MAPA – Material de Avaliação Prática da Aprendizagem

Acadêmico: GULTEMEBRGUE CARLOS REGO	**R.A.** 25531299-5
Curso: ANÁLISE E DESENVOLVIMENTO DE SISTEMAS	
Disciplina: LINGUAGEM E TÉCNICAS DE PROGRAMAÇÃO	
Valor da atividade: 5.0	**Prazo:** 07/12/25 23:59

Instruções para Realização da Atividade

1. Todos os campos acima deverão ser devidamente preenchidos;
2. É obrigatória a utilização deste formulário para a realização do MAPA.
3. Esta é uma atividade individual. Caso identificado cópia de colegas, o trabalho de ambos sofrerá decréscimo de nota.
4. Utilizando este formulário, realize sua atividade, salve em seu computador, renomeie e envie em forma de anexo.
5. Formatação exigida para esta atividade: documento Word, Fonte Arial ou Times New Roman tamanho 12, Espaçamento entre linhas 1,5, texto justificado.
6. Ao utilizar quaisquer materiais de pesquisa faça a referência conforme as normas da ABNT;
7. Critérios de avaliação: Utilização deste template (Formulário Padrão); Atendimento ao Tema; Constituição dos argumentos e organização das Ideias; Correção Gramatical e atendimento às normas ABNT.
8. Procure argumentar de forma clara e objetiva, de acordo com o conteúdo da disciplina.

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

Bons estudos!

AGORA É COM VOCÊ!

Imagine que você foi contratado para desenvolver um sistema de gerenciamento de livros para uma pequena biblioteca. O sistema deve permitir que o bibliotecário cadastre, pesquise, ordene e gerencie o acervo de 20 livros. Esta tarefa exigirá a aplicação de diversos conceitos fundamentais da linguagem C.

Dado o contexto, seguem as etapas que precisarão cumprir na atividade:

Etapa 1: Planejamento e Estrutura do Código - Antes de começar a codificar, planeje a estrutura do seu programa.

Defina a struct: Crie uma estrutura chamada Livro com os campos necessários:

```
int codigo;
char titulo[50];
char autor[30];
char area[30];
int ano;
char editora[30];
```

Declare as constantes e variáveis: Use `#define` para definir uma constante para o tamanho do acervo (`TAMANHO_ACERVO 20`). Na função `main`, declare um vetor dessa struct (`struct Livro acervo[TAMANHO_ACERVO]`) e outras variáveis auxiliares.

Desenhe o menu: Crie um menu principal para o usuário escolher entre as seguintes opções:

1 - Cadastrar livros
2 - Imprimir todos os livros
3 - Pesquisar livro por código
4 - Ordenar livros por ano de publicação
5 - Sair do programa

Esboce as funções: Crie as assinaturas das funções que você precisará. Por exemplo:

```
void cadastrarLivros(struct Livro acervo[], int tamanho);
void imprimirLivros(struct Livro acervo[], int tamanho);
void pesquisarLivro(struct Livro acervo[], int tamanho, int codigoBusca);
void ordenarLivros(struct Livro acervo[], int tamanho);
```

Etapa 2: Implementação do Código - Agora, implemente cada funcionalidade em seu código.

Função main: use um laço de repetição (`do-while` ou `while`) e uma estrutura de seleção (`switch-case` ou `if-else`) para exibir o menu e chamar a função correspondente à opção do usuário.

Função `cadastrarLivros`: use um laço `for` para percorrer o vetor de structs. Dentro do laço, use `printf` e `scanf` para solicitar e ler os dados de cada livro. Lembre-se de usar `fflush(stdin)` após cada `scanf` para limpar o buffer do teclado.

Função `imprimirLivros`: percorra o vetor com um laço `for` e use `printf` para exibir os dados de cada livro.

Função `pesquisarLivro`: solicite ao usuário o código do livro a ser pesquisado. Percorra o vetor com um laço `while`. Se o código for encontrado, exiba as informações do livro. Se não, mostre uma mensagem de "não encontrado".

Função `ordenarLivros`: implemente o método de ordenação da bolha (`BubbleSort`) para ordenar os livros por ano de publicação, usando laços for aninhados e uma variável temporária para a troca.

Documente o código: Adicione comentários (`//` ou `/* */`) em todas as partes importantes: o que cada função faz, o que cada variável armazena e por que certas decisões de lógica foram tomadas.

Etapa 3: Validação e Gravação

Compile e teste: compile seu programa e teste todas as funcionalidades, certificando-se de que não há erros de sintaxe e que o comportamento é o esperado.

Gravação do vídeo: grave um vídeo de até 5 minutos seguindo este roteiro:

- **Introdução:** apresente-se e explique o objetivo do projeto.
- **Visão Geral do Código:** mostre a estrutura principal do seu programa, destacando a `struct`, a constante e as funções que você criou.
- **Demonstração das Funcionalidades:** compile e execute o programa. Demonstre cada opção do menu (cadastrar, imprimir, pesquisar, ordenar) e explique o que acontece por trás da tela em cada etapa.
- **Discussão das Boas Práticas:** escolha um trecho de código (por exemplo, a função de ordenação ou de pesquisa) e comente a importância da documentação e da modularização. Explique como o uso de funções torna o código mais claro, fácil de manter e reutilizável.
- **Conclusão:** finalize o vídeo com suas considerações sobre o aprendizado na atividade e os desafios superados.

Postagem: poste o vídeo no YouTube no formato NÃO-LISTADO, ou seja, liberado apenas para quem tiver acesso ao link.

	Como devo entregar esta atividade
	Você deve colar neste *template* os códigos desenvolvidos, atendendo aos requisitos das Etapas 1 e 2 mencionados no enunciado **e o link do vídeo**, atendendo os requisitos da etapa 3. Após responder, você deverá salvar esse *template* (de preferência em PDF) e anexar no campo disponível na atividade. Clicando em "Responder", confira se o arquivo que está anexando é exatamente o arquivo correto, depois clique em "Finalizar" para enviar. Por fim, verifique se o arquivo aparece anexado na atividade.

	ATENÇÃO
	- Um vídeo explicativo da atividade MAPA está disponível para ajudá-lo nesse processo de criação e desenvolvimento. Você poderá acessar no Fórum ou em "ARQUIVOS > Material da Disciplina". - Plágios e cópias indevidas serão penalizados com descontos na nota, podendo chegar a zero. - Não são permitidas correções parciais no decorrer do módulo, pois a interpretação da atividade também faz parte da avaliação. - Atenção ao prazo de entrega da atividade. Sugerimos que envie sua atividade com pelo menos uma semana de antecedência para evitar transtornos e lentidão nos servidores. Evite o envio de atividade em cima do prazo.

Boa atividade!

RESPOSTA:

Link do Vídeo: https://youtu.be/pzur8OOxaFE

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

#define TAMANHO_ACERVO 20 // Constante que define o tamanho máximo do acervo

// Estrutura que representa um livro
struct Livro {
    int codigo;
```

```
    char titulo[50];
    char autor[30];
    char area[30];
    int ano;
    char editora[30];
};

// Assinaturas ou Funções principais do sistema
void cadastrarLivros(struct Livro acervo[], int tamanho, int *numLivros);
void imprimirLivros(struct Livro acervo[], int tamanho);
void pesquisarLivro(struct Livro acervo[], int tamanho, int codigoBusca);
void ordenarLivros(struct Livro acervo[], int tamanho);

int main() {
    setlocale(LC_ALL, "Portuguese");
    struct Livro acervo[TAMANHO_ACERVO];
    int opcao, codigoBusca;
    int numLivros = 0; // Contador de livros cadastrados

    do {
        // Menu principal
        printf("\n===== MENU BIBLIOTECA =====\n");
        printf("1 - Cadastrar livros\n");
        printf("2 - Imprimir todos os livros\n");
        printf("3 - Pesquisar livro por codigo\n");
        printf("4 - Ordenar livros por ano de publicacao\n");
        printf("5 - Sair\n");
        printf("Escolha uma opcao: ");
        scanf("%d", &opcao);

        switch(opcao) {
            case 1:
```

```
            cadastrarLivros(acervo, TAMANHO_ACERVO, &numLivros);
            break;
        case 2:
            imprimirLivros(acervo, numLivros);
            break;
        case 3:
            printf("Digite o codigo do livro: ");
            scanf("%d", &codigoBusca);
            pesquisarLivro(acervo, numLivros, codigoBusca);
            break;
        case 4:
            ordenarLivros(acervo, numLivros);
            printf("Livros ordenados por ano!\n");
            break;
        case 5:
            printf("Encerrando o programa...\n");
            break;
        default:
            printf("Opcao invalida!\n");
    }
} while(opcao != 5);

return 0;
}

// Função para cadastrar livros
void cadastrarLivros(struct Livro acervo[], int tamanho, int *numLivros) {
    int i = *numLivros;
    char continuar;
    while (i < tamanho) {
        printf("\nCadastro do Livro %d:\n", i + 1);
        printf("codigo: ");
```

```
        scanf("%d", &acervo[i].codigo);
        while (getchar() != '\n'); // Consome todos os caracteres restantes no buffer até encontrar a quebra de linha.
        printf("Titulo: ");
        fgets(acervo[i].titulo, 50, stdin);
        acervo[i].titulo[strcspn(acervo[i].titulo, "\n")] = 0;
        printf("Autor: ");
        fgets(acervo[i].autor, 30, stdin);
        acervo[i].autor[strcspn(acervo[i].autor, "\n")] = 0;
        printf("Area: ");
        fgets(acervo[i].area, 30, stdin);
        acervo[i].area[strcspn(acervo[i].area, "\n")] = 0;
        printf("Ano: ");
        scanf("%d", &acervo[i].ano);
        while (getchar() != '\n'); // Consome todos os caracteres restantes no buffer até encontrar a quebra de linha.
        printf("Editora: ");
        fgets(acervo[i].editora, 30, stdin);
        acervo[i].editora[strcspn(acervo[i].editora, "\n")] = 0;
        (*numLivros)++;
        i++;
        if (i < tamanho) {
            printf("\nDeseja cadastrar outro livro? (s/n): ");
            scanf("%c", &continuar);
            getchar(); // Consome todos os caracteres restantes no buffer até encontrar a quebra de linha.
            if (continuar != 's' && continuar != 'S') {
                break;
            }
        }
    }
}
```

```
// Função para imprimir todos os livros
void imprimirLivros(struct Livro acervo[], int tamanho) {
    if (tamanho == 0) {
        printf("\nNenhum livro cadastrado.\n");
        return;
    }

    printf("\n===== LISTA DE LIVROS =====\n");
    for (int i = 0; i < tamanho; i++) {
        printf("\nLivro %d:\n", i + 1);
        printf("codigo: %d\n", acervo[i].codigo);
        printf("Titulo: %s\n", acervo[i].titulo);
        printf("Autor: %s\n", acervo[i].autor);
        printf("Area: %s\n", acervo[i].area);
        printf("Ano: %d\n", acervo[i].ano);
        printf("Editora: %s\n", acervo[i].editora);
    }
}

// Função para pesquisar livro por codigo
void pesquisarLivro(struct Livro acervo[], int tamanho, int codigoBusca) {
    if (tamanho == 0) {
        printf("\nNenhum livro cadastrado para pesquisar.\n");
        return;
    }

    int encontrado = 0;
    for (int i = 0; i < tamanho; i++) {
        if (acervo[i].codigo == codigoBusca) {
            printf("\nLivro encontrado:\n");
            printf("Titulo: %s\n", acervo[i].titulo);
```

```
            printf("Autor: %s\n", acervo[i].autor);
            printf("Area: %s\n", acervo[i].area);
            printf("Ano: %d\n", acervo[i].ano);
            printf("Editora: %s\n", acervo[i].editora);
            encontrado = 1;
            break;
        }
    }
    if (!encontrado) {
        printf("Livro com codigo %d nao encontrado.\n", codigoBusca);
    }
}

// Função para ordenar livros por ano de publicação (Bubble Sort)
void ordenarLivros(struct Livro acervo[], int tamanho) {
    if (tamanho < 2) {
        // nao há o que ordenar
        return;
    }

    struct Livro temp;
    for (int i = 0; i < tamanho - 1; i++) {
        for (int j = 0; j < tamanho - i - 1; j++) {
            if (acervo[j].ano > acervo[j + 1].ano) {
                temp = acervo[j];
                acervo[j] = acervo[j + 1];
                acervo[j + 1] = temp;
            }
        }
    }
}
```