim permitindo que ficasse uma tela mais limpa. A função funciona tanto para o sistema operacional Windows quanto Linux.

```

// Função para limpar a tela
void limparTela() {
    // Windows
    #ifdef _WIN32
        system("cls");
    // Unix-like (Linux, macOS)
    #else
        system("clear");
    #endif
}

```

Exibe um mapa das funções para o usuário.

```
void ondeEstou() {
    limparTela();
    printf("\nMenu Principal\n");
    printf("\n");
    printf("|-- 1. Cadastrar\n");
    printf("\n");
    printf("|-- 2. Login\n");
    printf("\n");
    printf("|    |-- 2.1. Inserir filmes favoritos\n");
    printf("\n");
    printf("|    |-- 2.2. Buscar filmes\n");
    printf("\n");
    printf("|    |-- 2.3. Listar filmes\n");
    printf("\n");
    printf("|    |-- 2.4. Onde Estou?\n");
    printf("\n");
    printf("|    |-- 2.5. Sair\n");
    printf("\n");
    printf("|-- 3. Onde Estou?\n");
    printf("\n");
    printf("|-- 4. Sair\n");
    printf("\nPressione ENTER para continuar...\n");
    getchar();
}
```

Esta parte do código foi criada para permitir que o usuário pudesse inserir os seus filmes favoritos. Também existe uma condição de que um filme não pode ser inserido duas vezes, sendo assim o programa sinaliza o usuário de tal fato.

```
// Função para inserir filmes favoritos do usuário
void inserirFilmesFavoritos(const Filme filmes[], int numFilmes, char filmesFavoritos[][100], char genresFavoritos[][100], int maxFilmesFav)
{
    limparTela();
    printf("Digite seus filmes favoritos (ate %d filmes) [Presentes na base de dados]:\n", maxFilmesFavoritos);
    for (int i = 0; i < maxFilmesFavoritos; i++) {
        printf("Filme %d: ", i + 1);
        fgets(filmesFavoritos[i], sizeof(filmesFavoritos[i]), stdin);
        for (int j = 0; filmesFavoritos[i][j] != '\0'; j++) {
            filmesFavoritos[i][j] = tolower(filmesFavoritos[i][j]);
        }
        filmesFavoritos[i][strlen(filmesFavoritos[i])] = '\0';
        if (strlen(filmesFavoritos[i]) == 0) {
            printf("Por favor, insira o nome de um filme.\n");
            i--;
            continue;
        }
        if (filmeNoArquivo(filmesFavoritos[i], filmes, numFilmes)) {
            // Verifica se o filme já está na lista de favoritos
            bool filmeJaInserido = false;
            for (int j = 0; j < *numFilmesFavoritos; j++) {
                if (strcmp(filmesFavoritos[i], filmesFavoritos[j]) == 0) {
                    filmeJaInserido = true;
                    break;
                }
            }
            if (filmeJaInserido) {
                printf("Este filme já está na sua lista de favoritos.\n");
                i--;
                continue;
            }
        }
    }
}
```

Inserido um limite de filmes favoritos que o usuário pode colocar, se excedido o limite o sistema retorna com uma mensagem para o usuário.

```
    }
    if (!filmeJaInserido) {
        for (int j = 0; j < numFilmes; j++) {
            if (strcmp(filmes[j].title, filmesFavoritos[i]) == 0) {
                strcpy(genresFavoritos[*numFilmesFavoritos], filmes[j].genres);
                (*numFilmesFavoritos)++;
                break;
            }
        }
        if (*numFilmesFavoritos >= maxFilmesFavoritos) {
            printf("Voce atingiu o numero maximo de filmes favoritos (%d).\n", maxFilmesFavoritos);
            break;
        }
    } else {
        printf("Este filme ja foi inserido na lista de favoritos. Insira outro filme.\n");
        i--;
    }
} else {
    printf("Filme nao encontrado na base de dados. Por favor, insira um filme valido.\n");
    i--;
}
}
```

Como o programa se baseia em gêneros de filmes, teve de ser desenvolvido uma funcionalidade para que a partir da entrada de um gênero de filme, para assim o programa devolver uma saída baseada nisso.

```
    recomendarFilmes(filmesFavoritos, genresFavoritos, filmes, numFilmes, *numFilmesFavoritos);
}

// Função para buscar filmes por título ou gênero
void buscarFilmes(const Filme filmes[], int numFilmes) {
    char termoBusca[100];
    limparTela();
    printf("Digite um termo de busca (título ou genero, ignorando acentuacao): ");
    fgets(termoBusca, sizeof(termoBusca), stdin);
    for (int i = 0; termoBusca[i] != '\0'; i++) {
        termoBusca[i] = tolower(termoBusca[i]);
    }
    termoBusca[strlen(termoBusca)] = '\0';

    bool encontrado = false;
    printf("Resultados da busca:\n");
    for (int i = 0; i < numFilmes; i++) {
        if (strstr(filmes[i].title, termoBusca) != NULL || strstr(filmes[i].genres, termoBusca) != NULL) {
            printf("%s\n", filmes[i].title);
            encontrado = true;
        }
    }
}
```

Se porventura for pesquisado um filme que não consta nos arquivos do programa, o mesmo devolve uma mensagem informando a não existência do mesmo. Junto dessa função também foi aproveitado para colocar uma outra opção que permite a exibição de todos os filmes que estão dentro do programa.

```
}
if (!encontrado) {
    printf("Nenhum filme encontrado correspondente ao termo de busca.\n");
}
printf("Pressione ENTER para continuar...\n");
getchar();
}

// Função para listar todos os filmes
void listarFilmes(const Filme filmes[], int numFilmes) {
    limparTela();
    printf("Lista de filmes disponiveis:\n");
    for (int i = 0; i < numFilmes; i++) {
        printf("%s\n", filmes[i].title);
        if ((i + 1) % 25 == 0) {
            printf("Pressione ENTER para continuar...\n");
            getchar();
        }
    }
}
```

O sistema dispõe de um cálculo para aferir a reincidência entre gêneros de filmes e usuário. Tal função permite indicar qual a porcentagem de gênero mais vista pelo usuário.

```
// Função para calcular a reincidência de gêneros entre o usuário e um filme
float calcularReincidencia(const char* genresUsuario, const char* genresFilme) {
    int coincidencias = 0;
    int totalGenerosUsuario = 1;
    for (int i = 0; genresUsuario[i] != '\0'; i++) {
        if (genresUsuario[i] == ',') totalGenerosUsuario++;
    }

    char generosCopia[100];
    strcpy(generosCopia, genresFilme);

    char *token = strtok(generosCopia, ",");
    while (token != NULL) {
        if (strstr(genresUsuario, token) != NULL) {
            coincidencias++;
        }
        token = strtok(NULL, ",");
    }

    return (coincidencias / (float)totalGenerosUsuario) * 100.0;
}
```



Em outra parte do código existe uma função para pesquisar um filme dentro do programa, para auxiliar na verificação da existência do filme dentro do programa foi criado esta função com intuito de voltar com um saída aferindo se existe o filme ou não.

```
// Função para verificar se o filme está no arquivo
bool filmeNoArquivo(const char* titleFilme, Filme filmes[], int numFilmes) {
    for (int i = 0; i < numFilmes; i++) {
        if (strcmp(filmes[i].title, titleFilme) == 0) {
            return true;
        }
    }
    return false;
}
```

O programa é baseado em recomendação de filmes, com base no gênero inserido pelo usuário, o sistema retorna as opções de acordo com a entrada recebida.

```
// Função para recomendar filmes
void recomendarFilmes(const char filmesFavoritos[][100], const char genresFavoritos[][100], Filme filmes[], int numFilmes, int numFilmesFav) {
    float reincidencias[numFilmes];
    for (int i = 0; i < numFilmes; i++) {
        reincidencias[i] = 0.0;
        for (int j = 0; j < numFilmesFavoritos; j++) {
            reincidencias[i] += calcularReincidencia(genresFavoritos[j], filmes[i].genres);
        }
    }
}
```

Se o filme já tiver aparecido para o usuário existe uma função para que verifique essa reincidência e assim troque o filme.

```
// Ordenar os filmes pela reincidência (simples bubble sort)
for (int i = 0; i < numFilmes - 1; i++) {
    for (int j = i + 1; j < numFilmes; j++) {
        if (reincidencias[i] < reincidencias[j]) {
            // Trocar reincidências
            float tempReincidencia = reincidencias[i];
            reincidencias[i] = reincidencias[j];
            reincidencias[j] = tempReincidencia;

            // Trocar filmes
            Filme tempFilme = filmes[i];
            filmes[i] = filmes[j];
            filmes[j] = tempFilme;
        }
    }
}
```

O programa vai recomendar um filme para o usuário, contanto que o filme não esteja na lista de favoritos do usuário. É levado em conta o fator de reincidência .

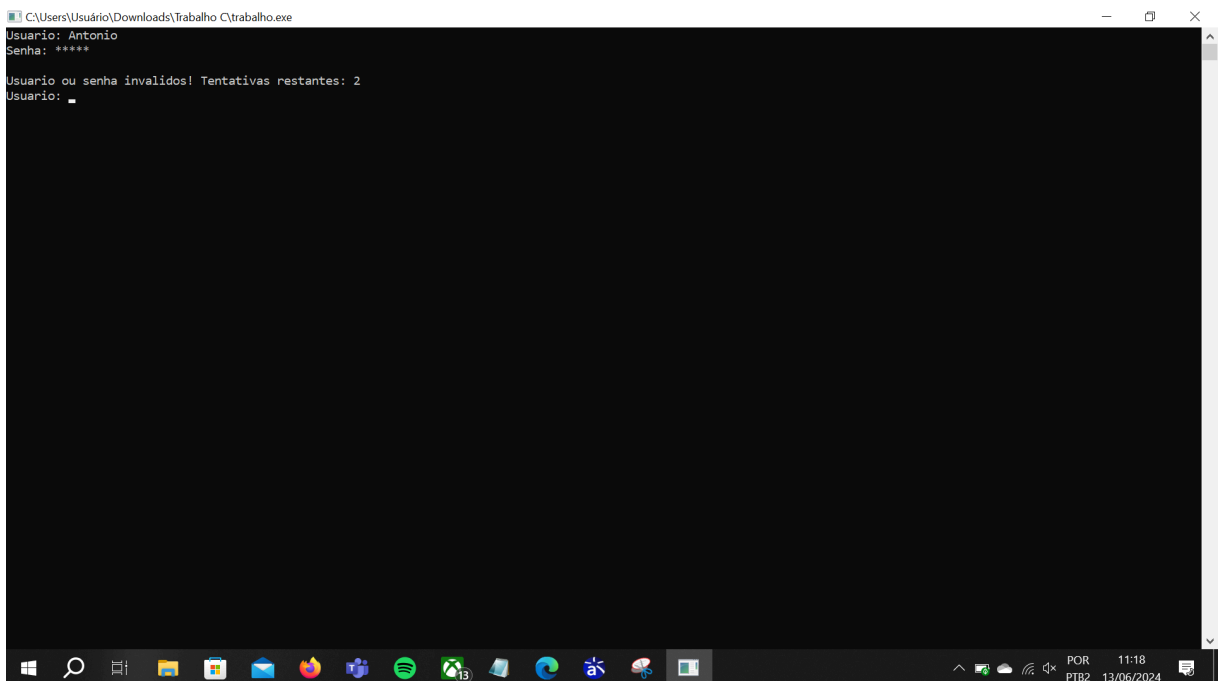
```
printf("Top filmes recomendados:\n");
for (int i = 0; i < NUM_FILMES_RECOMENDADOS && i < numFilmes; i++) {
    // Verifica se o filme j foi inserido na lista de favoritos
    bool filmeInserido = false;
    for (int j = 0; j < numFilmesFavoritos; j++) {
        if (strcmp(filmesFavoritos[j], filmes[i].title) == 0) {
            filmeInserido = true;
            break;
        }
    }

    // Se o filme não estiver na lista de favoritos, ele é recomendado
    if (!filmeInserido) {
        printf("%s (Reincidência: %.2f%%)\n", filmes[i].title, reincidencias[i]);
    }
}
printf("Pressione ENTER para continuar...\n");
getchar();
}
```

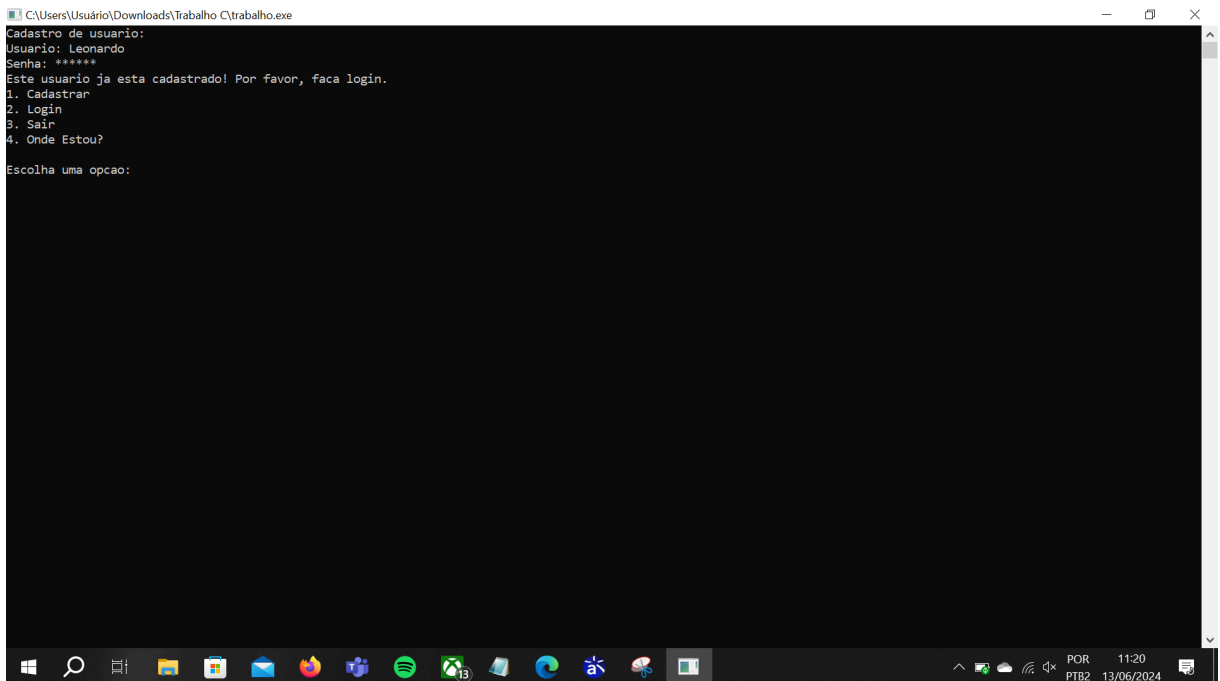
## 6. TESTES

Para determinar a eficácia do código foi realizada uma série de testes para assegurar de que está operante e funcional. Os testes que foram feitos tem como ênfase a busca de erros ou brechas dentro dele, com isso pode-se aperfeiçoar a confiabilidade do programa

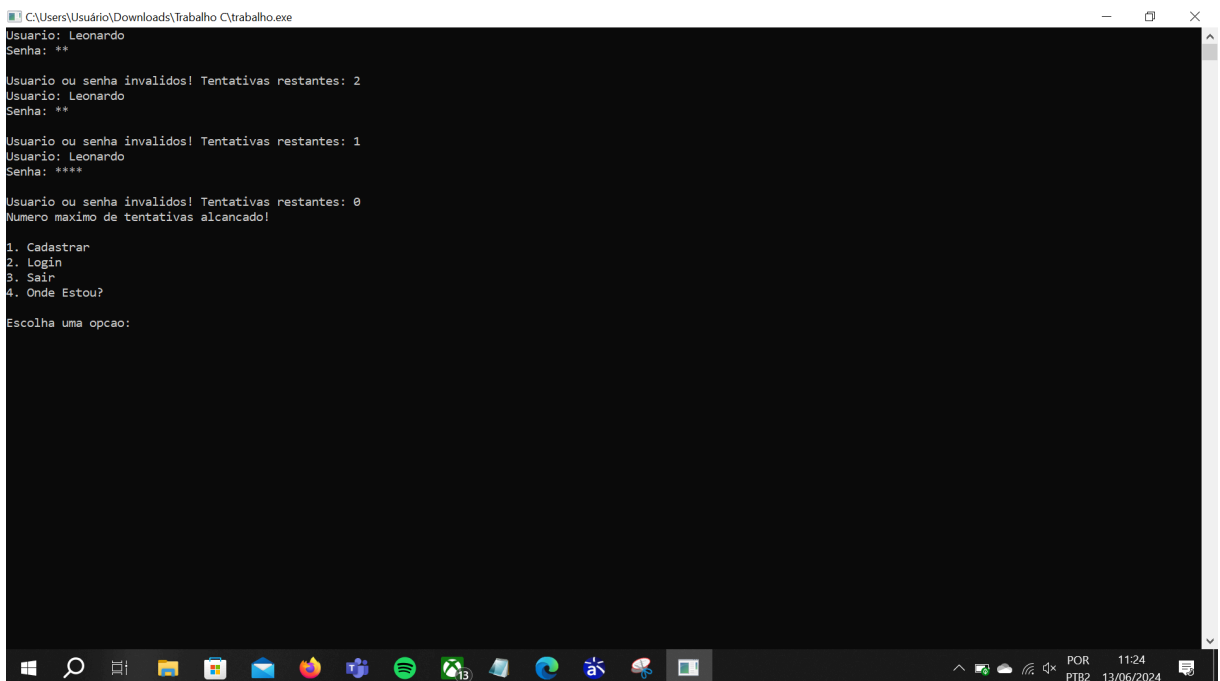
O sistema não permite a entrada de um usuário se ele não fizer um cadastro primeiramente.



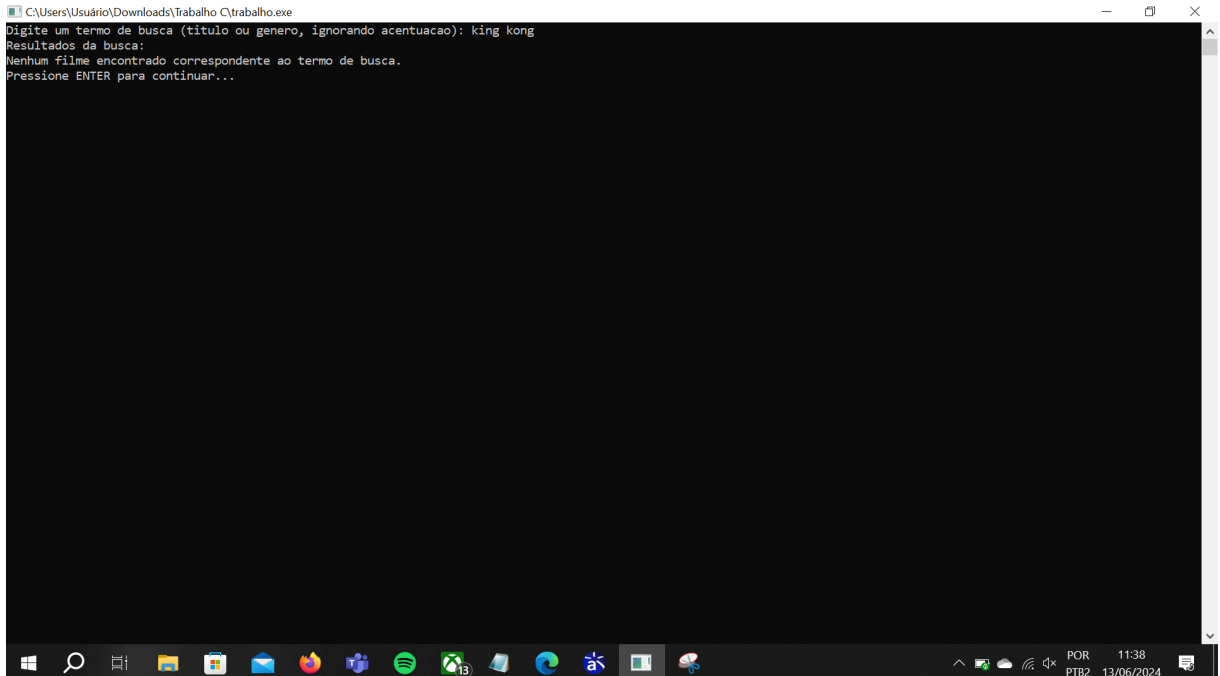
Não é possível existir dois usuários com o mesmo cadastro. O sistema notifica na própria tela que já tem um usuário com este nome.



Existe um número limite de tentativas que o usuário pode fazer para entrar no sistema. O limite foi estabelecido em 3 tentativas, após exceder o número de tentativas o sistema volta para a tela principal do sistema.

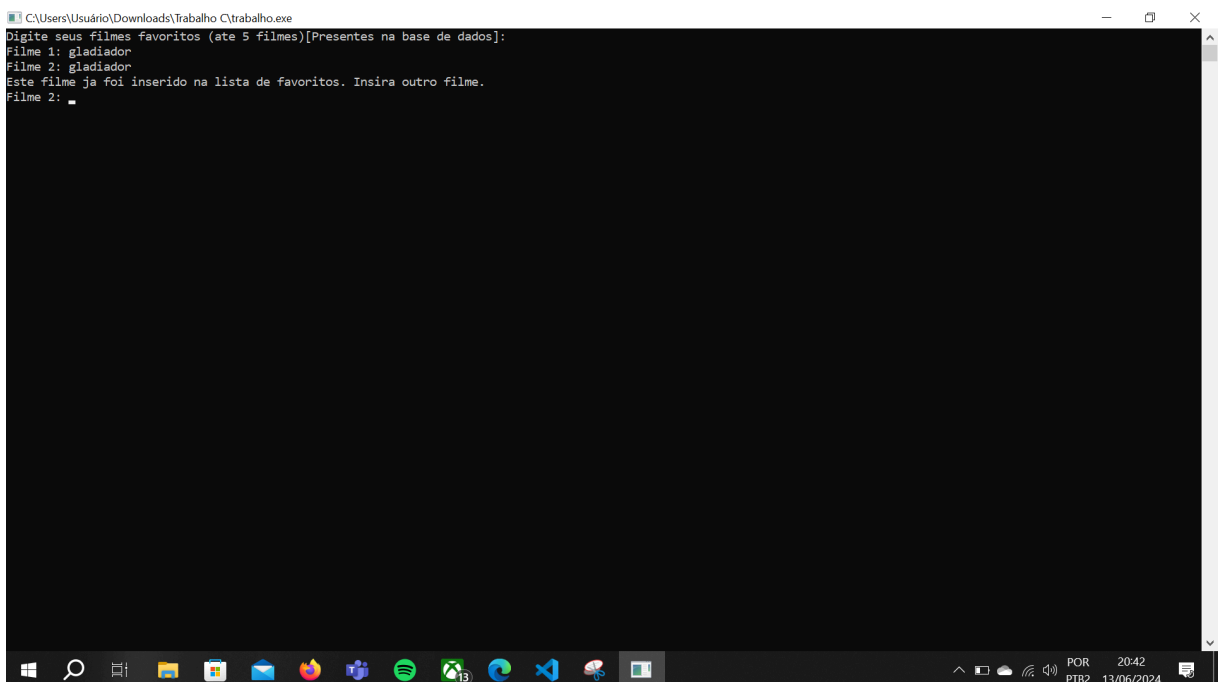


Ao se realizar uma busca por um filme específico e não for encontrado dentro dos arquivos do sistema, o mesmo retorna uma mensagem para o usuário com a seguinte informação: “Nenhum filme encontrado correspondente ao termo de busca”.



```
C:\Users\Usuário\Downloads\Trabalho C\trabalho.exe
Digite um termo de busca (titulo ou genero, ignorando acentuacao): king kong
Resultados da busca:
Nenhum filme encontrado correspondente ao termo de busca.
Pressione ENTER para continuar...
```

Ao se listar cinco filmes favoritos não pode haver repetição de uma mesmo filmes, caso contrário o sistema emite uma mensagem de que o filme já foi inserido.



```
C:\Users\Usuário\Downloads\Trabalho C\trabalho.exe
Digite seus filmes favoritos (ate 5 filmes)[Presentes na base de dados]:
Filme 1: gladiador
Filme 2: gladiador
Este filme ja foi inserido na lista de favoritos. Insira outro filme.
Filme 2: _
```

## 7. MANUAL DO USUÁRIO

O manual do usuário serve como orientação para quem tem interesse em fazer uso do programa. Para fazer uso do programa é disponibilizado um link onde pode baixar para usar.

Lembrando que ao baixar os arquivos no link todos têm de estar no mesmo diretório, caso contrário não será possível executar o programa.

### □ Arquivos do programa de recomendação de filmes

#### 1. Introdução

- Bem-vindo ao Sistema de Indicação de Filmes! (SIF) Este sistema permite que você cadastre-se, faça login, insira seus filmes favoritos e receba recomendações personalizadas com base nos seus gostos.

#### 2. Operações Disponíveis

##### 2.1. Cadastro

- Para se cadastrar no sistema, escolha a opção 1. Cadastrar no menu principal.
- Insira um nome de usuário e uma senha quando solicitado.
- Caso o usuário já exista, será exibida uma mensagem informando que o usuário já está cadastrado.

##### 2.2. Login

- Após se cadastrar, escolha a opção 2. Login no menu principal.
- Insira o nome de usuário e a senha cadastrados.
- Em caso de falha no login, serão fornecidas até três tentativas antes de ser bloqueado.

### 2.3. Escolher Filmes Favoritos

- Após efetuar login com sucesso, você poderá inserir seus filmes favoritos.
- Escolha a opção 2.1. Escolher filmes favoritos no menu.
- Insira o nome dos filmes favoritos (até 5 filmes).
- Os filmes favoritos serão usados para recomendações personalizadas.

### 2.4. Buscar Filmes

- Você pode buscar filmes por título ou gênero.
- Escolha a opção 2. Buscar filmes no menu.
- Insira um termo de busca (título ou gênero) e pressione Enter.
- Os resultados da busca serão exibidos.

### 2.5. Listar Filmes

- Visualize a lista completa de filmes disponíveis.
- Escolha a opção 3. Listar filmes no menu.
- A lista será exibida, permitindo a visualização de 25 (Se Houver) filmes por vez.

### 2.6. Sair

- Encerre sua sessão no sistema.
- Escolha a opção 4. Sair. (No Menu Principal) ou 5. Sair (No Menu Operacional)
- Você será desconectado e o programa será encerrado.

## 3. Onde estou?

- Em qualquer momento, você pode verificar onde está no sistema.
- Escolha a opção 3. Onde estou? no menu.
- Será exibido o menu atual em que você se encontra.

#### 4. Finalizando o Programa

Sempre que desejar sair do sistema, você pode escolher a opção correspondente no menu.

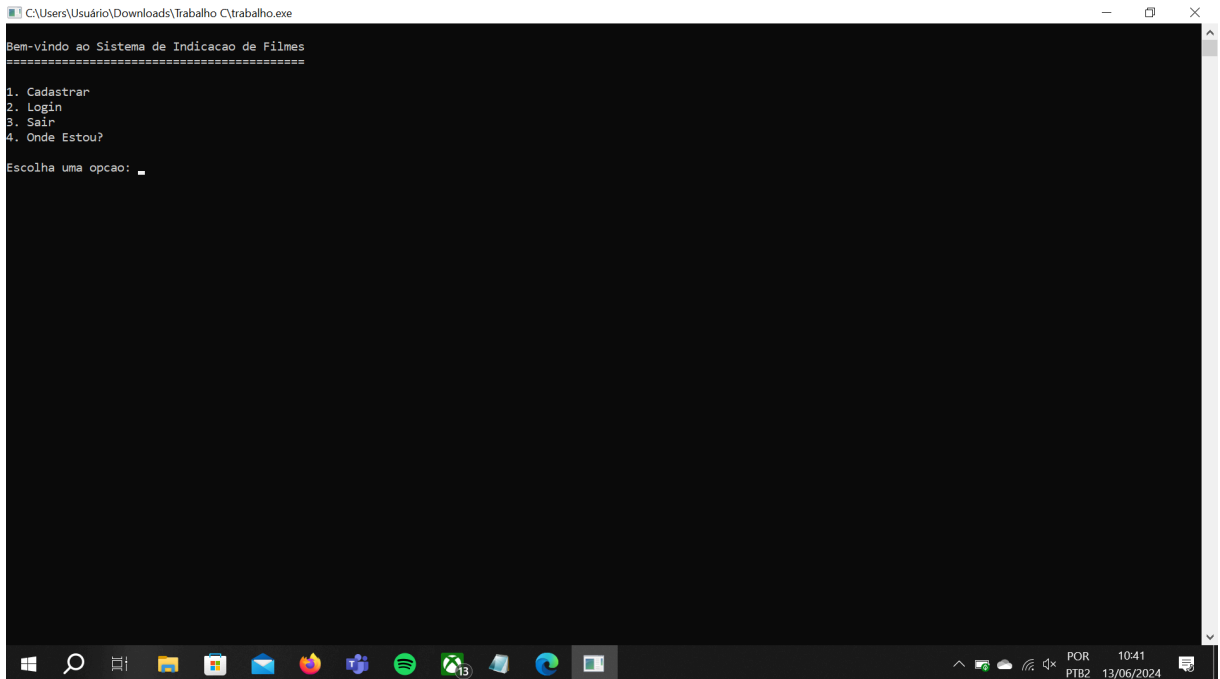
Todas as informações serão salvas automaticamente, e você será desconectado.



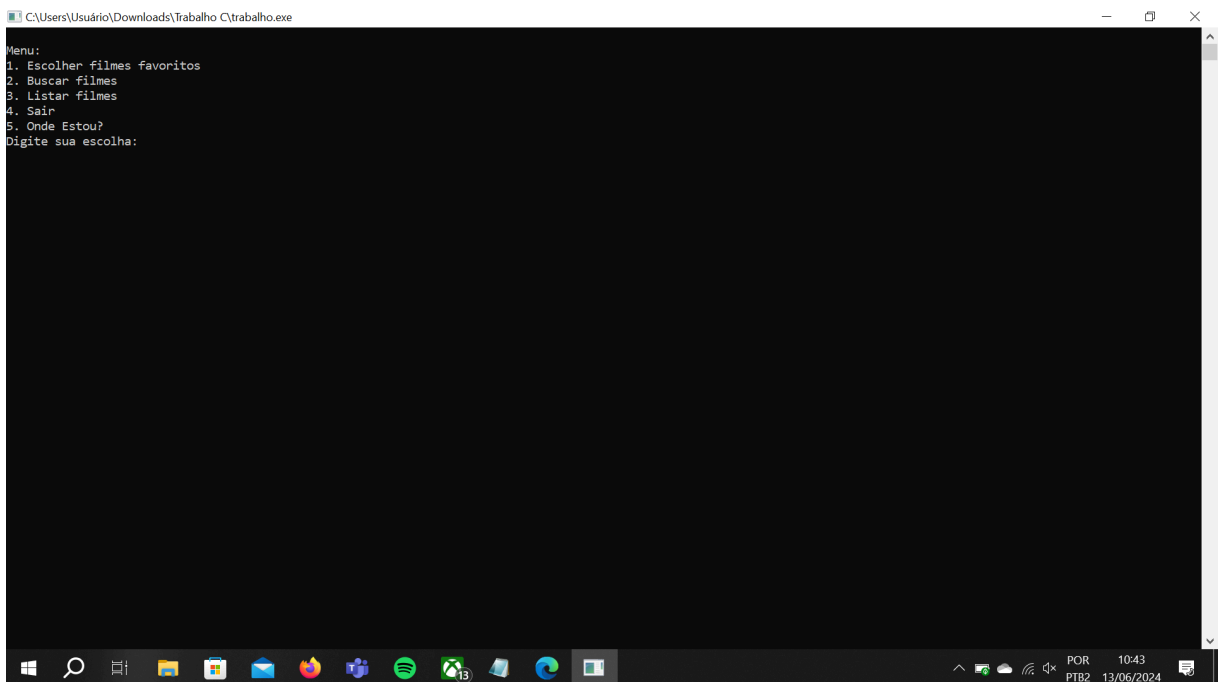
## 8. MAPA DOS MENUS

Menu Principal do sistema já em operação.

### 1. Cadastrar



### 2. Login



## **9. CONSIDERAÇÕES FINAIS**

Em conclusão, o estudo da linguagem C revelou-se uma ferramenta essencial para a compreensão dos fundamentos da programação. A sua sintaxe clara e estruturada, juntamente com a sua eficiência e flexibilidade, tornam-na uma escolha ideal para uma ampla gama de aplicações, desde a programação de sistemas até ao desenvolvimento de software de alto nível. A familiaridade com a linguagem C não só melhora a capacidade de um programador de trabalhar eficazmente com estruturas de dados complexas e algoritmos avançados, mas também fornece uma base sólida para o aprendizado de outras linguagens de programação.

Além disso, a importância da linguagem C no mundo acadêmico e industrial não pode ser subestimada. Ela continua a ser uma das linguagens de programação mais utilizadas, com uma vasta comunidade de programadores e uma rica biblioteca de recursos disponíveis. Através deste trabalho, esperamos ter destacado a relevância contínua da linguagem C e encorajamos os futuros programadores a explorar as suas potencialidades. A linguagem C, com a sua robustez e versatilidade, continuará a ser uma ferramenta valiosa para os programadores nas próximas décadas.