# Projeto de Biblioteca - Algoritmos e Estruturas de Dados

O enunciado pede que você construa, em C, um sistema de cadastro de livros de biblioteca utilizando lista encadeada persistida em arquivo binário, com usuários e empréstimos.

## 1. Tipos de registro e arquivos binários

### a) Registros

Você vai precisar de três estruturas principais em memória (em C, struct):

1. **Livro**

* typedef struct {  
   int codigo;  
   char titulo[151];  
   char autor[201];  
   char editora[51];  
   int edicao;  
   int ano;  
   int exemplares;  
   long prox; // ponteiro para o próximo nó (posição no arquivo)  
  } Livro;

1. **Usuário**

* typedef struct {  
   int codigo;  
   char nome[51];  
   long prox; // para lista encadeada de usuários (opcional)  
  } Usuario;

1. **Empréstimo**

* typedef struct {  
   int cod\_usuario;  
   int cod\_livro;  
   char data\_emp[11]; // "DD/MM/AAAA\0"  
   char data\_dev[11]; // vazio se ainda não devolvido  
   long prox; // lista encadeada de empréstimos  
  } Emprestimo;

### b) Cabeçalho do arquivo

No início de cada arquivo binário deve ficar um registro de cabeçalho contendo:

* long pos\_cabeca; → posição do primeiro nó válido
* long pos\_topo; → posição do próximo espaço livre no final do arquivo

Isso permite percorrer a lista via prox e reaproveitar posições livres.

## 2. Operações obrigatórias e como implementá-las

No main(), apresente um menu em loop com scanf() para ler opção; para cada opção, chame a função correspondente.

### Cadastrar livro

* Peça código, título, autor, editora, edição, ano, exemplares.
* Crie o struct Livro, posicione-se em pos\_topo, fwrite, atualize:
  + novo.prox = antigo pos\_cabeca
  + cabeçalho.pos\_cabeca = pos\_topo
  + cabeçalho.pos\_topo += sizeof(Livro)

### Imprimir dados do livro

* Leia código, percorra a lista (usando prox) até achar livro.codigo == código.
* Se achar, printf de todos os campos; senão, “Livro não cadastrado”.

### Listar todos os livros

* Varra a lista e exiba apenas codigo, titulo, autor e exemplares de cada livro.

### Busca por título

* Leia string, percorra e use strcmp/strcasestr para encontrar correspondência.
* Imprima dados completos.

### Calcular total

* Conte quantos nós existem na lista e imprima o total.

### Cadastrar usuário

* Análogo ao cadastrar livro, usando struct Usuario e outro arquivo/lógica.

### Emprestar livro

* Leia códigos do usuário e do livro.
* Valide que usuário existe e exemplares > 0.
* Se ok:
  + Crie Emprestimo com data atual (time, localtime, strftime)
  + Insira na lista de empréstimos como fizemos para livros
  + Atualize exemplares do livro (–1) e escreva de volta

### Devolver livro

* Leia códigos, procure empréstimo sem data de devolução que case ambos os códigos.
* Se achar:
  + Preencha data\_dev com data atual
  + Grave de volta empréstimo
  + Atualize exemplares do livro (+1)

### Listar livros emprestados

* Varra todos os empréstimos e exiba apenas os que têm data\_dev vazia, mostrando:
* Código usuário | Nome usuário | Código livro | Título livro | Data empréstimo

### Carregar arquivo texto (lote)

* Abra o .txt, leia linha a linha com fgets, use strtok(line, ";") para separar campos.
* Aplique trim() em cada token e, conforme o primeiro token:
  + "L" → chama inserir livro
  + "U" → inserir usuário
  + "E" → inserir empréstimo (data de devolução vazia → ainda não devolvido)

## 3. Detalhes que às vezes geram dúvida

* **Lista encadeada em arquivo**: cada nó armazena prox (long) para próximo nó.
* **Trim**: remove espaços em branco no início/fim de cada string lida.
* **Data atual**:
* time\_t t = time(NULL);  
  struct tm \*hoje = localtime(&t);  
  strftime(emp.data\_emp, sizeof(emp.data\_emp), "%d/%m/%Y", hoje);
* **Documentação**: em .c e .h, documente cada função com propósito, pré e pós-condições:
* /\*\*  
   \* insereLivro:  
   \* Propósito: insere um novo livro na lista encadeada persistida em arquivo.  
   \* Pré-condições: arquivo aberto em modo "rb+"; parâmetros válidos.  
   \* Pós-condições: livro gravado no arquivo; cabeçalho atualizado.  
   \*/  
  void insereLivro(...);

## 4. Plano de ação sugerido

1. Definir structs (Livro, Usuário, Empréstimo, Cabeçalho).
2. Implementar utilitários de arquivo: abre, lê/grava nós e cabeçalho.
3. Funções de inserção, busca e listagem para livros e usuários.
4. Funções de empréstimo e devolução, atualizando exemplares e datas.
5. Função de carga em lote a partir de arquivo texto.
6. Menu em main(), chamando cada função.
7. Testes manuais:
   * Cadastre livros e usuários
   * Empreste e devolva, conferindo exemplares
   * Carregue lote e verifique resultados