RELATÓRIO TÉCNICO — UNIDADE 2

Sistema de Biblioteca em C

1. Introdução

O presente relatório descreve o desenvolvimento da segunda etapa do Sistema de Biblioteca criado para a disciplina Introdução às Técnicas de Programação. Diferentemente do que ocorreu na primeira unidade, o projeto não começou modularizado. Inicialmente, todo o código estava concentrado em poucos arquivos, de forma simples, pois o sistema ainda era pequeno.

Entretanto, conforme novas funcionalidades foram sendo adicionadas e o programa começou a crescer em tamanho e complexidade, tornou-se evidente que a estrutura original já não era suficiente. A ausência de modularização dificultava a manutenção, a leitura e a expansão do sistema.

Diante desse cenário, a modularização foi implementada nesta Unidade 2, reorganizando o código em múltiplos arquivos .c e .h e separando responsabilidades de maneira adequada. Além disso, esta etapa ampliou e refinou funcionalidades, incorporando conteúdos estudados como manipulação de strings, estruturas de repetição aninhadas, ponteiros e estratégias de organização em memória.

O sistema mantém seu objetivo central: permitir o gerenciamento de uma biblioteca por meio do terminal, com cadastro de usuários, cadastro de livros, empréstimos, devoluções, buscas e listagens completas. Contudo, nesta unidade, seu funcionamento tornou-se mais robusto, organizado e tecnicamente consistente.

2. Metodologia

O desenvolvimento foi realizado em linguagem C, utilizando o compilador GCC e um ambiente de edição simples, focado na compilação via terminal. A organização geral do código permaneceu modular, distribuída entre diversos arquivos .c e .h, cada um responsável por uma operação específica do sistema.

Ferramentas utilizadas:

- ➔ Compilador: GCC
- ➔ Editor: Code::Blocks / editor de texto

- ➔ Bibliotecas utilizadas: stdio.h, stdlib.h, string.h
- ➔ Sistema operacional: Windows

A abordagem seguiu a filosofia incremental: cada funcionalidade foi criada como módulo independente e testada isoladamente, antes de ser integrada ao main.c. Esse processo diminuiu erros e facilitou a depuração.
Para esta unidade, novos arquivos foram adicionados ao projeto, como busca.c, listagem.c, devolucao.c, converte_minusculas.c e apaga_registro.c. Além disso, funções antigas foram aperfeiçoadas para receber entradas mais robustas, validar dados com precisão e padronizar strings antes de comparações, evitando inconsistências comuns.

O código está dividido em módulos de acordo com suas responsabilidades:

- ➔ Entrada e validação: remove_nova_linha.c
- ➔ Manipulação de usuários: cadastra_usuario.c
- ➔ Manipulação de livros: cadastra_livro.c
- ➔ Empréstimos e devoluções: pega_emprestado.c, devolucao.c, insere_emprestados.c
- ➔ Listagem e busca: listagem.c, busca.c
- ➔ Utilidades: converte_minusculas.c, apaga_registro.c
- ➔ Tipos estruturados: tipos.h

Essa estrutura modular facilita alterações futuras e atende ao requisito de organização clara do código.

3. Análise do Código

3.1 Manipulação de Strings
Strings são parte essencial do sistema, usadas para representar nomes, CPFs, autores, títulos e identificadores. O trabalho com strings foi amplamente expandido na U2.

Operações realizadas com strings:

Captura de entrada do usuário com fgets()

Comparações com strcmp()
Cópias com strcpy()
Padronização de texto com converte_minusculas()
Remoção do caractere de nova linha com remove_nova_linha()

Funções desenvolvidas

remove_nova_linha() — remove o '\n' deixado por fgets()
converte_minusculas() — padroniza letras
apaga_registro() — limpa conteúdo para reorganização

Essas funções aumentam a precisão das comparações e evitam erros como divergência entre maiúsculas e minúsculas.

3.2 Estruturas de Repetição Aninhadas

O projeto utiliza diversos loops for e while, incluindo repetições aninhadas. Eles foram empregados para:

- ➔ Verificar duplicidade de CPF e ISBN
- ➔ Validar entradas numéricas
- ➔ Buscar usuários e livros no sistema
- ➔ Reorganizar o vetor de empréstimos após devoluções

A complexidade de algumas operações chega a $O(n^2)$, especialmente na reorganização dos empréstimos, o que ainda é adequado para o limite de 1000 registros.

3.3 Matrizes

Embora o conteúdo programático da U2 inclua matrizes, o projeto não utilizou estruturas bidimensionais. Vetores lineares de structs foram suficientes para representar usuários, livros e empréstimos, sem necessidade de uma matriz. Introduzir uma matriz traria complexidade desnecessária sem benefícios práticos para o sistema.

3.4 Ponteiros

O uso de ponteiros ocorre principalmente pela passagem de vetores de structs para funções. As funções recebem vetores como parâmetros, permitindo modificar diretamente os elementos originais sem retorno explícito.

Exemplos de uso:

- ➔ Atualização da quantidade de livros
- ➔ Incremento do número de empréstimos por usuário
- ➔ Remoção de registros do vetor de empréstimos
- ➔ Variáveis simples são passadas por valor, enquanto vetores são passados por referência implícita.

3.5 Alocação Dinâmica

O projeto não utilizou malloc ou free. As estruturas foram armazenadas em vetores estáticos com tamanho fixo de 1000 elementos. Essa decisão foi suficiente para o escopo atual do programa, evitando complexidade desnecessária.

A reorganização dos vetores, aliada a funções de limpeza, garantiu ausência de fragmentação lógica e eliminou a possibilidade de vazamento de memória.

4. Dificuldades e Soluções

As principais dificuldades encontradas foram:

1. Padronização de strings
Solução: criação da função converte_minusculas().
2. Validação de entrada
Solução: loops verificando caractere por caractere antes de aceitar entradas numéricas.
3. Reorganização do vetor de empréstimos
Solução: laços aninhados para deslocamento dos registros.
4. Verificação de dados em múltiplos módulos
Solução: maior modularização e reaproveitamento de código.
5. Organização modular adequada

Solução: separação das funções em múltiplos arquivos .c e .h.

5. Conclusão

A segunda etapa do projeto trouxe avanços significativos, ampliando funcionalidades e melhorando a organização interna do código. A implementação de módulos como busca, listagem e devolução tornou o sistema mais completo e próximo de uma aplicação real. A utilização de strings, vetores de structs, ponteiros indiretos e estruturas de repetição aninhadas garantiu eficiência e clareza ao projeto.

Em comparação à Unidade 1, houve melhorias importantes na validação de entradas, padronização de dados e modularização. A ausência de alocação dinâmica foi uma decisão consciente, coerente com o escopo previsto. O projeto demonstra evolução técnica consistente e domínio crescente dos fundamentos da linguagem C.