
Agora é hora
da Revisao!

Atividade 01 - Struct

- Escrever um programa que cadastre o nome e três notas de 10 alunos.
- No final da execução, mostre os nomes, as notas e a média de cada um deles.

Obs.: Usar uma estrutura de registros (struct) para armazenar os dados.

Atividade 02 - Struct

- Faça um programa que armazene em uma *struct* os dados de 10 funcionários de uma empresa:
  - Nome,
  - Idade, Sexo (M/F),
  - CPF,
  - Data de Nascimento,
  - Código do Setor onde trabalha (0-99),
  - Cargo que ocupa (string de ate 30 caracteres) e
  - Salário.
- Os dados devem ser digitados pelo usuário e armazenados na *struct*.
- No final, deverá ser exibido na tela.

Atividade 03 - Struct

- Crie um programa que leia os dados de 10 alunos:
  - Nome e
  - Media Final,
- Os dados devem ser armazenados em uma *struct*.
- Uma vez lidos os dados exibir na tela os dados dos alunos aprovados, seguido dos dados dos alunos reprovados. Considere a média superior ou igual 7 para aprovados.

Obs.: Fazer dois procedimentos: mostra_aprovados e mostra_reprovados (passar o ponteiro do vetor para as funções).

Atividade 04

Struct + Procedimentos +Ponteiros

A) Escrever um programa que cadastre 10 produtos de supermercado: **Código, Nome e Preço**.

B) Após o cadastro, deverá ser executado um procedimento que imprima uma lista com o código, nome e preço da cada produto. Na chamada do procedimento, deverá ser passado como parâmetro o endereço do vetor da *struct*.

C) Por último, deverá ser chamado um procedimento para consultar o preço de um produto através de seu código.

Atividade 05
Struct + Ponteiros + malloc (realloc)

- Defina um registro pessoa para armazenar nome e idade de pessoas.
- O programa deverá ler um número **indeterminado** de pessoas: enquanto a for respondido sim para a pergunta: “Cadastrar outro?”
- Para cada novo registro, deverá ser redimensionado o vetor pessoa.

Revisão 06

Crie um programa que permita armazenar dados sobre pessoas: nome, a altura e da data de nascimento. O programa deverá ler dados de um número indeterminado de pessoas. Cada pessoa deve ser representada por uma struct dentro de um vetor.

O programa deve, na tela de abertura, apresentar opções para:

[1] Inserir uma pessoa;

[2] listar todas as pessoas;

[3] Sair

Cada uma destas opções deve ser implementada em uma função separada.

Observação: A função **inserir_pessoa**, deverá receber como parametro a quantidade de pessoas cadastradas e o ponteiro para o vetor de pessoas. Antes de inserir um novo registro, a função, deverá redimensionar o vetor adicionando mais uma posição. O vetor deverá retornar a ultima posição (o total) de pessoas cadastradas.

Revisão 07

Faça um programa que leia a constituição do Brasil e armazene cada palavra em um estrutura, formada pela palavra a q quantidade de vezes que ela aparece.

O programa deverá ler uma palavra e percorrer a estrutura verificando se a palavra já existe. Caso encontre a palavra, deverá incrementar a quantidade de vezes que ela existe. Caso não encontre, deverá redimensionar o vetor de estruturas e insere a nova palavra, definindo como 1 a quantidade de vezes que ela aparece.

No final do programa, deverá listar todas as palavra e suas respectivas quantidades de repetições.

Revisão 07

Para resolver esta atividade, voce deverá utilizar a função **strtok().**

Revisão 08

Faça um programa para administrar uma estrutura de 50 números inteiros.

```
Struct numero{
  int valor;
  char primo[1];
  int divisores;
}
```

Os números serão gerados aleatoriamente (usar função rand()).

Após gerar os números, fazer uma função para verificar se os números são primos (sim ou não) e outra para calcular a quantidade de divisores.

Criar um procedimento para organizar o vetor de estruturas.

No final do programa, deverão ser listados todos os valores, uma vez crescente e outra decrescente.