
	<p style="text-align: center;">MINISTÉRIO DA EDUCAÇÃO UNIVERSIDADE FEDERAL DO PIAUÍ – UFPI CAMPUS SENADOR HELVÍDIO NUNES DE BARROS – PICOS</p>		
Curso: Sistemas de Informação		Período: 2º	Ano/Semestre: 2025.1
Disciplina: Algoritmos e Programação II			Professor: José Denes Lima Araújo

5º ATIVIDADE – STRUCT

1. Crie uma **struct chamada Musica** com os seguintes campos:

- nome (nome da música)
- cantor (nome do cantor ou banda)

Em seguida, crie outra **struct chamada Playlist** com os seguintes campos:

- nome (nome da playlist)
- musicas (vetor de músicas, onde cada elemento será do tipo Musica)
- quantidade_musicas (quantidade de músicas na playlist)

Defina uma constante **#define MAX** que será o número máximo de músicas que uma playlist pode ter.

Crie um programa que:

- Pergunte ao usuário quantas playlists ele quer cadastrar.
- Cadastre as playlists perguntando o nome de cada uma.
- Em seguida, crie um menu interativo que ofereça as seguintes opções:
 - **Adicionar Música:** Permitir que o usuário adicione uma música à playlist escolhida, perguntando o nome e o cantor da música (desde que o número de músicas não ultrapasse o valor de MAX).
 - **Buscar Playlist:** Permitir que o usuário busque uma playlist pelo nome e exiba as músicas dessa playlist.
 - **Sair:** Encerra o programa.
- Ao final, o programa deve listar todas as playlists cadastradas, mostrando o nome da playlist e as músicas que pertencem a ela, de forma organizada.

OBS:

- O número máximo de músicas por playlist será definido pela constante **#define MAX**.
- O programa deve garantir que o número de músicas não ultrapasse o valor de MAX para cada playlist.

Exemplo de entrada e saída:

Quantas playlists você quer cadastrar? 2

Digite o nome da playlist 1: Rock Hits

Digite o nome da playlist 2: Pop Favorites

Menu:

1. Adicionar Música

2. Buscar Playlist

3. Sair

Escolha uma opção: 1

Digite o nome da música: Bohemian Rhapsody

Digite o nome do cantor: Queen

Escolha a playlist para adicionar a música: Rock Hits

Menu:

1. Adicionar Música

2. Buscar Playlist

3. Sair

Escolha uma opção: 2

Digite o nome da playlist para buscar: Rock Hits

Playlist: Rock Hits

Músicas:

1. Bohemian Rhapsody - Queen

2. Don't Stop Me Now - Queen

2. Defina uma struct chamada Fracao com os seguintes campos:

- numerador (número inteiro)
- denominador (número inteiro)

Crie um programa que tenha as seguintes funcionalidades:

- **Cadastrar duas frações:** O programa deve pedir ao usuário para inserir o numerador e o denominador de duas frações.
- **Somar as frações:** O programa deve calcular e exibir a soma das duas frações inseridas.

- **Simplificar a fração resultante:** O programa deve simplificar a fração resultante da soma. (Para simplificar, deve-se dividir o numerador e o denominador pelo **máximo divisor comum (MDC)**).

O programa deve exibir:

- As duas frações inseridas.
- A fração resultante da soma e a fração simplificada.

3. Crie um programa para registrar informações de um **produto** em um estoque. O produto pode ser registrado com dois tipos de informações diferentes: **ou o preço do produto ou a quantidade em estoque**.

Implemente um programa que:

1. **Defina uma union chamada** Produto com os seguintes campos:
 - preco - o preço do produto.
 - quantidade - a quantidade em estoque do produto.
2. **Crie uma struct chamada** RegistroProduto com os seguintes campos:
 - tipoProduto - onde 1 representa "preço" e 2 representa "quantidade".
 - produto - a union que irá armazenar o preço ou a quantidade.
3. **Crie um programa que:**
 - Pergunte ao usuário qual informação ele deseja registrar (preço ou quantidade).
 - Armazene a informação escolhida na union.
 - Exiba a informação registrada de forma organizada.

Exemplo de entrada e saída:

Escolha uma opção:

1. Registrar preço
2. Registrar quantidade

Escolha uma opção: 1

Digite o preço do produto: 25.50

Produto registrado:

Preço: 25.50

4. Crie um jogo de **adivinhação de números**. O computador vai gerar um número aleatório entre 1 e 100, e o jogador terá que tentar adivinhar qual é esse número. Para isso, você vai utilizar uma struct para armazenar as informações do jogador, como o nome e o número de tentativas feitas.

Implemente um programa que:

1. **Defina uma struct chamada Jogador** com os seguintes campos:
 - nome - o nome do jogador.
 - Tentativas - o número de tentativas feitas pelo jogador.
2. **Crie um programa que:**
 - Pergunte o nome do jogador.
 - Gere um número aleatório entre 1 e 100.
 - Permita que o jogador tente adivinhar o número. Para cada tentativa:
 - Informe se o palpite do jogador é maior ou menor que o número sorteado.
 - O programa deve contar as tentativas do jogador e exibir o número de tentativas no final, juntamente com o nome do jogador.
 - O jogo termina quando o jogador acertar o número.

Exemplo de entrada e saída:

Digite o nome do jogador: João

Tente adivinhar o número (entre 1 e 100): 50

Seu palpite foi maior que o número sorteado! Tente novamente.

Tente adivinhar o número (entre 1 e 100): 30

Seu palpite foi menor que o número sorteado! Tente novamente.

Tente adivinhar o número (entre 1 e 100): 40

Parabéns, João! Você acertou o número em 3 tentativas.

5. **Escreva um programa que:**

- Crie uma struct chamada Carro com os seguintes campos:
 - o Marca (uma string de no máximo 15 caracteres);
 - o Ano de fabricação (um número inteiro);
 - o Preço (um número real).

- Crie um **vetor** de 5 elementos do tipo **Carro** para armazenar os dados dos carros (marca, ano, preço).
- Após a leitura dos dados, o programa deve ler um valor p (preço de referência).
- Em seguida, exiba as informações de todos os carros cujo preço seja **inferior** a p.

6. Desenvolva um programa em linguagem C que gerencie o estoque de um mercado, utilizando structs para organizar as informações dos produtos.

Especificações:

- **Crie uma struct chamada Produto, contendo:**
 - o Código único (inteiro),
 - o Nome (string com no máximo 15 caracteres),
 - o Preço (valor real),
 - o Quantidade (inteiro).
- Crie e leia um vetor de 5 produtos, preenchendo as informações da struct com dados fornecidos pelo usuário.
 - o O programa não deve aceitar o cadastramento de produtos com códigos iguais.
- Em seguida, implemente a funcionalidade de processar pedidos:
 - o Cada pedido será composto pelo código do produto e a quantidade desejada.
 - o Busque o produto no vetor utilizando o código informado.
 - o Se o produto for encontrado e houver quantidade suficiente em estoque, atualize o estoque, subtraindo a quantidade pedida.
 - o Informe ao usuário que o pedido foi atendido.
 - o Caso não haja quantidade suficiente, informe que o pedido não pode ser atendido.
- O programa deve continuar lendo pedidos até que o usuário digite o código igual a zero, indicando o encerramento do processo.

7. Crie uma estrutura representando os alunos de um determinado curso. A estrutura deve conter a matrícula do aluno, nome e três notas do aluno.

Especificações:

- Permita ao usuário entrar com dados de 5 alunos.
- Encontre o aluno com maior nota da primeira prova.
- Encontre o aluno com maior média geral.
- Encontre o aluno com menor média geral.
- Para cada aluno diga se ele foi aprovado ou reprovado, considerando médias maiores ou iguais a 7 como aprovado.

8. Embaralhamento de cartas:

Faça um programa que simule o embaralhamento de cartas de um baralho francês com um deck de 52 cartas. Esse tipo de baralho possui 4 naipes (paus, ouros, copas e espadas) e treze cartas iniciando do Às (valor 1), os valores 2 até 10, o valete, a dama e o rei. O programa deve usar uma estrutura **CARTA (struct)** com os componentes **naipe e valor**. Esses componentes devem ser representados pelos seus respectivos nomes. Por exemplo, valores “Às”, “Seis”, “Rei” e naipes “Copas”, “Paus”. O programa deve inicialmente iniciar um deck de 52 cartas do baralho e solicitar ao usuário um número referente ao tipo de exibição.

Caso seja digitado 1 será impresso em ordem do Às até o Rei, na ordem de naipes Paus, Ouros, Copas e Espadas. O nome das cartas deve ser completo no formato “<valor> de <naipe>”, uma carta por linha.

Por exemplo:

Às de Paus

Dois de Paus

...

Rei de Paus

Às de Ouros

...

Caso seja digitado 0 o programa deve embaralhar o deck, e imprimir as 52 cartas embaralhadas. **Caso seja digitado outro valor fora dos especificados**, deve imprimir “ERROR”.

OBS: Use as bibliotecas Stdlib.h e time.h para aleatorizar as cartas. O uso da biblioteca string.h suas funções é permitido.

ENTRADA:

Num_digitado = 1

SAÍDA:

Às de Paus
Dois de Paus
Três de Paus
Quatro de Paus
Cinco de Paus
Seis de Paus
Sete de Paus
Oito de Paus
Nove de Paus
Dez de Paus
Valete de Paus
Dama de Paus
Rei de Paus
Às de Ouros
Dois de Ouros
Três de Ouros
Quatro de Ouros
Cinco de Ouros
Seis de Ouros
Sete de Ouros
Oito de Ouros
Nove de Ouros
Dez de Ouros
Valete de Ouros
Dama de Ouros
Rei de Ouros
Às de Copas
Dois de Copas
Três de Copas
Quatro de Copas
Cinco de Copas
Seis de Copas
Sete de Copas
Oito de Copas
Nove de Copas
Dez de Copas
Valete de Copas

ENTRADA:

Num_digitado = 0

SAÍDA:

Dois de Ouros
Sete de Espadas
Quatro de Paus
Sete de Copas
Rei de Ouros
Seis de Copas
Dois de Paus
Oito de Ouros
Cinco de Paus
Dois de Copas
Cinco de Espadas
Seis de Espadas
Nove de Paus
Rei de Espadas
Oito de Paus
Rei de Paus
Nove de Copas
Sete de Ouros
Dama de Paus
Dez de Espadas
Dez de Paus
Três de Ouros
Cinco de Ouros
Às de Copas
Às de Espadas
Sete de Paus
Oito de Copas
Valete de Paus
Três de Espadas
Dama de Espadas
Rei de Copas
Valete de Ouros
Nove de Ouros
Dois de Espadas
Quatro de Espada
Três de Paus
Três de Copas
Dez de Ouros
Quatro de Copas
Dama de Copas
Valete de Copas
Dez de Copas

Às de Paus
Valete de Espadas
Nove de Espadas
Dama de Ouros
Cinco de Copas
Seis de Ouros
Quatro de Ouros
Às de Ouros
Seis de Paus
Oito de Espadas