MAPA – Material de Avaliação Prática da Aprendizagem

Nome: Timóteo Dos Reis Nogueira	**R.A 25155923-5**
Curso: Engenharia de software	
Disciplina: Linguagem e técnicas de programação	
Valor da atividade: 5,00	**Prazo: 07/12/2025 23:59**

Instruções para Realização da Atividade

1. Revise seu arquivo antes do envio. Certifique-se de que é o arquivo correto, formato correto, se contempla todas as demandas da atividade, etc.
2. Após o envio não serão permitidas alterações.
3. Durante a disciplina, procure sanar suas dúvidas pontuais em relação ao conteúdo relacionado à atividade. Porém, não são permitidas correções parciais, ou seja, enviar para que o professor possa fazer uma avaliação prévia e retornar para que o aluno possa ajustar e enviar novamente. Isso não é permitido, pois descaracteriza o processo de avaliação.
4. Ao utilizar quaisquer materiais de pesquisa referencie conforme as normas da ABNT.

Em caso de dúvidas, entre em contato com seu Professor Mediador.

Bons estudos!

IMPORTANTE:

1. Acesse o link com um vídeo tutorial para ajudá-lo nesse processo de criação e desenvolvimento. O acesso deverá ser realizado através do fórum interativo: "Links das Aulas ao Vivo".
2. Disserte a respeito do tema, seguindo, como roteiro, os tópicos elencados.

3. A entrega deve ser feita exclusivamente usando o template de entrega da atividade MAPA, disponível no material da disciplina.
4. Antes de enviar a sua atividade, certifique-se de que respondeu a todas as perguntas e realize uma cuidadosa correção ortográfica.
5. Após o envio, não são permitidas alterações ou modificações. Logo, você tem apenas uma chance de enviar o arquivo corretamente. Revise bem antes de enviar!
6. Lembre-se de que evidências de cópias de materiais, incluindo de outros acadêmicos, sem as devidas referências, serão inquestionavelmente zeradas. As citações e as referências, mesmo que do livro da disciplina, devem ser realizadas de acordo com as normas da Instituição de Ensino.
7. Não são permitidas correções parciais no decorrer do módulo, ou seja, o famoso: “professor, veja se minha atividade está certa?”. Isso invalida o seu processo avaliativo. Lembre-se de que a interpretação da atividade também faz parte da avaliação.
8. Procure sanar as suas dúvidas junto à mediação em tempo hábil sobre o conteúdo exigido na atividade, de modo que consiga realizar a sua participação.
9. Atenção ao prazo de entrega. Evite o envio da atividade muito próximo do prazo. Você pode ter algum problema com a internet, o computador, o software etc., e os prazos não serão flexibilizados, mesmo em caso de comprovação.

Bons estudos!

Em caso de dúvidas, encaminhe mensagem ao seu professor mediador.

Resposta atividade MAPA:

MANUAL DO USUÁRIO

1) Cadastrar Livro

Esta função adiciona um novo livro ao acervo

O sistema exigira:

- Código do livro
- Título
- Autor
- Editora
- Ano de publicação
- Quantidade de exemplares

Após o preencher os requisitos, o livro será salvo automaticamente

2) Cadastrar Usuário

Cadastra um novo usuário da biblioteca.

O sistema exigira:

- Matrícula
- Nome completo
- Curso
- Telefone
- Data de cadastro

Após o preencher os requisitos, o usuário fica disponível para empréstimos imediatamente.

3) Realizar Empréstimo

Registra o empréstimo de um livro

O sistema exigira:

- Matrícula do usuário
- Código do livro
- Data do empréstimo

IMPORTANTE:

Para a realização do empréstimo o:

- O usuário deve estar cadastrado e o livro deve estar disponível

Após o preencher os requisitos, o empréstimo será criado e o livro passará para o status “emprestado”

A data de devolução é calculada automaticamente (7 dias depois)

4) Realizar Devolução

Finaliza um empréstimo ativo

O sistema exigira o código do empréstimo

Ao devolver:

- O status do empréstimo muda para “devolvido”
- O livro retorna ao status “disponível”

5) Renovar Empréstimo

Permite estender o prazo de um empréstimo ativo

O sistema exigira:

- Código do empréstimo
- A devolução prevista é estendida por mais 7 dias

6) Pesquisar Livro

Consulta informações de um livro.

O sistema exigira:

- Código do livro

O sistema mostrara:

- Título
- Autor
- Editora
- Ano
- Status (disponível/emprestado)

Se o código não existir, uma mensagem de erro aparece

7) Pesquisar Usuário

Exibe informações de um usuário

O sistema exigira:

- Matricula

O sistema exibira:

- Nome
- Curso
- Telefone

Se não estiver cadastrado, informa que o usuário não foi encontrado

8) Listar Empréstimos Ativos

Exibe todos os empréstimos não devolvidos

O sistema exibira:

- Código do empréstimo
- Matrícula do usuário
- Código do livro
- Data prevista de devolução

Se não houver nenhum empréstimo ativo, exibe uma mensagem correspondente

9) Relatório de Livros Mais Emprestados

Gera uma lista dos livros mais emprestados ao longo do tempo

O sistema exibira:

- Código do livro
- Quantidade de empréstimos realizados
- Ideal para análises da biblioteca

10) Relatório de Usuários em Atraso

Lista todos os usuários que deveriam ter devolvido livros e não devolveram

O sistema exigira:

- Data atual

O sistema exibira:

- Nome do usuário
- Matrícula
- Livro emprestado
- Data em que o prazo venceu

0) Sair

Encerra o programa

CASO NECESSÁRIO: LINK DO CODIGO EM ZIP E DESCOMPACTADO:

https://drive.google.com/drive/folders/14QRmEiVtwjv8cps2i779Y9TkPpr1UgcH?usp=sharing

CODIGO:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Struct simples pra datas
typedef struct {
    int dia, mes, ano;
} Data;

// Struct do livro
typedef struct {
    int codigo;
    char titulo[101];
    char autor[81];
    char editora[61];
    int ano_pub;
    int exemplares;
    char status[20]; // disponivel / emprestado
} Livro;

// Struct do usuário
typedef struct {
    int matricula;
    char nome[101];
```

```
    char curso[51];
    char telefone[16];
    Data cadastro;
} Usuario;

// Struct do empréstimo
typedef struct {
    int codigo;
    int matricula_usuario;
    int codigo_livro;
    Data emprestimo;
    Data devolucao_prevista;
    char status[20]; // ativo / devolvido
} Emprestimo;

// Constantes
#define MAX_LIVROS 100
#define MAX_USUARIOS 100
#define MAX_EMPRESTIMOS 100

// Inicialização
void inicializarLivros(Livro livros[], int *numLivros);
void inicializarUsuarios(Usuario usuarios[], int *numUsuarios);
void inicializarEmprestimos(Emprestimo emprestimos[], int *numEmprestimos);

// Cadastro
void cadastrarLivro(Livro livros[], int *numLivros);
void cadastrarUsuario(Usuario usuarios[], int *numUsuarios);

// Empréstimo e devolução
void realizarEmprestimo(Livro livros[], Usuario usuarios[], Emprestimo
emprestimos[],
```

```
                int *numLivros, int *numUsuarios, int *numEmprestimos);

void realizarDevolucao(Livro livros[], Emprestimo emprestimos[],
                int *numLivros, int *numEmprestimos);

void renovarEmprestimo(Emprestimo emprestimos[], int *numEmprestimos);

// Pesquisa
void pesquisarLivro(Livro livros[], int numLivros);
void pesquisarUsuario(Usuario usuarios[], int numUsuarios);
void listarEmprestimosAtivos(Emprestimo emprestimos[], int numEmprestimos);

// Relatórios
void relatorioLivrosMaisEmprestados(Emprestimo emprestimos[], int
numEmprestimos);
void relatorioUsuariosAtraso(Emprestimo emprestimos[], Usuario usuarios[],
                int numEmprestimos, int numUsuarios, Data dataAtual);

// Arquivos
void salvarLivros(Livro livros[], int numLivros);
void salvarUsuarios(Usuario usuarios[], int numUsuarios);
void salvarEmprestimos(Emprestimo emprestimos[], int numEmprestimos);

void carregarLivros(Livro livros[], int *numLivros);
void carregarUsuarios(Usuario usuarios[], int *numUsuarios);
void carregarEmprestimos(Emprestimo emprestimos[], int *numEmprestimos);

// Auxiliares
Data calcularDataDevolucao(Data emprestimo);
int compararDatas(Data d1, Data d2);
void definirDataAtual(Data *dataAtual);
```

```
// Lê string com segurança e remove o \n
void lerString(char *dest, int tam) {
    fgets(dest, tam, stdin);
    dest[strcspn(dest, "\n")] = '\0';
}

// ----------------------------
// Inicialização
// ----------------------------

void inicializarLivros(Livro livros[], int *numLivros) {
    *numLivros = 0;
    carregarLivros(livros, numLivros);
}

void inicializarUsuarios(Usuario usuarios[], int *numUsuarios) {
    *numUsuarios = 0;
    carregarUsuarios(usuarios, numUsuarios);
}

void inicializarEmprestimos(Emprestimo emprestimos[], int *numEmprestimos) {
    *numEmprestimos = 0;
    carregarEmprestimos(emprestimos, numEmprestimos);
}

// ----------------------------
// Cadastro de Livros
// ----------------------------

void cadastrarLivro(Livro livros[], int *numLivros) {
    if (*numLivros >= MAX_LIVROS) {
        printf("Limite de livros atingido!\n");
```

```
    return;
}

Livro l;

printf("Código do livro: ");
scanf("%d", &l.codigo);
getchar(); // limpa buffer

printf("Título: ");
lerString(l.titulo, sizeof(l.titulo));

printf("Autor: ");
lerString(l.autor, sizeof(l.autor));

printf("Editora: ");
lerString(l.editora, sizeof(l.editora));

printf("Ano de publicação: ");
scanf("%d", &l.ano_pub);
getchar();

printf("Quantidade de exemplares: ");
scanf("%d", &l.exemplares);
getchar();

strcpy(l.status, "disponivel");

livros[*numLivros] = l;
(*numLivros)++;

salvarLivros(livros, *numLivros);
```

```
    printf("Livro cadastrado!\n");
}

// ----------------------------
// Cadastro de Usuários
// ----------------------------

void cadastrarUsuario(Usuario usuarios[], int *numUsuarios) {
    if (*numUsuarios >= MAX_USUARIOS) {
        printf("Limite de usuários atingido!\n");
        return;
    }

    Usuario u;

    printf("Matrícula: ");
    scanf("%d", &u.matricula);
    getchar();

    printf("Nome completo: ");
    lerString(u.nome, sizeof(u.nome));

    printf("Curso: ");
    lerString(u.curso, sizeof(u.curso));

    printf("Telefone: ");
    lerString(u.telefone, sizeof(u.telefone));

    printf("Data de cadastro (dia mes ano): ");
    scanf("%d %d %d", &u.cadastro.dia, &u.cadastro.mes, &u.cadastro.ano);
    getchar();
```

```
    usuarios[*numUsuarios] = u;
    (*numUsuarios)++;

    salvarUsuarios(usuarios, *numUsuarios);

    printf("Usuário cadastrado!\n");
}


// ----------------------------
// Empréstimos
// ----------------------------

void realizarEmprestimo(Livro livros[], Usuario usuarios[], Emprestimo
emprestimos[],
                        int *numLivros, int *numUsuarios, int *numEmprestimos) {

    if (*numEmprestimos >= MAX_EMPRESTIMOS) {
        printf("Limite de empréstimos atingido!\n");
        return;
    }

    int mat, cod;

    printf("Matrícula do usuário: ");
    scanf("%d", &mat);
    getchar();

    printf("Código do livro: ");
    scanf("%d", &cod);
    getchar();
```

```
// Procura usuário
int posUsuario = -1;
for (int i = 0; i < *numUsuarios; i++) {
    if (usuarios[i].matricula == mat) {
        posUsuario = i;
        break;
    }
}

if (posUsuario == -1) {
    printf("Usuário não encontrado!\n");
    return;
}

// Procura livro
int posLivro = -1;
for (int i = 0; i < *numLivros; i++) {
    if (livros[i].codigo == cod && strcmp(livros[i].status, "disponivel") == 0) {
        posLivro = i;
        break;
    }
}

if (posLivro == -1) {
    printf("Livro não está disponível!\n");
    return;
}

// Criando empréstimo
Emprestimo e;
e.codigo = (*numEmprestimos) + 1;
```

```
    e.matricula_usuario = mat;
    e.codigo_livro = cod;

    printf("Data do empréstimo (dia mes ano): ");
    scanf("%d %d %d", &e.emprestimo.dia, &e.emprestimo.mes, &e.emprestimo.ano);
    getchar();

    // Calcula devolução prevista
    e.devolucao_prevista = calcularDataDevolucao(e.emprestimo);

    strcpy(e.status, "ativo");

    emprestimos[*numEmprestimos] = e;
    (*numEmprestimos)++;

    // Marca livro como emprestado
    strcpy(livros[posLivro].status, "emprestado");

    salvarLivros(livros, *numLivros);
    salvarEmprestimos(emprestimos, *numEmprestimos);

    printf("Empréstimo feito!\n");
}

// ----------------------------
// Devolução
// ----------------------------

void realizarDevolucao(Livro livros[], Emprestimo emprestimos[],
                       int *numLivros, int *numEmprestimos) {

    int cod;
```

```
    printf("Código do empréstimo: ");
    scanf("%d", &cod);
    getchar();

    int posEmp = -1;

    // Acha empréstimo ativo
    for (int i = 0; i < *numEmprestimos; i++) {
        if (emprestimos[i].codigo == cod && strcmp(emprestimos[i].status, "ativo") == 0)
{
            posEmp = i;
            break;
        }
    }

    if (posEmp == -1) {
        printf("Empréstimo não encontrado!\n");
        return;
    }

    strcpy(emprestimos[posEmp].status, "devolvido");

    // Libera o livro
    for (int i = 0; i < *numLivros; i++) {
        if (livros[i].codigo == emprestimos[posEmp].codigo_livro) {
            strcpy(livros[i].status, "disponivel");
            break;
        }
    }

    salvarLivros(livros, *numLivros);
```

```
    salvarEmprestimos(emprestimos, *numEmprestimos);

    printf("Devolução feita!\n");
}

// ----------------------------
// Renovação
// ----------------------------

void renovarEmprestimo(Emprestimo emprestimos[], int *numEmprestimos) {

    int cod;

    printf("Código do empréstimo: ");
    scanf("%d", &cod);
    getchar();

    int pos = -1;

    for (int i = 0; i < *numEmprestimos; i++) {
        if (emprestimos[i].codigo == cod && strcmp(emprestimos[i].status, "ativo") == 0)
{
            pos = i;
            break;
        }
    }

    if (pos == -1) {
        printf("Empréstimo não encontrado!\n");
        return;
    }
```

```
    // Soma mais 7 dias
    emprestimos[pos].devolucao_prevista =
        calcularDataDevolucao(emprestimos[pos].devolucao_prevista);

    printf("Empréstimo renovado!\n");
}
// ----------------------------
// Pesquisar Livro
// ----------------------------

void pesquisarLivro(Livro livros[], int numLivros) {
    int cod;

    printf("Código do livro: ");
    scanf("%d", &cod);
    getchar();

    for (int i = 0; i < numLivros; i++) {
        if (livros[i].codigo == cod) {

            printf("\n--- Livro Encontrado ---\n");
            printf("Título: %s\n", livros[i].titulo);
            printf("Autor: %s\n", livros[i].autor);
            printf("Editora: %s\n", livros[i].editora);
            printf("Ano: %d\n", livros[i].ano_pub);
            printf("Status: %s\n", livros[i].status);
            return;
        }
    }

    printf("Livro não encontrado!\n");
}
```

```
// ----------------------------
// Pesquisa de Usuário
// ----------------------------

void pesquisarUsuario(Usuario usuarios[], int numUsuarios) {
    int mat;

    printf("Matrícula do usuário: ");
    scanf("%d", &mat);
    getchar();

    for (int i = 0; i < numUsuarios; i++) {
        if (usuarios[i].matricula == mat) {

            printf("\n--- Usuário Encontrado ---\n");
            printf("Nome: %s\n", usuarios[i].nome);
            printf("Curso: %s\n", usuarios[i].curso);
            printf("Telefone: %s\n", usuarios[i].telefone);

            return;
        }
    }

    printf("Usuário não encontrado!\n");
}

// ----------------------------
// Lista Empréstimos Ativos
// ----------------------------

void listarEmprestimosAtivos(Emprestimo emprestimos[], int numEmprestimos) {
```

```
    printf("\n--- Empréstimos Ativos ---\n");

    int achou = 0;

    for (int i = 0; i < numEmprestimos; i++) {
        if (strcmp(emprestimos[i].status, "ativo") == 0) {
            printf("Código: %d | Usuário: %d | Livro: %d | Devolver até: %d/%d/%d\n",
                   emprestimos[i].codigo,
                   emprestimos[i].matricula_usuario,
                   emprestimos[i].codigo_livro,
                   emprestimos[i].devolucao_prevista.dia,
                   emprestimos[i].devolucao_prevista.mes,
                   emprestimos[i].devolucao_prevista.ano);
            achou = 1;
        }
    }

    if (!achou)
        printf("Nenhum empréstimo ativo.\n");
}

// ----------------------------
// Relatório: Livros Mais Emprestados
// ----------------------------

void relatorioLivrosMaisEmprestados(Emprestimo emprestimos[], int
numEmprestimos) {

    int contagem[MAX_LIVROS] = {0};

    for (int i = 0; i < numEmprestimos; i++) {
```

```
        contagem[emprestimos[i].codigo_livro]++;
    }

    printf("\n--- Livros Mais Emprestados ---\n");

    for (int i = 0; i < MAX_LIVROS; i++) {
        if (contagem[i] > 0)
            printf("Livro %d: %d empréstimos\n", i, contagem[i]);
    }
}

// ----------------------------
// Relatório: Usuários em Atraso
// ----------------------------

void relatorioUsuariosAtraso(Emprestimo emprestimos[], Usuario usuarios[],
                             int numEmprestimos, int numUsuarios, Data dataAtual) {

    printf("\n--- Usuários em Atraso ---\n");
    int achou = 0;

    for (int i = 0; i < numEmprestimos; i++) {

        if (strcmp(emprestimos[i].status, "ativo") == 0 &&
            compararDatas(dataAtual, emprestimos[i].devolucao_prevista) > 0) {

            int mat = emprestimos[i].matricula_usuario;

            for (int j = 0; j < numUsuarios; j++) {
                if (usuarios[j].matricula == mat) {
                    printf("Usuário: %s | Matrícula: %d | Livro: %d | Venceu em
%d/%d/%d\n",
```

```
                    usuarios[j].nome,
                    mat,
                    emprestimos[i].codigo_livro,
                    emprestimos[i].devolucao_prevista.dia,
                    emprestimos[i].devolucao_prevista.mes,
                    emprestimos[i].devolucao_prevista.ano);

                achou = 1;
                break;
            }
        }
    }

    if (!achou)
        printf("Nenhum usuário em atraso.\n");
}

// ----------------------------
// Salvar arquivos
// ----------------------------

void salvarLivros(Livro livros[], int numLivros) {
    FILE *f = fopen("livros.txt", "w");
    if (!f) return;

    fwrite(livros, sizeof(Livro), numLivros, f);
    fclose(f);
}

void salvarUsuarios(Usuario usuarios[], int numUsuarios) {
    FILE *f = fopen("usuarios.txt", "w");
```

```
    if (!f) return;

    fwrite(usuarios, sizeof(Usuario), numUsuarios, f);
    fclose(f);
}

void salvarEmprestimos(Emprestimo emprestimos[], int numEmprestimos) {
    FILE *f = fopen("emprestimos.txt", "w");
    if (!f) return;

    fwrite(emprestimos, sizeof(Emprestimo), numEmprestimos, f);
    fclose(f);
}

// --------------------------
// Carregar arquivos
// --------------------------

void carregarLivros(Livro livros[], int *numLivros) {
    FILE *f = fopen("livros.txt", "r");
    if (!f) {
        *numLivros = 0;
        return;
    }

    *numLivros = fread(livros, sizeof(Livro), MAX_LIVROS, f);
    fclose(f);
}

void carregarUsuarios(Usuario usuarios[], int *numUsuarios) {
    FILE *f = fopen("usuarios.txt", "r");
    if (!f) {
```

```
        *numUsuarios = 0;
        return;
    }

    *numUsuarios = fread(usuarios, sizeof(Usuario), MAX_USUARIOS, f);
    fclose(f);
}

void carregarEmprestimos(Emprestimo emprestimos[], int *numEmprestimos) {
    FILE *f = fopen("emprestimos.txt", "r");
    if (!f) {
        *numEmprestimos = 0;
        return;
    }

    *numEmprestimos = fread(emprestimos, sizeof(Emprestimo),
MAX_EMPRESTIMOS, f);
    fclose(f);
}

// ----------------------------
// Funções auxiliares
// ----------------------------

// retorna data + 7 dias (bem simples)
Data calcularDataDevolucao(Data d) {
    d.dia += 7;

    if (d.dia > 30) {
        d.dia -= 30;
        d.mes++;
```

```
        if (d.mes > 12) {
            d.mes = 1;
            d.ano++;
        }
    }
    return d;
}

// compara datas: 1 = d1 maior, -1 = d2 maior, 0 = iguais
int compararDatas(Data d1, Data d2) {

    if (d1.ano > d2.ano) return 1;
    if (d1.ano < d2.ano) return -1;

    if (d1.mes > d2.mes) return 1;
    if (d1.mes < d2.mes) return -1;

    if (d1.dia > d2.dia) return 1;
    if (d1.dia < d2.dia) return -1;

    return 0;
}

void definirDataAtual(Data *d) {
    printf("Informe a data atual (dia mes ano): ");
    scanf("%d %d %d", &d->dia, &d->mes, &d->ano);
    getchar();
}

int main() {

    Livro livros[MAX_LIVROS];
```

```
Usuario usuarios[MAX_USUARIOS];
Emprestimo emprestimos[MAX_EMPRESTIMOS];

int numLivros = 0, numUsuarios = 0, numEmprestimos = 0;

Data dataAtual;

// Carrega arquivos
inicializarLivros(livros, &numLivros);
inicializarUsuarios(usuarios, &numUsuarios);
inicializarEmprestimos(emprestimos, &numEmprestimos);

int opcao;

do {
    printf("\n======================================\n");
    printf("        SISTEMA DE BIBLIOTECA\n");
    printf("======================================\n");
    printf("1 - Cadastrar Livro\n");
    printf("2 - Cadastrar Usuário\n");
    printf("3 - Realizar Empréstimo\n");
    printf("4 - Realizar Devolução\n");
    printf("5 - Renovar Empréstimo\n");
    printf("6 - Pesquisar Livro\n");
    printf("7 - Pesquisar Usuário\n");
    printf("8 - Listar Empréstimos Ativos\n");
    printf("9 - Relatório: Livros Mais Emprestados\n");
    printf("10 - Relatório: Usuários em Atraso\n");
    printf("0 - Sair\n");
    printf("Escolha uma opção: ");
    scanf("%d", &opcao);
    getchar(); // limpa buffer
```

```
switch (opcao) {

    case 1:
        cadastrarLivro(livros, &numLivros);
        break;

    case 2:
        cadastrarUsuario(usuarios, &numUsuarios);
        break;

    case 3:
        realizarEmprestimo(livros, usuarios, emprestimos,
                           &numLivros, &numUsuarios, &numEmprestimos);
        break;

    case 4:
        realizarDevolucao(livros, emprestimos,
                          &numLivros, &numEmprestimos);
        break;

    case 5:
        renovarEmprestimo(emprestimos, &numEmprestimos);
        break;

    case 6:
        pesquisarLivro(livros, numLivros);
        break;

    case 7:
        pesquisarUsuario(usuarios, numUsuarios);
        break;
```

```
            case 8:
                listarEmprestimosAtivos(emprestimos, numEmprestimos);
                break;

            case 9:
                relatorioLivrosMaisEmprestados(emprestimos, numEmprestimos);
                break;

            case 10:
                definirDataAtual(&dataAtual);
                relatorioUsuariosAtraso(emprestimos, usuarios,
                                        numEmprestimos, numUsuarios, dataAtual);
                break;

            case 0:
                printf("Saindo...\n");
                break;

            default:
                printf("Opção inválida!\n");
        }

    } while (opcao != 0);

    return 0;
}
```