



# Estrutura de Dados

Ricardo José Cabeça de Souza

[www.ricardojcsouza.com.br](http://www.ricardojcsouza.com.br)

[ricardo.souza@ifpa.edu.br](mailto:ricardo.souza@ifpa.edu.br)

Parte 7



# ESTRUTURAS

- **O TIPO ESTRUTURA**

- Tipo de dado cujos campos são compostos de vários valores de tipos mais simples
- Serve basicamente para agrupar diversas variáveis dentro de um único contexto



# ESTRUTURAS

- **O TIPO ESTRUTURA**

- Exemplo

```
/* Captura e imprime as coordenadas de um ponto qualquer */  
  
#include <stdio.h>  
  
struct ponto {  
    float x;  
    float y;  
};  
  
int main (void)  
{  
    struct ponto p;  
  
    printf("Digite as coordenadas do ponto(x y): ");  
    scanf("%f %f", &p.x, &p.y);  
    printf("O ponto fornecido foi: (%.2f,%.2f)\n", p.x, p.y);  
    return 0;  
}
```



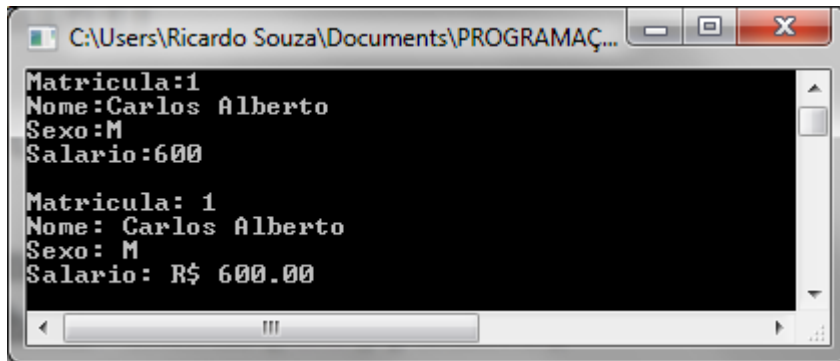
# ESTRUTURAS

- **TIPO ESTRUTURA**
  - Formato:

```
struct nome_tipo
{
tipo_1 nome_var_1;
tipo_2 nome_var_2;
...
tipo_n nome_var_n;
};
nome_tipo VARIAVEL;
```

# ESTRUTURAS

- O TIPO ESTRUTURA
  - Exemplo:



```
C:\Users\Ricardo Souza\Documents\PROGRAMAÇ...  
Matricula:1  
Nome: Carlos Alberto  
Sexo: M  
Salario: 600  
  
Matricula: 1  
Nome: Carlos Alberto  
Sexo: M  
Salario: R$ 600.00
```

```
#include <stdio.h>  
#include <conio.h>  
main()  
{  
    struct novotipo  
    {  
        int MAT;  
        char NOME[60], SEXO;  
        float SAL;  
    };  
    novotipo REG;  
    //Entrada de dados  
    printf("Matricula:");  
    scanf("%d", &REG.MAT);  
    fflush(stdin);  
    printf("Nome:");  
    gets(REG.NOME);  
    printf("Sexo:");  
    scanf("%c", &REG.SEXO);  
    printf("Salario:");  
    scanf("%f", &REG.SAL);  
    //Exibicao dos dados  
    printf("\nMatricula: %d", REG.MAT);  
    printf("\nNome: %s", REG.NOME);  
    printf("\nSexo: %c", REG.SEXO);  
    printf("\nSalario: R$ %.2f", REG.SAL);  
    getch();  
}
```





# ESTRUTURAS

- **VETOR DE ESTRUTURA**

– Formato:

```
struct nome_tipo
```

```
{
```

```
tipo_1 nome_var_1;
```

```
tipo_2 nome_var_2;
```

```
...
```

```
tipo_n nome_var_n;
```

```
};
```

```
nome_tipo VETOR[TAMANHO];
```

# ESTRUTURAS

- VETOR DE ESTRUTURA
  - Exemplo:



```
Matricula:1
Nome: Maria
Sexo: F
Salario: 700
Matricula:2
Nome: Joao
Sexo: M
Salario: 600
Matricula:3
Nome: Carlos
Sexo: M
Salario: 545

Matricula: 1
Nome: Maria
Sexo: F
Salario: R$ 700.00

Matricula: 2
Nome: Joao
Sexo: M
Salario: R$ 600.00

Matricula: 3
Nome: Carlos
Sexo: M
Salario: R$ 545.00
```

```
#include <stdio.h>
#include <conio.h>
main()
{
    struct novotipo
    {
        int MAT;
        char NOME[60], SEXO;
        float SAL;
    };
    novotipo REG[3];
    int I;
    //Entrada de dados
    for (I=0; I<3; I++)
    {
        printf("Matricula:");
        scanf("%d", &REG[I].MAT);
        fflush(stdin);
        printf("Nome:");
        gets(REG[I].NOME);
        printf("Sexo:");
        scanf("%c", &REG[I].SEXO);
        printf("Salario:");
        scanf("%f", &REG[I].SAL);
    }
    //Exibicao dos dados
    for (I=0; I<3; I++)
    {
        printf("\nMatricula: %d", REG[I].MAT);
        printf("\nNome: %s", REG[I].NOME);
        printf("\nSexo: %c", REG[I].SEXO);
        printf("\nSalario: R$ %.2f\n", REG[I].SAL);
    }
    getch();
}
```



# ESTRUTURAS

- **CRIAÇÃO DE NOVOS TIPOS**

- Podemos usar o nome Real como um mnemônico para o tipo float

```
typedef float Real;
```

- **Vetor** como um tipo que representa um vetor de quatro elementos
- A partir dessas definições, podemos declarar variáveis usando estes mnemônicos

```
typedef float Vetor[4];      Vetor v;  
...  
v[0] = 3;  
...
```





# ESTRUTURAS

- **CRIAÇÃO DE NOVOS TIPOS**

- A sintaxe de um **typedef** pode parecer confusa, mas é equivalente à da declaração de variáveis
- Por exemplo, na definição abaixo:  
**typedef float Vector[4];**
- Se omitíssemos a palavra **typedef**, estaríamos declarando a variável **Vector** como sendo um vetor de 4 elementos do tipo **float**
- Com **typedef**, estamos definindo um nome (**Vector**) que representa o **tipo vetor de 4 elementos float**



# ESTRUTURAS

- **CRIAÇÃO DE NOVOS TIPOS**

- Podemos definir a estrutura e associar mnemônicos para elas em um mesmo comando

```
typedef struct ponto {  
    float x;  
    float y;  
} Ponto, *PPonto;
```



# ESTRUTURAS

- CRIAÇÃO DE NOVOS TIPOS

```
struct ponto {  
    float x;  
    float y;  
};
```

`struct ponto p;` Definindo uma variável para a estrutura.

`struct ponto *pp;` Definindo um ponteiro para a estrutura.

```
(*pp).x = 12.0;
```



# ESTRUTURAS

- **CRIAÇÃO DE NOVOS TIPOS**

- O acesso de campos de estruturas é tão comum em programas C que a linguagem oferece outro operador de acesso
- Ele permite acessar campos a partir do ponteiro da estrutura
- Este operador é composto por um traço seguido de um sinal de maior, formando uma seta (->)

```
(*pp) .x = 12.0;
```

```
pp->x = 12.0;
```

# Estrutura de Dados



- **REFERÊNCIAS**
- Tenenbaum, Aaron M. Langsam, Yedidiah, Augenstein, Moshe J. **Estruturas de dados usando C**. São Paulo : MAKRON *Books*, 1995.
- Veloso, Paulo. et. al. **Estrutura de dados**. Rio de Janeiro: Campus, 1986.
- Moraes, Celso Roberto. **Estrutura de dados e algoritmos**. 2. ed. São Paulo: Futura, 2003.
- Celes, W. Rangel, J. L. **Curso de Estrutura de Dados**. PUC-Rio, 2002.
- W. Celes, R. Cerqueira, J.L. Rangel. **Introdução a Estruturas de Dados - com técnicas de programação em C**. Rio de Janeiro: Campus, 2004.