

**FA<sup>U</sup>EC**

**CENTRO PAULA SOUZA**  
COMPETÊNCIA EM EDUCAÇÃO PÚBLICA PROFISSIONAL

Faculdade de Tecnologia Prof. Jessen Vidal  
São José dos Campos

GOVERNO DO ESTADO DE  
**SÃO PAULO**

STRUCTS

# DEFINIÇÕES

- Um **registro** (= *record*) é uma coleção de várias variáveis, possivelmente de tipos diferentes.
- Na linguagem C, registros são conhecidos como **structs** .

# CRIAÇÃO (Definição de um tipo)

- Para se criar uma estrutura usa-se o comando **struct**. Sua forma geral é:

```
struct nome_da_estrutura {  
    tipo_1 nome_1;  
    tipo_2 nome_2;  
    ...  
    tipo_n nome_n;  
} variáveis_estrutura;
```

- O `nome_da_estrutura` é o nome para a estrutura.
- As `variáveis_estrutura` são opcionais e seriam nomes de variáveis que o usuário já estaria declarando e que seriam do tipo `nome_da_estrutura`.

# Exemplo – Criando uma Estrutura

```
struct endereco {  
    char rua [50];  
    int numero;  
    char bairro [20];  
    char cidade [30];  
    char estado [3];  
    long int CEP;  
};
```

# Exemplo - Declarando uma estrutura

```
struct endereco endereco_res;
```

Ou

```
struct endereco {  
    char rua [50];  
    int numero;  
    char bairro [20];  
    char cidade [30];  
    char estado [3];  
    long int CEP;  
} endereco_res, endereco_com;
```

# Exemplos:

## Criando um struct:

```
struct endereco {  
    char rua [50];  
    int numero;  
    char bairro [20];  
    char cidade [30];  
    char estado [3];  
    long int CEP;  
};
```

## Utilizando um struct criado na criação de outro struct:

```
struct ficha_pessoal{  
    char nome [50];  
    long int telefone;  
    struct endereco end;  
};
```

# Exemplos:

## Criando um struct sem nome:

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

É válido dar um nome para utilizá-lo mais vezes:

```
struct dma {  
    int dia;  
    int mes;  
    int ano;  
};  
  
struct dma x;  
struct dma y;
```

# Atribuindo Valores

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

```
x.dia = 31;  
x.mes = 8;  
x.ano = 1998;
```

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

```
struct dma x;  
  
x.dia = 31;  
x.mes = 8;  
x.ano = 1998;
```



# Atribuindo Valores

```
struct endereco {  
    char rua [50];  
    int numero;  
    char bairro [20];  
    char cidade [30];  
    char estado [3];  
    long int CEP;  
};
```

```
struct ficha_pessoal {  
    char nome [50];  
    long int telefone;  
    struct endereco end;  
};
```

---

```
main ( ) {  
    struct ficha_pessoal ficha;  
    strcpy (ficha.nome, "Luiz Silva");  
    ficha.telefone=4921234;  
    strcpy (ficha.end.rua, "Rua das Flores");  
    ficha.end.numero=10;  
    strcpy (ficha.end.bairro, "Jardins");  
    strcpy (ficha.end.cidade, "Belo Horizonte");  
    strcpy (ficha.end.estado, "MG");  
    ficha.end.CEP=31340230;  
}
```

# Atribuindo Valores

- Podemos atribuir uma variável do tipo struct a uma outra do mesmo tipo:

```
struct ficha_pessoal ficha, ficha2;
```

```
strcpy (ficha.nome, "Luiz Silva");
```

```
ficha.telefone=4921234;
```

```
strcpy (ficha.end.rua, "Rua das Flores");
```

```
ficha.end.numero=10;
```

```
strcpy (ficha.end.bairro, "Jardins");
```

```
strcpy (ficha.end.cidade, "Belo Horizonte");
```

```
strcpy (ficha.end.estado, "MG");
```

```
ficha.end.CEP=31340230;
```

```
ficha2 = ficha;
```

# Matrizes de Structs

- Um **struct** é como qualquer outro tipo de dado no C.
- Podemos, portanto, fazer matrizes de structs.
- Por exemplo, declaração de uma matriz de 100 fichas pessoais:

```
struct ficha_pessoal funcionarios [100];
```

- Acessando a segunda letra da sigla do estado da décima terceira ficha:

```
funcionarios[12].estado[1];
```

# Passando para funções

- Passando para uma função um struct inteiro:

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

# Utilizando com Ponteiros

- Podemos ter um ponteiro para uma estrutura. Por exemplo:

```
int main( )  
{  
    struct ponto{  
        int x;  
        int y;  
    } ponto1;
```

```
    struct ponto *pontoponteiro;  
    pontoponteiro=&ponto1;  
  
    printf("Digite o valor de x\n");  
    scanf("%d",&pontoponteiro->x);  
    printf("Digite o valor de y\n");  
    scanf("%d",&pontoponteiro->y);  
    printf(" X: %d\n",pontoponteiro->x);  
    printf(" Y: %d",pontoponteiro->y);  
}
```

# Utilizando com Ponteiros

```
int main( )
```

```
{
```

```
    struct ponto{
```

```
    int x;
```

```
    int y;
```

```
    } ponto1;
```

```
    struct ponto *pontoponteiro;
```

```
    pontoponteiro = &ponto1;
```

```
    printf("Digite o valor de x1\n");
```

```
    scanf("%d",&(*pontoponteiro).x);
```

```
    printf("Digite o valor de y\n");
```

```
    scanf("%d",&(*pontoponteiro).y);
```

```
    printf(" X: %d\n",(*pontoponteiro).x);
```

```
    printf(" Y: %d",(*pontoponteiro).y);
```

```
    getch();
```

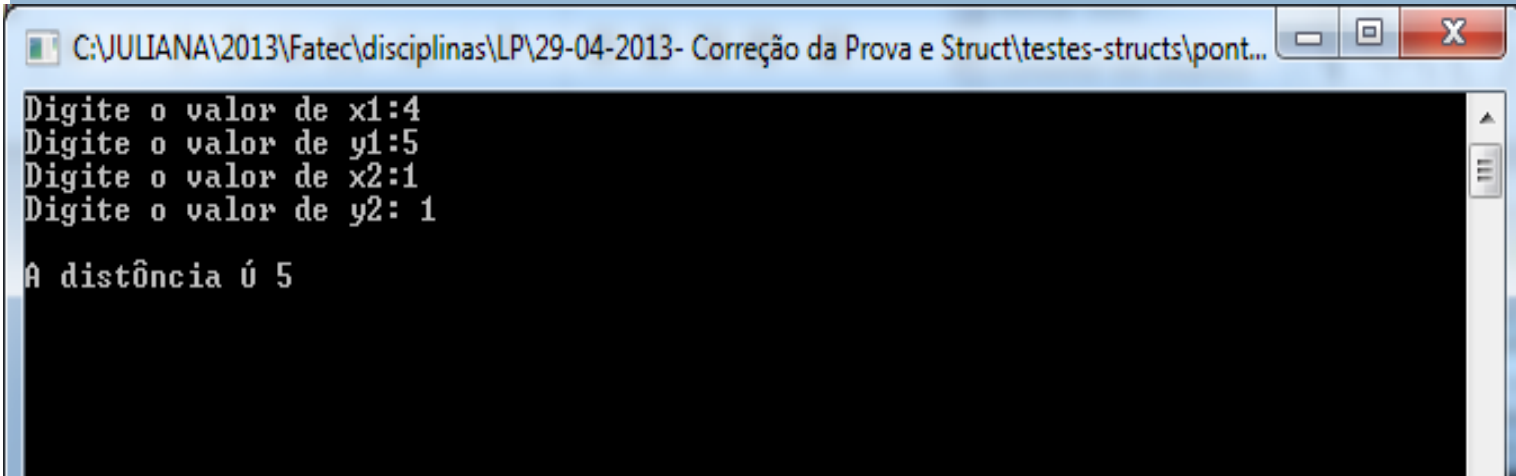
```
}
```

# Exercícios de Fixação - Struct

- Resolva o seguinte problema utilizando struct (sem utilizar função e ponteiro):
  - ▣ Criar um tipo chamado ponto, contendo apenas a posição x e y (inteiros) do ponto.
  - ▣ Declarar 2 pontos;
  - ▣ Ler a posição (coordenadas x e y) de cada um
  - ▣ Calcular a distância entre eles.
  - ▣ Apresente no final a distância entre os dois pontos.

Cálculo da distância entre dois pontos: 
$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Funções: `sqrt(X)` e `pow(num, 2)`



```
C:\JULIANA\2013\Fatec\disciplinas\LP\29-04-2013- Correção da Prova e Struct\testes-structs\pont...  
Digite o valor de x1:4  
Digite o valor de y1:5  
Digite o valor de x2:1  
Digite o valor de y2: 1  
A distância é 5
```

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$