3 - SISTEMA EM LINGUAGEM DE PROGRAMAÇÃO C - LIVRARIA

Neste capítulo, a proposta de trabalho consistiu no desenvolvimento de um sistema para uma livraria, no âmbito da disciplina de Estrutura de Dados ministrada pelo professor Me. Júlio Fernando Lieira. O sistema foi implementado em linguagem C, utilizando a *Integrated Development Environment* (IDE) *Visual Studio Code,* versão 1.103.2, no sistema operacional Linux-Ubuntu.

Segundo Bertolini et al. (2019, p. 20), “a linguagem de programação C é uma linguagem de propósito geral, sendo adequada à programação estruturada. Ela é muito utilizada para escrever compiladores, analisadores léxicos, bancos de dados, editores de texto, entre outros programas.”

3.1 Bibliotecas

As bibliotecas são coleções de códigos prontos (funções, constantes, tipos de dados) que alguém já escreveu, testou e otimizou para você não ter que reinventar as instruções. Em C, se você quiser apenas mostrar uma mensagem na tela ou ler o que o usuário digitou, você precisa de uma biblioteca, porque a linguagem C pura, por si só, é extremamente minimalista.

> As bibliotecas em C constituem um conjunto de funções prontas que podem ser utilizadas para os mais devidos fins. Por exemplo, a biblioteca *stdio.h* é utilizada para funções de entrada e saída. No exemplo apresentado, utiliza-se o comando *printf,* responsável por exibir mensagens da tela. Esse comando está definido na biblioteca stdio.h. (BERTOLINI et al., 2019, p. 35).

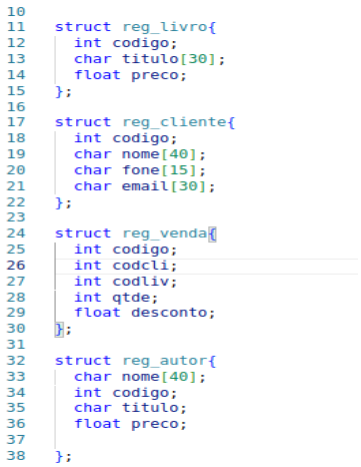
3.2 Conceito de Armazenamento Binário

O armazenamento binário consiste em gravar os dados no disco exatamente como estão representados na memória, sem conversões para texto. Cada campo da struct (inteiros, *floats*, *strings* etc.) é salvo em sua forma binária — isto é, em sequência de bytes correspondentes ao tipo de dado.

Isso difere do formato texto, que grava caracteres legíveis (por exemplo, o número 125 seria salvo como '1' '2' '5'), enquanto no modo binário o mesmo valor é armazenado diretamente como um inteiro de 4 *bytes*, conforme observado na **Figura 3.1**.

Na prática, essa abordagem torna o acesso mais rápido e compacto, permitindo que o programa leia ou grave um registro completo (*struct*) com um único comando.

Figura 3.1 – Representação da estrutura s*truct* em memória



```
10
11  struct reg_livro{
12    int codigo;
13    char titulo[30];
14    float preco;
15  };
16
17  struct reg_cliente{
18    int codigo;
19    char nome[40];
20    char fone[15];
21    char email[30];
22  };
23
24  struct reg_venda{
25    int codigo;
26    int codcli;
27    int codliv;
28    int qtde;
29    float desconto;
30  };
31
32  struct reg_autor{
33    char nome[40];
34    int codigo;
35    char titulo;
36    float preco;
37
38  };
```

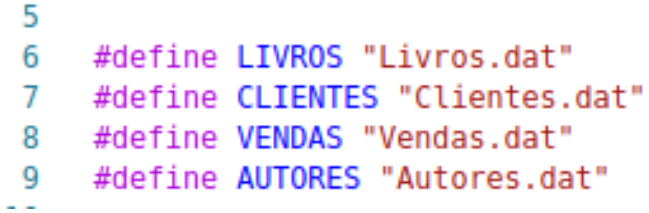
Fonte: Elaborado pelo autor (2025).

3.3 Estrutura e Representação dos Dados

Nesta seção, descreve-se o papel das *struct* no processo de armazenamento binário, destacando como cada registro em memória é convertido em bytes no arquivo .dat.

Cada entidade do sistema é representada por uma estrutura (*struct*) que contém os campos descritivos de cada registro. Ao gravar um registro com `fwrite()`, o compilador converte toda a estrutura diretamente em uma sequência de bytes binários, respeitando a organização interna da memória, conforme observado na **Figura 3.2**. O arquivo .dat não contém texto legível, mas sim dados binários brutos.

Figura 3.2 – Definição da estrutura de dados utilizando struct em linguagem C



```
5
6   #define LIVROS "Livros.dat"
7   #define CLIENTES "Clientes.dat"
8   #define VENDAS "Vendas.dat"
9   #define AUTORES "Autores.dat"
```

Fonte: Elaborado pelo autor (2025).

3.4 Inserção de Registro no Sistema

Durante a execução do sistema no terminal da IDE *Visual Studio Code*, o usuário realiza o cadastro de um novo livro, informando os dados solicitados pelo programa.

- “Código: 1”
- “Título: *Programação C*”
- “Preço: 100,00”

Ao confirmar a entrada, a função `cadastrarLivro()` é chamada internamente. Essa função coleta as informações digitadas, armazena os valores em uma variável do tipo *struct* Livro e grava os dados no arquivo binário Livros.dat por meio do comando: `fwrite(&livro, sizeof(Livro), 1, arquivo);`

O arquivo é aberto no modo *append* binário (“`ab`”), garantindo que o novo registro seja adicionado ao final do arquivo, sem sobrescrever os registros anteriores.

A partir desse ponto, o sistema grava no arquivo os valores de cada campo da estrutura em formato binário, de acordo com a sua representação interna na memória:

- O inteiro (código) é salvo como 4 *bytes*;
- O texto (título) é gravado como uma sequência de caracteres ASCII[1];
- O valor real (preço) é armazenado em 4 *bytes* correspondentes ao tipo *float.*

A **Figura 3.3** exibe o momento do cadastro de um livro no terminal, enquanto as **Figuras 3.4 e 3.5** ilustram a visualização do arquivo binário resultante, destacando as diferenças entre os dados legíveis e não legíveis.

1

1 A tabela ASCII (*American Standard Code for Information Interchange* – Código Padrão Americano para o Intercâmbio de Informação) define um conjunto de código numéricos para representar caracteres alfabéticos, numéricos e símbolos, permitindo que textos seja armazenado e interpretado por computadores.

Figura 3.3 – Cadastro de livro no terminal da IDE Visual Studio *Code*

```
#########   Livraria                  ###########
#                                               #
# 1) Cadastrar Livro                            #
# 111) Cadastrar Cliente                        #
# 2) Listar Todos os Livros                     #
# 222) Listar Todos os Clientes                 #
# 3) Consultar Livro pelo Codigo                #
# 4) Consultar Livro pelo Titulo                #
# 5) Consultar Livro por Palavra-Chave          #
# 6) Alterar Livro                              #
# 7) Excluir Livro                              #
# 8) Efetuar Venda                              #
# 9) Listar Vendas                              #
# 10) Relatorio de Vendas Detalhado             #
# 333) Cadastrar Autor                          #
# 444) listar Todos Autores                     #
# 555) Consultar Autor pelo Nome                #
# 666) Fechar Venda Caixa                       #
# 0) Sair                                       #
#                                               #
# Opcao-> 2

Relatorio Todos os Livros:
Codigo: 1
Titulo: Linguagem de programação C
Preco: 100.00
-----------------------------------------------
```

Fonte: Elaborado pelo autor (2025).

Para inspecionar esse conteúdo e verificar como os dados foram efetivamente armazenados, foram utilizados comandos do terminal *Linux* que permitem visualizar arquivos binários de maneira técnica. A **Figura 3.4,** exibe o conteúdo do arquivo em formato binário (0 e 1), permitindo observar os bits correspondentes a cada campo da *struct*. Esse comando mostra como os dados inteiros, caracteres e números reais são representados na memória secundária.

- Comando `xxd -b`

Figura 3.4 - Visualização do arquivo binário com comando xxd no terminal Linux

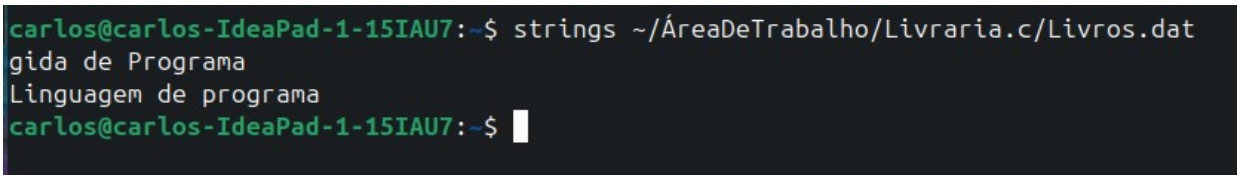
```
carlos@carlos-IdeaPad-1-15IAU7:~$ xxd -b ~/ÁreaDeTrabalho/Livraria.c/Livros.dat
00000000: 00000010 00000000 00000000 00000000 01001100 11000011  ....L.
00000006: 10110011 01100111 01101001 01100100 01100001 00100000  .gida
0000000c: 01100100 01100101 00100000 01010000 01110010 01101111  de Pro
00000012: 01100111 01110010 01100001 01101101 01100001 11000011  grama.
00000018: 10100111 11000011 10100011 01101111 00000000 00000000  ...o..
0000001e: 00000000 00000000 00000001 00000000 00000000 00000000  ......
00000024: 00000000 00000000 11001000 01000010 00000001 00000000  ...B..
0000002a: 00000000 00000000 01001100 01101001 01101110 01100111  ..Ling
00000030: 01110101 01100001 01100111 01100101 01101101 00100000  uagem
00000036: 01100100 01100101 00100000 01110000 01110010 01101111  de pro
0000003c: 01100111 01110010 01100001 01101101 01100001 11000011  grama.
00000042: 10100111 11000011 10100011 01101111 00100000 01000011  ...o C
00000048: 00000000 00000000 00000000 00000000 00000000 00000000  ......
0000004e: 11001000 01000010                                      .B
```

Fonte: Elaborado pelo autor (2025).

A **Figura 3.5** exibe apenas as sequências de caracteres legíveis (ASCII) contidas no arquivo binário. Assim, foi possível identificar os textos armazenados – como título e autor – enquanto os valores numéricos permanecem em formato não legível, confirmando que o arquivo contém informações em nível binário.

- Comando `strings`

Figura 3.5 – Visualização de strings do arquivo binário utilizando comando strings



Fonte: Elaborado pelo autor (2025).

Essas análises demonstram, de forma prática, que o armazenamento utilizado pelo sistema é, de fato, binário: apenas os dados de texto aparecem de maneira legível, enquanto os valores inteiros e reais são representados por *bytes* binários puros.

3.5 Relação com Estruturas em Memória

O funcionamento do arquivo binário assemelha-se ao de uma lista sequencial, qual cada *struct* representa um nó fixo. A posição física no arquivo substitui o ponteiro de ligação, e os dados permanecem persistidos em disco.

3.6 Armazenamento Binário e Create, Read, Update e Delete (CRUD) em Linguagem C

3.6.1 *CREATE* – Cadastrar é responsável por cadastrar novos registros.

As funções `cadastrarLivro()`, `cadastrarCliente()` e `efetuarVenda()` coletam os dados e gravam no arquivo com `fwrite(&registro;, sizeof(registro), 1, arquivo)`. O modo `'ab'` (`append binary`) adiciona o registro no final do arquivo, caracterizando a operação Create do CRUD. Usa `fopen(..., 'ab')` e `fwrite()` para adicionar registros.

3.6.2 *READ* – Consultar e Listar tem função de consultar e listar os registros.

As funções `listarLivros()`, `listarClientes()`, `consultarPeloCodigo()` e `consultarPeloTitulo()` abrem o arquivo em modo `'rb'` (`read binary`) e utilizam `fread()` para ler um registro completo por vez. Cada leitura reconstrói a struct na memória, permitindo exibição dos dados na tela. Usa `fopen(..., 'rb')` e `fread()` para percorrer os registros.

3.6.3 ***UPDATE*** **– Alterar** se encarrega de realizar as alterações dos registros existentes.

A função `alterarLivro()` usa o modo `'rb+'` (leitura e escrita binária) e percorre o arquivo com `fread()` até encontrar o registro desejado. Em seguida, reposiciona o ponteiro com `fseek()` e sobrescreve o conteúdo com `fwrite()`, atualizando apenas aquele registro. Usa `fopen(..., 'rb+')` com `fseek()` para reposicionar e `fwrite()` para regravar.

3.6.4 ***DELETE*** **– Excluir** se designa a excluir o registro existente.

A função `excluirLivro()` recria o arquivo sem o registro escolhido. Os registros válidos são copiados para um novo arquivo, o antigo é excluído e o novo é renomeado. Essa é a operação Delete do *CRUD*.

Usando:

`system("del livros.dat");` Excluir o arquivo antigo `livros.dat` do sistema.

`system("ren livrosnew.dat livros.dat");` Renomear o novo arquivo (`livrosnew.dat`) para (`livros.dat`).

3.7 Considerações sobre Persistência

No modelo atual, utilizando arquivos binários, os dados são armazenados de forma duradoura no disco, garantindo que permaneçam disponíveis mesmo após o encerramento do programa. Cada registro é gravado em sua representação binária na memória, o que permite a leitura e a escrita consistentes de diferentes tipos de dados (inteiros, *floats*, *strings*).

Para operações de exclusão ou atualização, o sistema cria arquivos temporários e copia os registros necessários, garantindo integridade e segurança, pois o arquivo original não é alterado até que a operação seja concluída com sucesso.

Embora a manipulação com ponteiros seja essencial em listas e pilhas para gerenciar a memória dinamicamente, no modelo de arquivos binários, a persistência é garantida pelo próprio armazenamento em disco, tornando o sistema independente da memória volátil.

3.8 Struct

Após compreender o funcionamento das *structs* no armazenamento binário, está seção apresenta as estruturas de dados utilizadas no código-fonte do sistema, detalhando suas definições e finalidades.

> Registros são estruturas de dados que permitem armazenar em uma única variável um conjunto de variáveis diferentes. Em linguagem de programação chamamos estas estruturas de registro, no entanto, na linguagem C os registros chamam-se *struct. (*BERTOLINI et al., 2019, p.87).

Nas linhas 11 a 38 são definidas as *struct* que são usadas para organizar os dados relacionados: livros, clientes, vendas e autores. As *structs* permitem armazenar várias informações em uma única variável, facilitando a manipulação e armazenamento em arquivos.

3.9 Main

Nesta etapa a função “`main ()`” **Figura 3.6** que é a função principal do programa em “C”, ela é responsável por iniciar a execução, controlar o fluxo principal da aplicação e chamar todas as funções auxiliares necessárias. Ela pode retornar um valor inteiro ao sistema operacional, indicando o sucesso ou falha da execução. Nesta etapa, o código totaliza 772 linhas implementadas.

Figura 3.6 – Estrutura da função *main* do sistema.

```
747  main(){
748    int op;
749
750    do {
751      printf("\n\n\n");
752      printf("\n#########   Livraria Alma do Saber   ##########");
753      printf("\n#                                             #");
754      printf("\n# 1) Cadastrar Livro                         #");
755      printf("\n# 111) Cadastrar Cliente                     #");
756      printf("\n# 2) Listar Todos os Livros                  #");
757      printf("\n# 222) Listar Todos os Clientes             #");
758      printf("\n# 3) Consultar Livro pelo Codigo            #");
759      printf("\n# 4) Consultar Livro pelo Titulo            #");
760      printf("\n# 5) Consultar Livro por Palavra-Chave      #");
761      printf("\n# 6) Alterar Livro                          #");
762      printf("\n# 7) Excluir Livro                          #");
763      printf("\n# 8) Efetuar Venda                         #");
764      printf("\n# 9) Listar Vendas                         #");
765      printf("\n# 10) Relatorio de Vendas Detalhado        #");
766      printf("\n# 333) Cadastrar Autor                     #");
767      printf("\n# 444) listar Todos Autores               #");
768      printf("\n# 555) Consultar Autor pelo Nome          #");
769      printf("\n# 666) Fechar Venda Caixa                  #");
770      printf("\n# 0) Sair                                  #");
771      printf("\n#                                         #");
772      printf("\n# Opcao-> ");
```

Fonte: Elaborado pelo autor (2025).

Dessa forma, a função `main()` consolida toda a lógica desenvolvida ao longo do código, integrando as funcionalidades implementadas e garantindo o correto funcionamento do sistema como um todo.

3.10 Considerações Técnicas

Neste capítulo, pôde-se observar como se representa um sistema de cadastro simples aplicado a uma livraria, no qual foram desenvolvidas funcionalidades voltadas ao gerenciamento de livros e clientes. O código cumpre bem seu papel em organizar dados de forma estruturada e interativa, usando conceitos fundamentais de lógica e programação. Além disso, observa-se que a aplicação não se limita apenas a um CRUD tradicional, incorporando operações de consultas tornando-se mais completo.

O desenvolvimento do sistema teve como principal propósito aprofundar o aprendizado sobre estruturas de dados e persistência em arquivos binários, permitindo compreender como as informações podem ser armazenadas diretamente em formato de bytes, representando fielmente o conteúdo das estruturas (structs) na memória. Essa abordagem proporcionou uma experiência prática no tratamento de dados persistentes sem uso de banco de dados externos, reforçando o entendimento sobre leitura, gravação e manipulação de registros de forma eficiente.