

# Estrutura de Dados

Prof. Rafael Nunes

Sabemos que as matrizes  
armazenam dados  
do mesmo tipo...

O que podemos utilizar na Linguagem C  
quando queremos armazenar um  
conjunto de dados de tipos diferentes?

# Estruturas

+ Uniões, Enumerações e  
Definição de Tipos

O que é uma *Estrutura*?

Definida na Linguagem C...

# Estruturas

- Uma **estrutura em C** é similar aos registros em Pascal
- É um agrupamento de dados de **tipos diferentes**, referenciados por um **nome**
- Facilita bastante o **acesso** e a **organização**

Como definir uma *Estrutura*  
na Linguagem C?

# Como definir uma Estrutura?

- A declaração básica da estrutura em C:

```
struct nome_do_tipo_da_estrutura
```

```
{
```

```
    tipo_1 nome_1;
```

```
    tipo_2 nome_2;
```

```
    ...
```

```
    tipo_n nome_n;
```

Membros da Estrutura

```
} <variáveis_estrutura>;
```

# Como definir uma Estrutura?

- Geralmente, os dados agrupados em uma estrutura ***estão relacionados!***
- Como exemplo podemos citar os dados do ***endereço de uma pessoa***, onde temos:
  - o nome da rua (uma ***string***),
  - o número da casa (um ***inteiro***),
  - o complemento (uma ***string***),
  - o CEP (um ***inteiro longo***),
  - o nome do bairro (uma ***string***),
  - o nome da cidade (uma ***string***)
  - e a sigla do estado (uma ***string*** de 2 caracteres).



# Como definir uma Estrutura?

Definir a estrutura  
anterior, no quadro!

# Como definir uma Estrutura?

- Definindo a estrutura endereço:

```
struct endereco
{
    char  rua[30];
    int   numero;
    char  complem[30];
    long  int cep;
    char  bairro[15];
    char  cidade[15];
    char  estado[3];
};
```

Qual é a variável  
que vamos utilizar para  
preencher os dados?

# Como definir uma Estrutura?

- Quando declaramos uma estrutura como a anterior, ainda não estamos declarando nenhuma *variável*.
  - Estamos simplesmente especificando o *tipo*, ou a *estrutura* da nova struct...
- Para declarar uma variável da estrutura *struct endereco*:  

```
struct endereco dados;
```

# Como definir uma Estrutura?

- Podemos também declarar uma variável de uma estrutura colocando o nome da variável após a segunda chave da estrutura, antes do ponto-e-vírgula:

```
struct endereco
{
    char   rua[30];
    int    numero;
    char   complem[30];
    long   int cep;
    char   bairro[15];
    char   cidade[15];
    char   estado[3];
} dados;
```

# Como definir uma Estrutura?

- Se precisarmos de apenas uma variável da estrutura no nosso programa, não é necessário atribuir nome a estrutura:

```
struct /* sem nome... */  
{  
    char   rua[30];  
    int    numero;  
    char   complem[30];  
    long   int cep;  
    char   bairro[15];  
    char   cidade[15];  
    char   estado[3];  
} dados;
```

Como acessar ***os membros***  
de uma Estrutura?

# Membros da Estrutura

- Para acessar as variáveis internas à estrutura (elemento), usamos o **ponto** entre o nome da *variável* e o nome do *elemento* que queremos acessar.

*<variável> . <elemento\_da\_estrutura>*

- Ex:

```
struct endereco dados;  
strcpy (dados.rua, "Rua Cobre");
```




# Vamos praticar...

Sabendo que uma estrutura pode  
ser utilizada ***dentro de outra...***

<< 10 minutos >>

# structs\_prg1.c

- Sabendo que uma estrutura pode ser utilizada dentro de outra...
- ... **desenvolva** um programa para preencher uma **ficha de cadastro** de clientes de acordo com os dados abaixo
  - O **cliente** possui nome, endereço e telefone
  - O **endereço** é composto por rua, número, bairro, cidade, sigla\_estado e CEP
- Utilize a função **strcpy()** para preencher os dados de um cliente e mostre os resultados na tela
  - Rafael Nunes. Tel: 31235432
  - End: Rua Cobre, 200. Bairro **Atlético** 
  - Belo Horizonte/MG
  - CEP: 30310190

# Inicialização de Estruturas

Uma estrutura pode ser inicializada da mesma maneira como inicializamos variáveis?

Ex: `int x = 10;`

# Inicialização de Estruturas

- Podemos inicializar uma estrutura da mesma forma que inicializamos matrizes e arrays:

Ex:

```
struct paciente
```

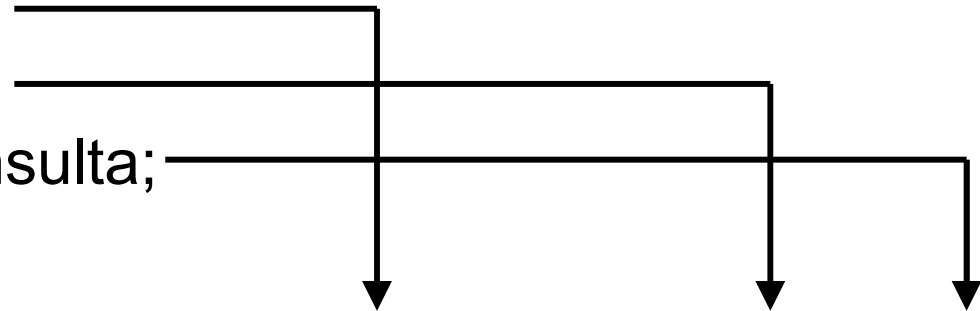
```
{
```

```
    char nome[20];
```

```
    int idade
```

```
    float preco_consulta;
```

```
};
```



```
struct paciente consulta = { "Fulano da Silva", 58, 175.00 };
```


- Atenção!** Depende de onde a *variável* é declarada.

# Vamos praticar...

## Inicializando Estruturas

<< 2 minutos >>

# structs\_prg2.c

- **Copie** o programa anterior (structs\_prg1.c) para um novo projeto
- Comente as linhas onde o cadastro do cliente foi preenchido utilizando a função **strcpy()**
- Inicialize a estrutura com os dados abaixo
  - Flavio Laper. Tel: 98761234
  - End: Rua Cobre, 200. Bairro **Atlético**. 
  - Belo Horizonte/MG
  - CEP: 30310190

# Atribuição de Estruturas

Quando podemos atribuir  
uma estrutura a outra?

# Atribuição de Estruturas

- Podemos atribuir uma estrutura a outra quando as duas estruturas **forem do mesmo tipo**.
- O compilador da Linguagem C irá copiar uma estrutura na outra.

Ex:

```
struct cad_cliente ficha;  
struct cad_cliente ficha2 = { ... };  
ficha = ficha2;
```



# Atribuição de Estruturas

## Mais um exemplo

```
struct produto {  
    char nome[30];  
    float preco;  
};  
int main(){  
    struct produto p1, p2;  
  
    strcpy(p1.nome, "Computador Core 2 Duo");  
    p1.preco= 1699.99;  
  
    p2= p1; //Atribuição entre estruturas  
  
    printf("%s = %.2f", p2.nome, p2.preco);  
    return 0;  
}
```

# Vamos praticar...

## Atribuindo Estruturas

<< 2 minutos >>

# structs\_prg3.c

- **Copie** o programa anterior (structs\_prg2.c) para um novo projeto
- Suponhamos que você inicializou o cliente “Flavio Laper” na variável ficha da estrutura *cad\_cliente* conforme abaixo:

```
struct cad_cliente ficha = {"Flavio Laper", ...};
```

- Renomeie a ficha do “Flavio Laper” para ficha2
- Crie uma nova variável do mesmo tipo (cad\_cliente) chamada ficha e atribua uma a outra conforme abaixo:

```
struct cad_cliente ficha2 = {"Flavio Laper", ...};  
struct cad_cliente ficha;  
ficha = ficha2;    //Atribuição
```

Não altere mais nada no programa. Isso é necessário porque o programa anterior tem a parte de impressão na tela que já está imprimindo a ficha e não a ficha2.

# Matrizes de Estruturas

Porque podemos criar  
matrizes de Estruturas?

# Matrizes de Estruturas

- Podemos criar matrizes de estruturas porque estruturas são um *tipo qualquer*, que foi determinado pelo usuário
- A declaração é a mesma:  

```
struct cad_cliente fichas[100]
```
- O acesso também:  

```
fichas[12].endereco.sigla_estado[1]
```

# Passando Estruturas para Funções

# Passando Estruturas para Funções

- Na linha de comando abaixo, **apenas um** elemento da estrutura **dados** (“rua”) é passada para a função strcpy()

```
strcpy (dados.rua, "Rua Cobre");
```

- Podemos passar para uma função a **estrutura inteira**:

```
void PreencheFicha (struct cad_cliente ficha)  
{  
    ...  
}
```

# Passando Estruturas para Funções

O que acontece quando passamos uma  
estrutura inteira para uma função???



# Passando Estruturas para Funções

- A passagem da estrutura para a função na linguagem C é feita por valor (cópia)
- As alterações na estrutura realizadas dentro da função não serão “enxergadas” pela variável do lado de fora da função
- Esse procedimento pode comprometer o desempenho do programa
- ... O que podemos fazer para contornar estes problema???

# Utilizamos Ponteiros!

Basta passar para a função um  
ponteiro para a estrutura

# Ponteiros para Estruturas

# Campos de Bits

Unões

# Enumerações

# Definição de Tipos

Até a próxima...



# Referências

- Curso de C do CPDEE/UFMG - 1996-1999. <http://www.cpdee.ufmg.br/cursos/C>
- ...