Lista de exercicios (Ponteiros, Alocação Dinâmica de Memória, Struct)

1. Struct e Ponteiros - Cadastro de Pessoa

Crie um programa que utilize uma `struct` chamada `Pessoa` contendo:

- `nome` (string alocada dinamicamente)
- `idade` (int)
- `altura` (float)

O programa deve:

1. Alocar dinamicamente uma `Pessoa`.
2. Solicitar os dados do usuário.
3. Exibir os dados.
4. Liberar a memória alocada.

2. Array Dinâmico de Inteiros

Crie um programa que solicite ao usuário um número **N** e aloque dinamicamente um array de inteiros para armazenar `N` valores. O programa deve:

1. Permitir que o usuário insira os valores.
2. Exibir os valores na tela.
3. Liberar a memória antes de encerrar.

3. Cadastro de Produtos

Crie uma `struct` chamada `Produto`, contendo:

- `nome` (string alocada dinamicamente)
- `preco` (float)
- `quantidade` (int)

O programa deve:

1. Perguntar ao usuário quantos produtos deseja cadastrar.
2. Alocar dinamicamente um array de `Produto`.
3. Permitir o cadastro de `N` produtos.
4. Exibir os produtos cadastrados.
5. Liberar a memória antes de encerrar.

4. Manipulação de Structs com Ponteiros

Crie uma `struct` chamada `Carro` contendo:

- `marca` (string alocada dinamicamente)
- `ano` (int)
- `preco` (float)

O programa deve:

1. Criar dinamicamente uma variável do tipo `Carro`.
2. Solicitar ao usuário que insira os dados do carro.
3. Exibir as informações utilizando **ponteiros** e o operador `->`.
4. Liberar a memória antes de encerrar.

5. Vetor Dinâmico com realloc()

Crie um programa que permita ao usuário cadastrar alunos, mas sem um número fixo. O programa deve:

1. Começar com um vetor dinâmico de `Aluno` (struct com `nome` e `nota`).
2. A cada novo aluno, utilizar `realloc()` para aumentar o tamanho do vetor.
3. Exibir os dados dos alunos cadastrados.
4. Liberar a memória antes de encerrar.

6. Lista Encadeada Simples

Crie um programa que implemente uma **lista encadeada** para armazenar números inteiros.

1. Defina uma `struct` chamada `No` contendo um campo `valor` (int) e um ponteiro `prox` para o próximo nó.
2. Crie funções para:
    - Inserir um novo número no início da lista.
    - Exibir todos os números armazenados.
    - Liberar toda a memória ao final.

7. Aritmética de Ponteiros em Arrays Dinâmicos

Crie um programa que:

1. Solicite ao usuário um número **N** e aloque dinamicamente um vetor de `N` inteiros.
2. Insira valores no vetor.
3. Percorra o vetor **utilizando aritmética de ponteiros** (`ptr + i`).
4. Exiba os valores armazenados.
5. Libere a memória antes de encerrar.

8. Tabela de Funcionários

Crie uma `struct` chamada `Funcionario`, contendo:

- `nome` (string alocada dinamicamente)
- `cargo` (string alocada dinamicamente)
- `salario` (float)

O programa deve:

1. Criar dinamicamente um array de `Funcionario`.
2. Solicitar os dados do usuário.
3. Exibir os dados utilizando **aritmética de ponteiros**.
4. Liberar a memória antes de encerrar.

9. Struct com Ponteiro para Outra Struct

Crie uma `struct` chamada `Endereco` contendo:

- `rua` (string alocada dinamicamente)
- `numero` (int)

Agora, crie outra `struct` chamada `Pessoa`, contendo:

- `nome` (string alocada dinamicamente)
- `idade` (int)
- `endereco` (ponteiro para `Endereco`)

O programa deve:

1. Criar uma `Pessoa` dinamicamente.
2. Alocar memória para `Endereco`.
3. Solicitar os dados do usuário.
4. Exibir os dados armazenados.
5. Liberar toda a memória antes de encerrar.

10. Cadastro de Livros com Busca

Crie uma `struct` chamada `Livro`, contendo:

- `titulo` (string alocada dinamicamente)
- `autor` (string alocada dinamicamente)
- `ano` (int)

O programa deve:

1. Permitir ao usuário cadastrar **N** livros.
2. Exibir todos os livros cadastrados.
3. Permitir que o usuário busque um livro pelo título.
4. Liberar toda a memória antes de encerrar.