Relatório Técnico - Unidade 1

Projeto: Sistema de biblioteca
Autora: Ana Louise Câmara da Costa
Matrícula: 20250065261
Disciplina: Introdução às técnicas de programação
Data: 27/09/2025

1. Introdução

1.1 Objetivo do Projeto

O objetivo principal deste trabalho é desenvolver um Sistema Básico de Gerenciamento de Biblioteca utilizando a Linguagem C. O foco é aplicar os conceitos da programação estruturada para criar uma ferramenta funcional capaz de simular e otimizar as rotinas essenciais de catalogação e empréstimo, priorizando a agilidade e a estabilidade.

1.2 Problema que o Projeto Resolve

O projeto visa criar um modelo simples e eficiente para resolver problemas crônicos observados em sistemas de biblioteca: a lentidão e a ocorrência frequente de erros que interrompem o fluxo de trabalho.

1.3 Fonte de Dados para o Usuário

Para utilizar o sistema, o operador da biblioteca precisa de informações fornecidas manualmente no momento da execução do programa. Os dados necessários são aqueles que o operador obtém do material físico (livro) ou do próprio usuário (dados pessoais).

1.4 Justificativa da Escolha do Projeto

A escolha do projeto é diretamente motivada pela experiência pessoal no uso de ferramentas no ambiente da biblioteca. A motivação central é gerar um sistema que consiga mitigar as reclamações mais recorrentes dos operadores facilitando o processo de catalogação, oferecendo ferramentas e validações que simplificam o processo diário de quem gerencia o acervo.

2. Análise Técnica

2.1 Metodologia: Ferramentas Utilizadas

O desenvolvimento do projeto foram utilizadas as seguintes ferramentas:

- Linguagem: C
- Compilador: GCC (via MinGW/MSYS2) 15.2.0, no sistema Windows

- Editor: Visual Studio Code.
- Sistema de versionamento: GitHub.

2.2 Aplicação dos Conceitos da Unidade 1

A implementação do sistema utilizou integralmente os conceitos abordados na Unidade 1:

- Variáveis com tipos definidos: Foram utilizadas variáveis dos tipos `int`, `char` e `bool` para representar dados (códigos, textos, status), garantindo o uso eficiente da memória.
- Operações aritméticas e relacionais: Operações relacionais (==, !=, >, <) foram cruciais nas funções de validação de dados (`validarNumero`, `validarSenha`), garantindo a integridade da informação inserida.
- Vetores: A estrutura de dados principal. Vetores paralelos como `titulos`, `autores` e `emprestados` são usados para armazenar todo o acervo e a base de usuários, permitindo acesso direto aos dados via índice.
- Comandos condicionais (`if/else`): Condicionais foram usadas para direcionar o fluxo de `menu()` e, de forma crítica, na função `emprestimo()` para checar a disponibilidade do livro e a pendência do usuário, garantindo a estabilidade da transação.
- Comandos de repetição (`for`/`while`): Utilizados para processar listas e realizar buscas. O `for` é usado para percorrer o acervo nas consultas, e o `while` é empregado na função `emprestimo()` para buscar o usuário pelo e-mail e na função `compararStrings()`.
- Funções: O projeto implementou 8 funções além da `main()`, como `menu()`, `cadastrarLivro()`, `emprestimo()` e `limparBuffer()` e outras.

2.3 Estrutura de Dados

O acervo de livros e a base de usuários são mantidos em memória por meio da técnica de Vetores. Cada propriedade possui um vetor dedicado, onde o índice `i` em todos os vetores aponta para o mesmo registro. Variáveis como `contadorLivros` e `contadorUsuarios` gerenciam o tamanho real dos dados. A `bool pendencias` e a **`bool emprestados`** são *flags* binárias vitais para a lógica de empréstimo e para garantir a agilidade na checagem de status.

3. Implementação e Reflexão

3.1 Dificuldades e Soluções Implementadas

O desenvolvimento apresentou desafios práticos que exigiram soluções focadas na estabilidade e na agilidade do sistema:

- Lentidão e Erros de Entrada de Dados: O uso de `scanf` causava "lixo" no *buffer*, levando a erros operacionais e lentidão. A solução foi a implementação da função

auxiliar *limparBuffer()*, que consome o caractere de quebra de linha. Esta solução direta evita erros operacionais e garante que o sistema espere pela entrada correta do operador.

- Busca e Validação Lenta: A lógica de empréstimo foi desenhada para fazer as checagens de pendência e senha imediatamente após a busca, usando condicionais eficientes. Aumentando a velocidade de atendimento.

3.2 Conclusão e Aprendizados

O principal aprendizado foi a capacidade de traduzir a necessidade de resposta rápida em código estruturado. A utilização eficiente de comandos de repetição e condicionais na lógica de busca e validação demonstrou como o código pode otimizar o tempo de processamento, resolvendo as reclamações sobre a lentidão de sistemas. A experiência confirmou que modularidade e organização são a chave para a estabilidade.

3.3 Melhorias para as Próximas Unidades

As seguintes melhorias são propostas para expandir a funcionalidade e a robustez do sistema, focando em novas ferramentas para facilitar o uso diário:

- Refatoração para `structs` (Agrupamento Lógico): Prioridade máxima é substituir os vetores paralelos por `structs` de `Livro` e `Usuário`. Esta é uma ferramenta de simplificação que agrupa todos os dados de um item, facilitando a manipulação e reduzindo a chance de erros de programação.
- Entrada de Texto Robusta (Facilitação): Substituir todas as leituras de *string* com `scanf` por `fgets`. Esta é uma ferramenta de facilitação crucial na catalogação, pois permitirá que os operadores digitem títulos e nomes com espaços (ex: "A Volta ao Mundo em 80 Dias"), tornando a catalogação mais rápida e completa.
- Controle de Devolução com Penalidade (Rastreabilidade e Automação): Adicionar a funcionalidade de Devolução. Esta função incluirá a lógica para verificar o prazo e definir o *flag* `pendências` do usuário automaticamente, auxiliando o operador no controle de inadimplência de forma automatizada.
- Implementação de novas ferramentas para fazer assim um sistema mais completo.