

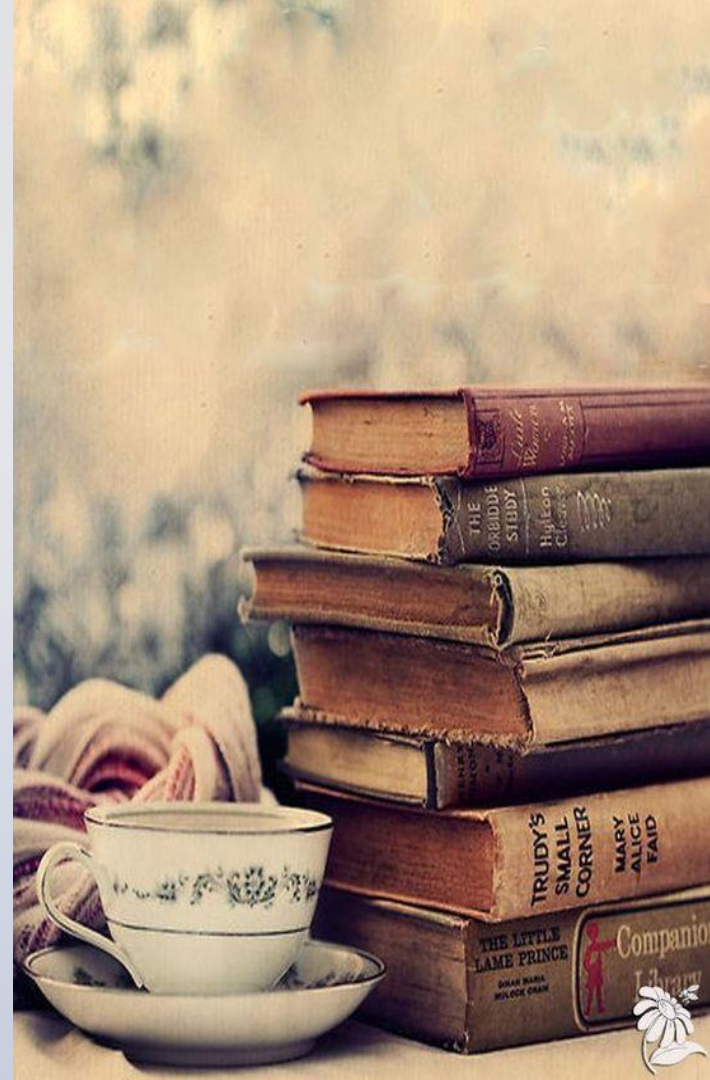
LABORATÓRIO DE PROGRAMAÇÃO II

Estruturas

Bibliografia:

Cap 6 - Estudo Dirigido de Linguagem C

José Augusto Manzano, Ed. Érica



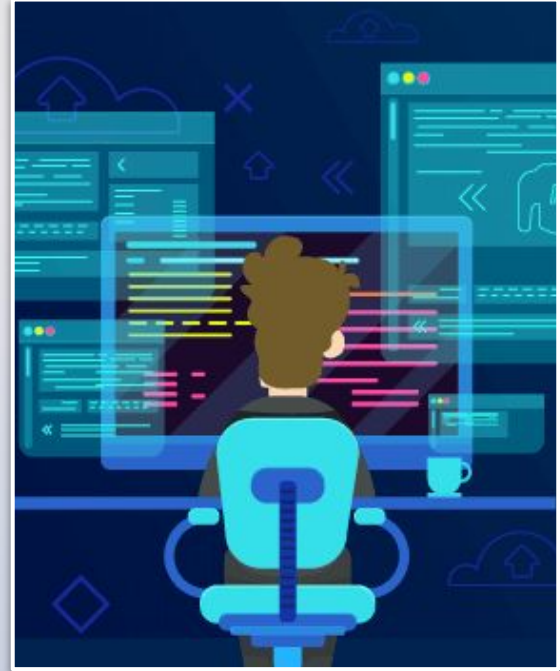
Agenda

Objetivos:

Apresentar conceitos, sintaxe e aplicações de Estruturas e desenvolver a habilidade de manipulação de registros

Agenda:

- Estruturas
- Sintaxe
- Escopo de Declaração
- Acesso aos Campos
- Estruturas Aninhadas
- Exemplo
- Passando Estruturas para Funções
- Atividade



Estruturas

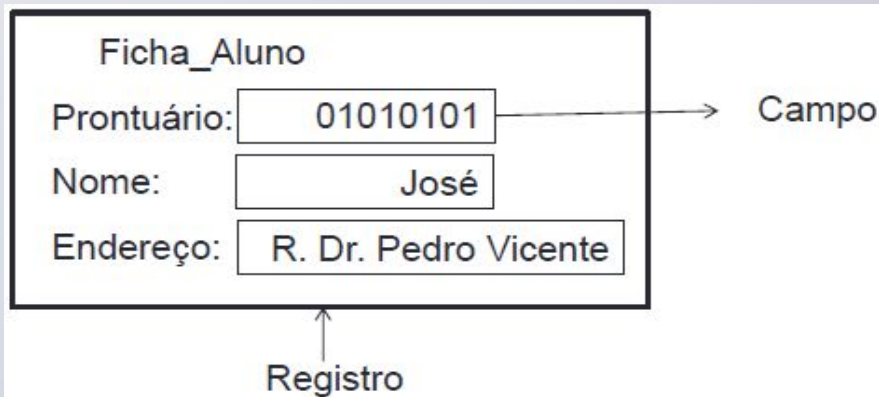
Uma estrutura é uma coleção de dados heterogêneos organizados em uma mesma estrutura de dados.

Os tipos primitivos (int, float, char, ...) comportam dados apenas do respectivo tipo.

Estruturas possibilitam definir um tipo de dado com campos de diferentes tipos.



Arquivo



Estruturas

Sintaxe

```
struct <nome>{  
    <declaração dos campos>  
};  
struct <nome> <var>;
```

Exemplo:

```
struct Ficha_Aluno{  
    int prontuario;  
    char nome[30];  
    char endereco[80];  
};  
struct Ficha_Aluno A;
```

Ficha_Aluno é o nome da estrutura criada; é um novo tipo de dado definido pelo programador. Assim, a variável A é do tipo struct Ficha_Aluno, e possui 3 campos

Essa estrutura geralmente é aplicada para um conjunto de registros.


Ex. struct Ficha_Aluno REG[5]



Acesso aos campos



```
struct Ficha_Aluno{  
    int prontuario;  
    char nome[30];  
    char endereco[80];  
} Ficha_Aluno A, REG[5];
```



Membros da estrutura são acessados com o operador ponto (.)
A.prontuario, A.endereco, REG[0].prontuario, REG[1].nome






Estruturas aninhadas

Membros das estruturas podem ser outras estruturas

```
struct endereco{  
    char rua[80];  
    int cep;  
    char cidade;  
    char estado;  
};  
  
struct ficha {  
    int prontuario;  
    char nome[40];  
    struct endereco casa;  
    struct endereco trabalho;  
};  
  
struct ficha func;
```

Então, a referência aos membros ocorre assim:

```
func.prontuario  
func.nome  
func.casa.rua  
func.casa.cep  
func.casa.cidade  
func.casa.estado  
func.trabalho.rua  
func.trabalho.cep  
func.trabalho.cidade  
func.trabalho.estado
```



Estruturas

Exemplo

Programa: `a_Struct.c`

Antes de abrir o programa, tente elaborar a seguinte solução:

- Criar uma estrutura para armazenar nome do aluno e um vetor de duas notas. A solução deve permitir armazenar dados de até 5 alunos

Faça um laço para armazenar os dados dos 5 alunos.

Em seguida, um laço deverá imprimir todos os registros.

Para esta tarefa, **não use modularidade.**



Estruturas

Exemplo

Programa: `b_Struct.c`

- Aprimore a solução `a_Struct.c`, (nomeando-a como
- `b_Struct.c`), tal que, após imprimir os dados dos alunos,
- imprima também a média de notas da turma, a maior e a menor notas da turma.



Passando estruturas para funções

Os membros de uma estrutura podem ser passados individualmente ou toda a estrutura pode ser passada.

Da mesma maneira, membros podem ser retornados pela função, assim como a estrutura completa.

A passagem de uma estrutura como argumento para uma função, assim como o seu retorno, pode ser realizado por valor ou por referência .

A passagem por valor não altera qualquer valor dos seus membros dentro da função, sendo necessário o seu retorno após tais modificações. Para alterar o valor dos membros sem a necessidade de retorno da estrutura, fazer passagem da estrutura por referência.





Atividade

Programa: **c_Struct.c**

Faça uma solução usando modularidade, para realizar as operações:

- [1] Incluir conta
- [2] Pesquisar conta
- [9] Sair

A pesquisa deve ser realizada por número da conta. Se encontrar, imprimir todos os dados da conta



```
struct Data {  
    int mes;  
    int dia;  
    int ano;  
};
```

```
struct Conta {  
    int num_conta;  
    char tipo_conta;  
    char nome[80];  
    float saldo;  
    struct Data ultpag;  
};
```



Atividade

Programa: `d_Struct.c`

Usando modularidade, faça uma solução para realizar as operações:

- [1] Incluir prontuário
- [2] Imprimir alunos em ordem de prontuário
- [3] Pesquisar prontuário
- [4] Alterar prontuário
- [5] Excluir prontuário
- [6] Sair

