Exercícios de Structs e Funções em C:

Resolva os exercícios usando structs, tipos de dados (typedef) e criando funções unitárias para resolver cada operação que se pede nos exercícios.

1. Ponto no Plano Cartesiano:

- Crie uma struct chamada `Ponto` com dois membros: `x` e `y` (ambos do tipo `float`).
- Escreva um programa que declare uma variável do tipo `Ponto`, leia as coordenadas x e y do usuário e as armazene na struct.
- Imprima as coordenadas do ponto no formato (x, y).

2. Retângulo:

- Crie uma struct chamada `Retangulo` com dois membros do tipo `Ponto`: `superiorEsquerdo` e `inferiorDireito`.
- Escreva um programa que declare uma variável do tipo `Retangulo`, leia as coordenadas dos pontos superior esquerdo e inferior direito do usuário e as armazene na struct.
- Calcule e imprima a área e o perímetro do retângulo.

3. Cor RGB:

- Crie uma struct chamada `Cor` com três membros do tipo `int`: `vermelho`, `verde` e `azul` (representando os valores RGB de uma cor, de 0 a 255).
- Escreva um programa que declare uma variável do tipo `Cor`, leia os valores RGB do usuário e os armazene na struct.
- Imprima os valores RGB no formato (R, G, B).

4. Data:

- Crie uma struct chamada `Data` com três membros do tipo `int`: `dia`, `mes` e `ano`.
- Escreva um programa que declare uma variável do tipo `Data`, leia uma data do usuário no formato dd/mm/aaaa e a armazene na struct.
- Imprima a data no formato dd/mm/aaaa.
- **Desafio:** Valide a data inserida pelo usuário (dia entre 1 e 31, mês entre 1 e 12, ano maior que 0).

5. Horário:

- Crie uma struct chamada `Horario` com três membros do tipo `int`: `hora`, `minuto` e `segundo`.
- Escreva um programa que declare uma variável do tipo `Horario`, leia um horário do usuário no formato hh:mm:ss e o armazene na struct.
- Imprima o horário no formato hh:mm:ss.
- **Desafio:** Valide o horário inserido pelo usuário (hora entre 0 e 23, minuto e segundo entre 0 e 59).

6. Aluno:

- Crie uma struct chamada `Aluno` com os seguintes membros:
  - `nome` (string de até 50 caracteres)
  - `matricula` (inteiro)
  - `notas` (array de 3 floats)
- Escreva um programa que declare uma variável do tipo `Aluno`, leia os dados do aluno (nome, matrícula e notas) do usuário e os armazene na struct.
- Calcule e imprima a média das notas do aluno.

7. Funcionário:

- Crie uma struct chamada `Funcionario` com os seguintes membros:
  - `nome` (string de até 50 caracteres)
  - `idade` (inteiro)
  - `salario` (float)
  - `cargo` (string de até 30 caracteres)
- Escreva um programa que declare um array de 5 variáveis do tipo `Funcionario`, leia os dados dos funcionários do usuário e os armazene no array.
- Imprima os dados de todos os funcionários.

8. Livro:

- Crie uma struct chamada `Livro` com os seguintes membros:
  - `titulo` (string de até 100 caracteres)
  - `autor` (string de até 50 caracteres)
  - `ano` (inteiro)
  - `genero` (string de até 30 caracteres)
- Escreva um programa que declare um array de 10 variáveis do tipo `Livro`, leia os dados dos livros do usuário e os armazene no array.
- Imprima a lista de livros ordenada por título.

9. Time de Futebol:

- Crie uma struct chamada `Time` com os seguintes membros:
  - `nome` (string de até 50 caracteres)
  - `pontos` (inteiro)
  - `vitorias` (inteiro)
  - `empates` (inteiro)
  - `derrotas` (inteiro)
- Escreva um programa que declare um array de 20 variáveis do tipo `Time`, leia os dados dos times do usuário e os armazene no array.
- Imprima a tabela do campeonato ordenada por pontos.