

Sistema de Biblioteca

1. Introdução e Contexto

O projeto consiste na criação de um **sistema de biblioteca** desenvolvido integralmente em **linguagem C**, no qual a interação entre o usuário (administrador) e o programa é realizada diretamente pelo **terminal do sistema operacional**.

O objetivo do sistema é permitir o **gerenciamento completo de uma biblioteca**, possibilitando o registro de livros e usuários, o controle de empréstimos e devoluções, a verificação de disponibilidade de exemplares, a consulta da situação de cada usuário (se possui empréstimos pendentes ou não), a contagem de exemplares disponíveis e a organização do acervo de forma dinâmica e atualizada.

O gerenciamento de informações em larga escala requer soluções tecnológicas eficientes, pois a análise e a verificação manuais de dados são processos lentos e sujeitos a erros. Nesse sentido, o sistema proposto tem como propósito **automatizar e simplificar o acesso às informações**, evitando o uso de planilhas ou arquivos físicos para armazenar dados de histórico, informações pessoais ou registros de livros.

Este projeto foi escolhido por reunir diversos conteúdos abordados na disciplina **Introdução às Técnicas de Programação**, como o uso de **funções**, **strings**, **vetores**, **ponteiros** e **estruturas definidas pelo usuário (*typedef*)**, entre outros.

2. Análise Técnica

2.1 Ferramentas Utilizadas

- **IDE:** Code::Blocks
- **Editor de código:** Integrado ao Code::Blocks
- **Compilador:** GCC (GNU Compiler Collection)
- **Sistema operacional:** Windows

- **Terminal:** Prompt de Comando (Windows Terminal)
-

2.2 Estruturas e Organização do Código

Para iniciar o projeto, foi necessário utilizar **estruturas *typedef***, permitindo que informações relacionadas a um mesmo “objeto” fossem agrupadas em uma única variável. Esse recurso possibilita **maior controle sobre os dados e facilita operações de atribuição e comparação**. Caso não fosse utilizado, o código seria mais complexo e de difícil manutenção, comprometendo o andamento do projeto.

As **estruturas de decisão** foram construídas para permitir que o usuário escolha ações específicas, e o programa execute a função correspondente. Cada função chamada depende do **input do usuário**, como observado no **loop principal** presente na função *main*.

Durante a execução, o sistema apresenta opções como “Cadastrar Livro”, “Cadastrar Usuário” e “Pegar Emprestado”. Por exemplo, ao digitar o número **1**, o programa chama a função *cadastra_usuario()* e realiza o cadastro solicitado.

2.3 Estruturas Condicionais e Comparações

As **estruturas condicionais** também foram utilizadas na **verificação de *strings***, permitindo comparar dois textos e determinar se são iguais. Um exemplo fundamental está na função *cadastra_livro()*, na qual o sistema solicita o título, o autor e o número ISBN (uma espécie de “CPF” do livro).

Caso o ISBN informado já esteja presente no acervo, o programa informa que o livro já foi cadastrado. Para isso, é usada a função *strcmp()* da biblioteca *string.h*, que retorna **0** quando duas *strings* são idênticas.

2.4 Estruturas de Dados e Repetição

Como o sistema precisa armazenar uma grande quantidade de informações, foi necessário reservar espaço em **vetores**, que servem para listar livros e usuários. Manter esses dados organizados em posições específicas permite que o programa **localize registros rapidamente** e verifique se ainda há espaço disponível para novos cadastros.

Os vetores *acervo[]* e *cadastrados[]* armazenam, respectivamente, as informações de cada livro e de cada usuário. Com o uso de **estruturas de repetição**, o programa percorre os vetores para realizar operações como encontrar espaços livres, identificar duplicidades (ISBN ou CPF já cadastrados) e atualizar dados de empréstimos.

A comparação entre *strings* digitadas pelo usuário e os valores armazenados no vetor permite verificar se determinado livro ou usuário já existe no sistema. Esse mesmo processo é repetido na função *pega_emprestado()*.

2.5 Funções Implementadas

As funções do sistema foram desenvolvidas com **objetivos específicos**, voltados ao **processamento de dados** e ao **tratamento de possíveis erros**.

- **Função *cadastra_livro()*** – Solicita nome, autor e ISBN, registrando as informações no vetor *acervo[]*.
- **Função *cadastra_usuario()*** – Solicita nome e CPF, além de registrar o número de empréstimos realizados pelo usuário.
- **Função *pega_emprestado()*** – Verifica a existência do livro no acervo, a disponibilidade de exemplares e o cadastro do usuário antes de confirmar a operação.
- **Função *remove_nova_linha()*** – Corrige um erro comum gerado pelo uso de *fgets()*, que adiciona o caractere “\n” ao final das *strings*. Esse caractere poderia causar falhas nas comparações de texto, por isso a função é chamada após cada entrada de dados para garantir a integridade das *strings*.

Foram definidos dois **tipos estruturados** (*typedefs*):

- **Livro:** contém os campos *char nome*, *char autor*, *char isbn* e *int qnt* (quantidade disponível).
 - **Usuario:** contém os campos *char nome*, *char cpf* e *int emprestimos* (quantidade de livros emprestados).
-

3. Implementação e Reflexão

O código do projeto foi totalmente modularizado, dividido em funções independentes, o que facilita a **organização**, a **manutenção** e o **entendimento do fluxo de execução**. O caminho seguido pelo programa depende exclusivamente das ações do usuário, o que torna o sistema dinâmico e adaptável.

O trabalho com *strings* apresentou algumas dificuldades, especialmente nas **operações de comparação e atribuição**, que exigem atenção à sintaxe e ao controle de memória. Problemas com o caractere “\n”, inserido pela função *fgets()*, resultaram em erros lógicos, pois *strings* visualmente idênticas eram tratadas como diferentes. A implementação da função *remove_nova_linha()* resolveu essa questão.

Outra dificuldade inicial foi compreender como modificar vetores dentro das funções. A correta **passagem de parâmetros por referência**, utilizando o operador “&” e ponteiros, foi essencial para garantir que as alterações fossem refletidas nas variáveis originais.

Apesar dos desafios, o uso de *typedefs* e *strings* desde o início deu corpo ao projeto e acelerou o desenvolvimento.

A estrutura atual do código foi escolhida por sua **simplicidade e clareza**, utilizando conceitos básicos de programação em C. No entanto, há espaço para aprimoramentos. Por exemplo, na função *main()*, o usuário interage digitando números para escolher ações. Embora funcional, essa abordagem é considerada simples demais e poderá ser substituída por um menu mais interativo no futuro.

4. Conclusão

A experiência obtida com o desenvolvimento deste sistema proporcionou uma compreensão mais profunda sobre:

- a **manipulação de vetores** dentro de funções;
- o uso de estruturas *typedef*;
- e o **tratamento adequado de *strings***.

Ainda há melhorias planejadas, como a verificação de entradas inválidas, a otimização do fluxo das funções e a prevenção de erros causados por entradas incorretas. Também está prevista a implementação de **novas funcionalidades**,

como a **devolução de livros**, a **listagem do acervo** e a **ordenação dos exemplares**.

O projeto, portanto, representa uma base sólida para o aprendizado e a aplicação prática dos conceitos fundamentais de programação em C, demonstrando como técnicas simples podem ser combinadas para criar um sistema funcional e escalável.