Com base no enunciado, você precisa desenvolver um sistema de gerenciamento de biblioteca em C que utiliza **listas encadeadas implementadas em arquivos binários**. Vou explicar passo a passo o que precisa ser feito, destacando os pontos críticos:

### ****1. Estruturas de Dados e Arquivos Binários****

Você precisará de **3 arquivos binários** distintos:

1. livros.bin → Armazena registros de livros
2. usuarios.bin → Armazena registros de usuários
3. emprestimos.bin → Armazena registros de empréstimos

Cada arquivo deve ter uma **estrutura de lista encadeada** com:

* **Cabeçalho**: Primeiros bytes do arquivo com:

c

* typedef struct {

int head; // Posição do primeiro registro válido (ou -1 se vazio)

int top; // Próxima posição livre no fim do arquivo (offset)

} Header;

* **Registros**: Estruturas com campo next (próximo nó):

c

* // Para livros

typedef struct {

int next; // Próximo nó na lista

int codigo;

char titulo[151];

char autor[201];

char editora[51];

int edicao;

int ano;

int exemplares;

} Livro;

// Para usuários

typedef struct {

int next;

int codigo;

char nome[51];

} Usuario;

// Para empréstimos

typedef struct {

int next;

int codigo\_usuario;

int codigo\_livro;

char data\_emprestimo[11]; // "DD/MM/AAAA"

char data\_devolucao[11]; // Vazio se não devolvido

} Emprestimo;

### ****2. Funcionalidades Obrigatórias****

Implemente um menu com as seguintes opções:

| Opção | Ação |
| --- | --- |
| **Cadastrar livro** | Insere novo registro em livros.bin, atualizando cabeçalho e lista encadeada |
| **Imprimir livro** | Busca em livros.bin por código e exibe todos os campos |
| **Listar todos** | Percorre a lista encadeada de livros.bin exibindo código, título, autor e exemplares |
| **Buscar por título** | Percorre livros.bin comparando títulos (case-insensitive) |
| **Calcular total** | Conta registros em livros.bin durante o percurso da lista |
| **Cadastrar usuário** | Similar ao cadastro de livros, mas em usuarios.bin |
| **Emprestar livro** | Verifica disponibilidade, decrementa exemplares, adiciona em emprestimos.bin com data atual |
| **Devolver livro** | Atualiza data\_devolucao e incrementa exemplares no livro |
| **Listar empréstimos** | Percorre emprestimos.bin exibindo apenas registros com data\_devolucao vazia |
| **Carregar arquivo** | Lê arquivo texto no formato especificado e insere em lote nos binários |

### ****3. Pontos Críticos de Implementação****

#### a) ****Lista Encadeada em Arquivo****

* **Inserção**:

c

* // Pseudocódigo para inserir em livros.bin

Abrir arquivo (r+b)

Ler cabeçalho

Mover para posição header->top

Escrever novo livro com next = -1

Se lista vazia (head == -1):

header->head = header->top

Senão:

Percorre lista até último nó (next == -1)

Atualiza next do último nó para header->top

Atualiza header->top += sizeof(Livro)

Escrever cabeçalho atualizado

Fechar arquivo

#### b) ****Carregamento de Lote****

* Formato do arquivo texto:

text

* L;7;Memórias Póstumas de Brás Cubas;Machado de Assis;Bookman;4;2022;5

U;11;João da Silva

E;5;27;03/02/2025;15/02/2025

* Passos:
  1. Ler linha por linha
  2. Remover espaços extras com trim()
  3. Usar strtok() para dividir por ;
  4. Converter strings para tipos adequados (ex: atoi() para números)
  5. Chamar funções de inserção nos binários

#### c) ****Validações Cruciais****

* **Empréstimo**:

c

* if (livro.exemplares > 0) {

// Permite empréstimo

} else {

printf("Não há exemplares disponíveis\n");

}

* **Datas**: Usar strftime() para gerar data atual no formato "DD/MM/AAAA"
* **Chaves únicas**: Verificar se código já existe durante inserção

### ****4. Estratégia Recomendada****

#### a) Divisão de Código

| Arquivo | Responsabilidade |
| --- | --- |
| main.c | Menu principal e controle de fluxo |
| livro.c | Funções para CRUD de livros (inserir, buscar, listar) |
| usuario.c | CRUD de usuários |
| emprestimo.c | Lógica de empréstimos/devoluções e listagem |
| arquivo.c | Manipulação de arquivos binários (cabeçalhos, inserção genérica) |
| utils.c | Funções auxiliares (trim, data atual, parsing) |

#### b) Cronograma Sugerido (9 dias)

| Dia | Tarefas |
| --- | --- |
| 1 | Estruturas de dados + funções básicas de arquivos binários |
| 2-3 | Implementar CRUD para livros e usuários |
| 4 | Gerenciamento de empréstimos (com validações) |
| 5 | Sistema de menu interativo |
| 6 | Carregamento de arquivos texto (parser de lote) |
| 7 | Funções de listagem e consultas |
| 8 | Testes com dados reais e correção de bugs |
| 9 | Documentação do código + relatório (README.pdf) |

### ****5. Dicas Técnicas Essenciais****

* **Manipulação de Arquivos**:

c

* FILE \*f = fopen("livros.bin", "r+b");

fseek(f, offset, SEEK\_SET);

fread(&reg, sizeof(Livro), 1, f);

* **Remoção de Espaços**:

c

* void trim(char \*str) {

int i = 0, j = 0;

while (str[i]) {

if (!isspace(str[i])) str[j++] = str[i];

i++;

}

str[j] = '\0';

}

* **Data Atual**:

c

* #include <time.h>

void data\_atual(char \*buffer) {

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

sprintf(buffer, "%02d/%02d/%04d", tm.tm\_mday, tm.tm\_mon+1, tm.tm\_year+1900);

}

### ****6. Entregas Obrigatórias****

1. **Código Fonte** (.c e .h) com:
   * Documentação em formato Doxygen (pré/pós condições)
   * Organização modular
2. **README.pdf** contendo:
   * Nomes dos integrantes
   * Manual rápido de uso
   * Explicação das estruturas

### ****7. Como Testar****

Crie scripts com casos de teste:

bash

# Exemplo de teste para empréstimo

echo -e "E;7;13;$(date +'%d/%m/%Y');\n" > teste.txt

./biblioteca -lote teste.txt

Este é um projeto complexo, mas seguindo essa estrutura você conseguirá entregar dentro do prazo! Comece pelos arquivos binários e CRUD básico, depois implemente o menu.