UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

INSTITUTO METRÓPOLE DIGITAL

INTRODUÇÃO ÀS TÉCNICAS DE PROGRAMAÇÃO

RELATÓRIO TÉCNICO

Discente: Síntia Raianne Bezerra de Souza

Natal/RN
2025

1. INTRODUÇÃO E CONTEXTO

1.1. NOME DO PROJETO E OBJETIVO

O projeto desenvolvido consiste em um Sistema de Gerenciamento de Biblioteca, implementado como uma aplicação de console utilizando a Linguagem de Programação C. O principal objetivo do sistema foi auxiliar na administração e gerenciamento do acervo de uma biblioteca, permitindo o cadastro e listagem de livros e usuários adicionados, como também o controle de empréstimos.

1.2. PROBLEMA QUE O PROJETO RESOLVE

O sistema aborda a necessidade de automatizar operações que, em bibliotecas de pequeno porte, ainda são executadas manualmente, como o controle de empréstimos e do acervo, normalmente realizados através de planilhas ou fichas de papel. Esse método apresenta maior suscetibilidade a erros, como empréstimos duplicados, perda de registros, dificuldade para verificar a disponibilidade dos livros, ausência de controle sobre limites de empréstimos por usuários e dificuldade na identificação dos responsáveis pelos livros emprestados. A solução proposta busca automatizar e centralizar esse processo, oferecendo maior confiabilidade, agilidade e facilidade ao consultar os dados.

1.3. JUSTIFICATIVA DA ESCOLHA DO PROJETO

A escolha de um Sistema de Gerenciamento de Biblioteca como tema do projeto foi motivada por diversos fatores. Em primeiro lugar, a relevância prática do tema, pois esse tipo de sistema representa uma aplicação real e de fácil assimilação, considerando que a maioria das pessoas já teve algum contato com o funcionamento de bibliotecas. Outro fator importante foi o alinhamento do escopo do projeto com os conteúdos abordados na primeira unidade da disciplina, permitindo a aplicação direta dos conceitos apresentados até o momento e garantindo complexidade ao nível atual de aprendizagem. Além disso, o tema apresenta possibilidade de expansão para as próximas unidades, permitindo a inserção gradual de novas funcionalidades e o aprofundamento em conceitos mais complexos.

2. ANÁLISE TÉCNICA

2.1. METODOLOGIA

Para o projeto foram utilizadas as seguintes Ferramentas de Desenvolvimento:

- **Linguagem de Programação:** A linguagem C foi a base para o desenvolvimento de toda a lógica do sistema, implementação de algoritmos e manipulação das estruturas de dados necessárias.
- **Compilador:** O compilador GCC (GNU Compiler Collection), integrado ao ambiente de desenvolvimento MinGW e instalado por meio da plataforma MSYS2 no Windows, foi o responsável por traduzir o código-fonte em C para um programa executável.
- **Editor de Código:** O Visual Studio Code foi utilizado para a escrita, depuração e organização do código.
- **Sistema Operacional:** O projeto foi desenvolvido no Windows 11.

- **Controle de Versão:** O Git foi utilizado para gerenciar o histórico de alterações do código, registrando a evolução do projeto através de commits com mensagens descritivas.
- **Hospedagem do Repositório**: O GitHub serviu como plataforma de hospedagem remota do repositório, facilitando o armazenamento seguro do código, a documentação detalhada no README.md e o acompanhamento da evolução do projeto.

2.2. APLICAÇÃO DOS CONCEITOS DA U1

2.2.1. COMO FORAM USADAS AS ESTRUTURAS CONDICIONAIS

Na função *cadastrarLivros(),* por exemplo, a condição verifica se o limite máximo de livros já foi atingido, antes de permitir um novo cadastro. Já na função *realizarEmprestimo()* múltiplas verificações são realizadas por meio do *if*, como a existência de livros e usuários cadastrados, a validação dos IDs informados, a disponibilidade do livro solicitado e o controle do limite de empréstimos por usuário. Nas funções de *listagem listarLivros()* e *listarUsuarios()* o *if* e o *else*, verificam se existem registros antes de exibir as informações. E ainda na *função listarLivros()* um bloco *if* e *else* determina o status de cada livro, identificando se está disponível ou emprestado.

A estrutura *switch-case,* presente na função *main(),* foi implementada para direcionar a execução para diferentes funcionalidades do sistema, de acordo com a opção numérica inserida pelo usuário. O *default* do *switch* também garante o tratamento de entradas inválidas. Essa abordagem torna a implementação do menu mais clara e organizada do que seria com múltiplos blocos de *if* e *else if*.

2.2.2. QUAL A LÓGICA DAS ESTRUTURAS DE REPETIÇÃO IMPLEMENTADAS

O loop principal *do while*, presente na função *main()* garante a execução contínua do programa, mantendo-o ativo até que o usuário seleciona a opção 0, para encerrar. Essa condição é controlada pelo *while (opcao != 0)* e verificada ao final de cada iteração. Nas funções *listarLivros()* e *listarUsuarios()* os laços *for* percorrem sequencialmente os vetores desde o índice inicial até o limite de elementos cadastrados, permitindo a exibição de todos os registros. Já na função *realizarEmprestimo(),* o laço é utilizado para percorrer o vetor de empréstimos do usuário em busca de uma posição disponível. Por fim, na função *cadastrarUsuario()* o *for* é empregado para inicializar o vetor de empréstimos com valores padrão.

2.2.3. COMO OS VETORES FORAM APLICADOS NO PROJETO

No gerenciamento dos livros, foram implementados vetores paralelos: *titulo_livro[maximo_livros][maximo_titulo]* para armazenar os títulos das obras, *autor_livro[maximo_livros][maximo_autor]* para registrar os autores de cada livro, e o vetor inteiro *disponivel_livro[maximo_livros]*, responsável por controlar a disponibilidade dos livros por meio de valores binários (1 para disponível e 0 para emprestado). A mesma lógica foi aplicada no gerenciamento de usuários, onde *nome_usuario[maximo_usuarios][maximo_nome]* armazena os nomes completos e *matricula_usuario[maximo_usuarios]* registram as matrículas.

Para organizar os empréstimos, foi utilizado o vetor bidimensional *emprestimos_usuario[maximo_usuarios][maximo_emprestimos]* que relaciona usuários aos livros emprestados. Neste vetor cada linha representa um usuário e as colunas registram os IDs dos livros emprestados por ele, funcionando como uma tabela de empréstimos. E os vetores de controle *numero_emprestimo_usuario[maximo_usuarios]* que verificam a quantidade de empréstimos ativos por usuário, enquanto *disponivel_livro[maximo_livros]* garante que nenhum livro possa ser emprestado mais de uma vez simultaneamente.

2.2.4.ORGANIZAÇÃO E FUNÇÃO DAS FUNÇÕES CRIADAS

As funções *cadastrarLivro()* e *cadastrarUsuario()* são responsáveis pela criação de registros, recebendo os dados fornecidos pelo usuário e armazenando – os nos vetores correspondentes. As funções *listarLivros()* e *listarUsuarios()* tratam exclusivamente da exibição dos dados cadastrados, enquanto a função *realizarEmprestimo()* centraliza toda a lógica de empréstimos, englobando as validações necessárias e a atualização dos vetores envolvidos. Além disso, o sistema também conta com funções auxiliares: *menuBiblioteca()* é responsável por exibir as opções do menu, e *limparTela()* melhora a experiência do usuário ao limpar o console antes de retornar ao menu principal. E, por fim, a função *main()* controla o fluxo principal do programa, gerenciando a exibição do menu e chamando as demais funções conforme as entradas do usuário.

2.3. ESTRUTURAS DE DADOS

O controle de estado do programa é realizado pelas variáveis globais *numero_livros* e *numero_usuarios*, que atuam como contadores e índices para os cadastros ativos, enquanto o vetor *numero_emprestimo_usuario[]*, monitora a quantidade de empréstimos por usuário. Além das variáveis globais, também foram utilizadas variáveis locais para gerenciar escopos específicos dentro das funções. Na função *main()*, a variável *opção* controla o fluxo do menu principal, armazenando temporariamente o número fornecido pelo usuário. A função *listarLivros()* utiliza a variável *status_livro* que converte valores binários em texto para indicar se o livro selecionado está disponível ou emprestado. As variáveis *id_livro* e *id_usuario* presentes na função *realizarEmprestimo(),* registram temporariamente as entradas fornecidas pelo usuário durante a operação de empréstimo.

O controle da capacidade e a alocação de memória do sistema são gerenciados por constantes de pré-processador *#define*, que determinam limites operacionais. Essas constantes foram divididas em dois grupos: O primeiro grupo de constantes, estabelece os limites quantitativos, onde *maximo_livros* define o tamanho máximo do acervo bibliográfico, *maximo_usuarios* determina o número máximo de usuários que podem ser registrados no sistema, e *maximo_emprestimos* impõe o limite de empréstimos simultâneos que um único usuário pode ter. O segundo grupo especifica o comprimento máximo dos campos de texto. As constantes *maximo_titulo*, *maximo_autor* e *maximo_nome* delimitam o tamanho limite para os nomes das obras, autores e usuários, respectivamente.

3. IMPLEMENTAÇÃO E REFLEXÃO

3.1. DIFICULDADES ENCONTRADAS

Durante a implementação, diversas dificuldades técnicas surgiram, especialmente relacionadas à manipulação de *strings* na linguagem C. Uma delas ocorreu com o uso do *scanf* convencional, que interrompia a leitura ao encontrar o primeiro espaço, tornando impossível capturar títulos de livros ou nomes completos de usuários que continham múltiplas palavras. Outra dificuldade foi no gerenciamento do *buffer* do teclado, pois, após a leitura de um dado numérico o caractere de nova linha *\n*, gerado ao pressionar *Enter*, permanecia no buffer de entrada, fazendo com que a próxima chamada de *scanf()* destinada a leitura de uma string fosse ignorada, resultando em entradas vazias. Além disso, a busca pela compatibilidade do sistema em diferentes plataformas, na função *limparTela(),* uma vez que os comandos de terminal variam entre sistemas operacionais, no Windows utiliza-se o *cls,* enquanto no Linux e Mac o comando é *clear.*

3.2. SOLUÇÕES IMPLEMENTADAS

Para solucionar os problemas relacionados à leitura de strings contendo espaços, foi adotado o formato *" %[^\n]"* no *scanf*, que possibilita capturar toda a entrada do usuário até que seja pressionada a tecla Enter. Referente ao gerenciamento do *buffer* do teclado, a função *getchar()* foi utilizada após cada leitura numérica, removendo os caracteres "*\n"* que sobraram no *buffer* de entrada e evitando interferências nas próximas operações de entrada. Já para garantir a compatibilidade do sistema em diferentes plataformas, a função *limparTela()* foi implementada com diretivas condicionais *#ifdef _WIN32*, de modo que o compilador seleciona automaticamente entre *system("cls"),* no Windows e *system("clear")* no Unix/Linux/Mac durante a compilação.

3.3. ORGANIZAÇÃO DO CÓDIGO

A organização do sistema foi estruturada com base no princípio da modularização por funcionalidade, onde cada operação foi separada em funções específicas. Adotou-se também a utilização de variáveis e vetores globais, para armazenamento de informações compartilhadas, e variáveis locais, para o tratamento de dados temporários. Esta técnica permitiu simplificar o acesso entre funções e reduzir parâmetros. Além disso, o fluxo de execução segue uma estrutura intuitiva, menu → seleção → execução → retorno, com verificações preventivas e mensagens de erro específicas que orientam o usuário sobre correções necessárias, tornando o uso do sistema simples e compreensível.

3.4. CONCLUSÃO

O desenvolvimento do sistema possibilitou a aplicação prática dos conceitos da primeira unidade da disciplina. Durante a implementação, foi possível compreender de forma mais aprofundada o funcionamento de variáveis, vetores, estruturas condicionais, laços de repetição e funções.

Referente às possíveis melhorias, o sistema pode ser expandido em diferentes perspectivas, como a utilização de structs para agrupar variáveis relacionadas em um único tipo de dado, tornando o código mais organizado. Além disso, novas funcionalidades podem ser incorporadas, incluindo a devolução de livros, renovação de empréstimos e mecanismos de busca por títulos, autores ou usuários (por nome ou matrícula), ampliando a usabilidade e eficiência do sistema.

4. PERGUNTAS ORIENTADORAS

4.1. QUAIS CONCEITOS DA UNIDADE 1 FORAM APLICADOS E ONDE

Variáveis globais (*numero_livros* e *numero_usuarios* para controle de quantidades.), vetores globais (*titulo_livro[][], autor_livro[][], nome_usuario[][], matricula_usuario[], emprestimos_usuario[][]* e *disponivel_livro[]* para armazenamento de dados referente), variáveis locais (*i, id_livro, id_usuario e status_livro[]* para controle temporário). Estruturas condicionais (*if/else* para validações e *switch* para controle do menu), laços de repetição *(for* para listagens e *do while* para o menu principal) e funções (*cadastrarLivro(), listarLivros(), cadastrarUsuario(), listarUsuarios(), realizarEmprestimo(), menuBiblioteca() e limparTela()* para organização do código.

4.2. COMO A ORGANIZAÇÃO EM FUNÇÕES FACILITA A MANUTENÇÃO DO CÓDIGO

As funções foram responsáveis por separar tarefas específicas, como cadastros, listagem e empréstimos, reduzindo repetições e simplificando o processo de depuração. Essa organização possibilitou que alterações em uma operação, como por exemplo cadastro de livros, impactasse apenas na função correspondente. Além disso, a modularização tornou o código mais legível, de fácil manutenção e com maior potencial de reutilização em projetos futuros.

4.3. QUAIS FORAM OS PRINCIPAIS DESAFIOS NA IMPLEMENTAÇÃO DAS ESTRUTURAS DE REPETIÇÃO

O controle de finalização dos loops, especialmente na estrutura *do while* do menu principal, que precisava encerrar apenas quando o usuário selecionasse a opção de saída. A manipulação de índices, devido à diferença entre os IDs fornecidos pelo usuário (iniciando em 1) e os índices internos dos vetores (iniciando em 0). E por fim, a necessidade de inicializar o vetor *emprestimos_usuario[][]* com o valor -1, para indicar posições vazias, onde, antes de inserir novos empréstimos, o sistema busca por essas posições livres.

4.4. COMO OS VETORES FORAM UTILIZADOS PARA RESOLVER O PROBLEMA PROPOSTO

Os vetores foram utilizados como estrutura principal para o armazenamento e organização dos dados em memória, permitindo a estruturação de informações relacionadas, como livros e usuários e a conexão entre as entidades por meio dos registros de empréstimos. O uso de índices possibilitou acesso rápido e eficiente aos registros, enquanto os vetores de controle auxiliaram no gerenciamento de limites e quantidades, garantindo a integração de todas as operações do sistema.

4.5. QUE MELHORIAS PODERIAM SER IMPLEMENTADAS NAS PRÓXIMAS UNIDADES

Uso de *structs* para organizar os dados de livros e usuários, a implementação de novas funcionalidades, como a devolução de livros, a renovação de empréstimos e mecanismos de busca avançada. Ampliar o conjunto de informações armazenadas, referente aos usuários, incluir dados como telefone e e-mail; e, para os livros, registrar o ano de publicação e o gênero literário. Também pode ser incorporada a geração de relatórios estatísticos sobre o acervo e o uso de ponteiros.