

# Estruturas, Enumerações e Tipos

## PE-08 – v1.0

**Prof. Paulo Joia Filho**

- 1 Introdução
- 2 Estruturas, Enumerações e Tipos
- 3 Conclusões

- 1 **Introdução**
  - Objetivos
- 2 Estruturas, Enumerações e Tipos
- 3 Conclusões

# Objetivos

Os principais objetivos desta aula são:

- Apresentar o conceito de estruturas em C.
- Entender enumerações e tipos definidos pelo programador.
- Explorar a utilização de estruturas e enumerações.

# Objetivos

Os principais objetivos desta aula são:

- Apresentar o conceito de estruturas em C.
- Entender enumerações e tipos definidos pelo programador.
- Explorar a utilização de estruturas e enumerações.

# Objetivos

Os principais objetivos desta aula são:

- Apresentar o conceito de estruturas em C.
- Entender enumerações e tipos definidos pelo programador.
- Explorar a utilização de estruturas e enumerações.

### 1 Introdução

### 2 Estruturas, Enumerações e Tipos

- Estruturas
- Enumerações e Tipos

### 3 Conclusões

# Estruturas (Struct)

- O que conhecemos como registros em outras linguagens de programação, em C é conhecido como **struct** (o nome é uma abreviatura de *structure*).
- Uma estrutura é um coleção de variáveis referenciadas por um nome, fornecendo uma maneira conveniente de atribuir informações (variáveis) relacionadas de forma agrupada.
- A definição de uma estrutura é um modelo a ser seguido por todas as variáveis de seu tipo.
- As *variáveis* que compreendem a estrutura são também conhecidas como *campos* ou *atributos* da estrutura.



# Estruturas (Struct)

- Sintaxe:

```
struct type_name {  
    member_type1 member_name1;  
    member_type2 member_name2;  
    .  
    .  
} object_names;
```

- Exemplo para criar uma estrutura de representação de um produto:

```
struct produto {  
    int peso;  
    double preco;  
};  
/* Para declarar uma variável  
do tipo produto: */  
struct produto melao;
```

```
struct produto {  
    int peso;  
    double preco;  
} maca, banana, melao;  
/* maca, banana, melao  
são variáveis declaradas  
do tipo produto */
```

# Estruturas (Struct)

## Importante:

Depois do símbolo de fecha chaves ( `}` ) da estrutura é necessário colocar um ponto e vírgula ( `;` ).

- Isso é necessário uma vez que a estrutura pode ser também declarada no escopo local.
- Por questões de simplificações, e por se tratar de um **novo tipo**, é possível logo na definição da **struct** definir algumas variáveis desse tipo.
- Para isso, basta colocar os nomes das variáveis declaradas após fechar chaves ( `}` ) da estrutura e antes do ponto e vírgula ( `;` ).

# Estruturas (Struct)

[dados\_pessoais1.c]

## Exemplo 1 (Atribuição e acesso às variáveis dentro da estrutura)

```
1 #include <stdio.h>
  #include <string.h>
3
4 struct cadastro{
5     char nome[50];
6     char endereco[80];
7     int idade;
8 };
9
10 int main() {
11     struct cadastro c;           // define variável do tipo da estrutura
12     strcpy(c.nome, "Carlos");    // atribui valor aos campos
13     strcpy(c.endereco, "Avenida Brasil, 1082");
14     c.idade = 18;
15
16     printf("Dados Pessoais \nNome: %s; Idade: %d anos;\n", c.nome, c.idade);
17     printf("Endereço: %s.\n", c.endereco);
18     return 0;
19 }
```

O que será impresso?



# Estruturas (Struct)

`[dados_pessoais1.c]`

## Exemplo 1 (Atribuição e acesso às variáveis dentro da estrutura)

```
1 #include <stdio.h>
  #include <string.h>
3
4 struct cadastro{
5     char nome[50];
6     char endereco[80];
7     int idade;
8 };
9
10 int main() {
11     struct cadastro c;           // define variável do tipo da estrutura
12     strcpy(c.nome, "Carlos");    // atribui valor aos campos
13     strcpy(c.endereco, "Avenida Brasil, 1082");
14     c.idade = 18;
15
16     printf("Dados Pessoais \nNome: %s; Idade: %d anos;\n", c.nome, c.idade);
17     printf("Endereço: %s.\n", c.endereco);
18     return 0;
19 }
```

### O que será impresso?

```
Dados Pessoais
Nome: Carlos; Idade: 18 anos;
Endereço: Avenida Brasil, 1082.
```



# Estruturas (Struct)

`[dados_pessoais2.c]`

## Exemplo 2 (Lendo do teclado as variáveis da estrutura)

```
1 #include "../common/pe_utility2.h"
```

O que será impresso?

```
3 struct cadastro{
4     char nome[50];
5     char endereco[80];
6     int idade;
7 };
```

```
9 int main() {
10     struct cadastro c;           // define variável do tipo da estrutura
11     printf("Informe o nome: ");
12     fgets(c.nome, 50, stdin);    // lê do teclado e armazena nos campos
13     printf("Informe o endereco: ");
14     fgets(c.endereco, 80, stdin);
15     printf("Informe a idade: ");
16     scanf("%d", &c.idade);
17     printf("Dados Pessoais \nNome: %s; Idade: %d anos;\n", pe_strtrim(c.nome), c.idade);
18     printf("Endereço: %s.\n", pe_strtrim(c.endereco));
19     return 0;
20 }
```



# Estruturas (Struct)

[dados\_pessoais2.c]

## Exemplo 2 (Lendo do teclado as variáveis da estrutura)

```
1 #include "../common/pe_utility2.h"
```

```
3 struct cadastro{
4     char nome[50];
5     char endereco[80];
6     int idade;
7 };
```

```
9 int main() {
10     struct cadastro c;           // define variável do tipo da estrutura
11     printf("Informe o nome: ");
12     fgets(c.nome, 50, stdin);    // lê do teclado e armazena nos campos
13     printf("Informe o endereco: ");
14     fgets(c.endereco, 80, stdin);
15     printf("Informe a idade: ");
16     scanf("%d", &c.idade);
17     printf("Dados Pessoais \nNome: %s; Idade: %d anos;\n", pe_strtrim(c.nome), c.idade);
18     printf("Endereço: %s.\n", pe_strtrim(c.endereco));
19     return 0;
20 }
```

### O que será impresso?

```
Informe o nome: Jose da Silva
Informe o endereco: Rua Portugal, 328
Informe a idade: 32
Dados Pessoais
Nome: Jose da Silva; Idade: 32 anos;
Endereço: Rua Portugal, 328
```

# Estruturas (Struct)

## Função typedef

- A instrução **typedef** permite denotar novos nomes à linguagem.
- Assim, pode-se utilizar o comando **typedef** para simplificar a declaração de variáveis de estrutura.
- A declaração se faz como um tipo primitivo (**char**, **int**, etc).
- Sintaxe:

**typedef** existing\_type new\_type\_name;

- Exemplo:

```
typedef struct {  
    int peso;  
    double preco;  
} produto;  
// declaração  
produto banana;
```

```
typedef char field[50];  
  
// declaração  
field name;
```

# Estruturas (Struct)

`[dados_pessoais3.c]`

## Exemplo 3 (Atribuição e acesso ao elementos dentro da estrutura)

```
1 #include <stdio.h>
  #include <string.h>
```

O que será impresso?

```
3
  typedef struct{
5     char nome[50];
      char endereco[80];
7     int idade;
  } cadastro;
```

```
9
  int main() {
11     cadastro c;           // define variável do tipo da estrutura
      strcpy(c.nome, "Carlos"); // atribui valor aos campos
13     strcpy(c.endereco, "Avenida Brasil, 1082");
      c.idade = 18;
15
      printf("Dados Pessoais \nNome: %s; Idade: %d anos;\n", c.nome, c.idade);
17     printf("Endereço: %s.\n", c.endereco);
      return 0;
19 }
```





# Estruturas (Struct)

`[dados_pessoais3.c]`

## Exemplo 3 (Atribuição e acesso ao elementos dentro da estrutura)

```
1 #include <stdio.h>
  #include <string.h>
```

```
3
  typedef struct{
5     char nome[50];
      char endereco[80];
7     int idade;
  } cadastro;
```

```
9
10 int main() {
11     cadastro c;           // define variável do tipo da estrutura
      strcpy(c.nome, "Carlos"); // atribui valor aos campos
13     strcpy(c.endereco, "Avenida Brasil, 1082");
      c.idade = 18;
15
      printf("Dados Pessoais \nNome: %s; Idade: %d anos;\n", c.nome, c.idade);
17     printf("Endereço: %s.\n", c.endereco);
      return 0;
19 }
```

### O que será impresso?

```
Dados Pessoais
Nome: Carlos; Idade: 18 anos;
Endereço: Avenida Brasil, 1082.
```

# Estruturas (Struct)

[dados\_pessoais4.c]

## Exemplo 4 (Leitura e acesso ao elementos dentro da estrutura)

```
1 #include "../common/pe_utility2.h"
```

O que será impresso?

```
3 typedef struct {  
4     char nome[50];  
5     char endereco[80];  
6     int idade;  
7 } cadastro;
```

```
9 int main() {  
10     cadastro c; // define variável do tipo da estrutura  
11     printf("Informe o nome: ");  
12     fgets(c.nome, 50, stdin); // lê do teclado e armazena nos campos  
13     printf("Informe o endereco: ");  
14     fgets(c.endereco, 80, stdin);  
15     printf("Informe a idade: ");  
16     scanf("%d", &c.idade);  
17     printf("Dados Pessoais \nNome: %s; Idade: %d anos;\n", pe_strtrim(c.nome), c.idade);  
18     printf("Endereço: %s.\n", pe_strtrim(c.endereco));  
19     return 0;  
}
```

# Estruturas (Struct)

`[dados_pessoais4.c]`

## Exemplo 4 (Leitura e acesso ao elementos dentro da estrutura)

```
1 #include "../common/pe_utility2.h"
```

```
3 typedef struct {  
4     char nome[50];  
5     char endereco[80];  
6     int idade;  
7 } cadastro;
```

```
9 int main() {  
10     cadastro c; // define variável do tipo da estrutura  
11     printf("Informe o nome: ");  
12     fgets(c.nome, 50, stdin); // lê do teclado e armazena nos campos  
13     printf("Informe o endereco: ");  
14     fgets(c.endereco, 80, stdin);  
15     printf("Informe a idade: ");  
16     scanf("%d", &c.idade);  
17     printf("Dados Pessoais \nNome: %s; Idade: %d anos;\n", pe_strtrim(c.nome), c.idade);  
18     printf("Endereço: %s.\n", pe_strtrim(c.endereco));  
19     return 0;  
}
```

### O que será impresso?

```
Informe o nome: Jose da Silva  
Informe o endereco: Rua Portugal, 328  
Informe a idade: 32  
Dados Pessoais  
Nome: Jose da Silva; Idade: 32 anos;  
Endereço: Rua Portugal, 328
```

# Enumerações

- Uma **enum** é um conjunto de constantes inteiras que determina quais os possíveis valores de uma variável desse tipo.

- Sintaxe:

```
typedef enum { value1, value2, ... } type_name;
```

- Exemplo:

```
typedef enum { x, y, z } exemplo;
```

- Por padrão, os valores começam de zero. Por exemplo, na enumeração definida acima, as variáveis possuem os seguintes valores:
  - $x = 0$
  - $y = 1$
  - $z = 2$

# Enumerações

## Alterando o valor das variáveis de uma enumeração

- Contudo, as variáveis podem ter sua numeração alterada com uma simples atribuição no momento da definição.

- Exemplo:

```
typedef enum { x, y = 10, z } exemplo;
```

- Assim, as variáveis possuem os seguintes valores:
  - x = 0
  - y = 10
  - z = 11

# Enumerações

[tipo\_boolean.c]

## Exemplo 5 (Definindo o tipo **boolean**)

```
1  #include <stdio.h>

3  typedef enum {false, true} boolean;

5  int main() {
    boolean b;

7
    b = (3 > 1);
9    if (b) {
        printf("3 > 1\n");
11   }

13   b = !b;

15   if (b == true) {
        printf("3 <= 1\n");
17   }

19   return 0;
}
```

O que será impresso?

# Enumerações

[tipo\_boolean.c]

## Exemplo 5 (Definindo o tipo **boolean**)

```
1  #include <stdio.h>

3  typedef enum {false, true} boolean;

5  int main() {
    boolean b;

7
    b = (3 > 1);
9    if (b) {
        printf("3 > 1\n");
11   }

13   b = !b;

15   if (b == true) {
        printf("3 <= 1\n");
17   }

19   return 0;
}
```

O que será impresso?

3 > 1

# Enumerações

[tipo\_diaSemana.c]

## Exemplo 6 (Definindo o tipo **diaSemana**)

```
1  #include <stdio.h>

3  typedef enum { domingo, segunda, terca, quarta,
    quinta, sexta, sabado } diaSemana;

5

7  int main() {
    diaSemana d = segunda; /* Poderia informar 1 */

9      if (d == sabado || d == domingo) {
        printf("Fim de Semana!\n");
11     } else {
        printf("Dia Util.\n");
13     }

15     return 0;
}
```

O que será impresso?



# Enumerações

[tipo\_diaSemana.c]

## Exemplo 6 (Definindo o tipo **diaSemana**)

```
1  #include <stdio.h>

3  typedef enum { domingo, segunda, terca, quarta,
    quinta, sexta, sabado } diaSemana;

5

7  int main() {
    diaSemana d = segunda; /* Poderia informar 1 */

9      if (d == sabado || d == domingo) {
        printf("Fim de Semana!\n");
11     } else {
        printf("Dia Util.\n");
13     }

15     return 0;
}
```

O que será impresso?

Dia Util.

# Enumerações

## Observações Importantes:

- Observe que **segunda** refere-se ao inteiro 1.  
`d = segunda` equivale a `d = 1`
- Isso indica que **segunda** não é o string "**segunda**".  
`d = segunda` **não equivale a** `d = "segunda"`

## 1 Introdução

## 2 Estruturas, Enumerações e Tipos

## 3 Conclusões

- Exercícios de aprendizagem
- Considerações finais
- Referências bibliográficas

# Exercícios de Aprendizagem

## Exercício 1

Faça um programa para armazenar os dados de um cliente na **struct cliente**, definida abaixo:

```
#define MAX_NOME          80
#define MAX_LOGRADOURO 100

struct endereco {
    char logradouro[MAX_LOGRADOURO];
    int numero;
};

struct cliente {
    int codigo;
    char nome[MAX_NOME];
    struct endereco end;
};
```

Note que o campo endereço deve ser armazenado dentro de outra estrutura, contendo logradouro e número.

# Exercícios de Aprendizagem

## Exercício 1

### Exemplo de Funcionamento

```
Informe nome: Universidade Federal do ABC  
Logradouro: R. Oratorio  
Numero: 1234
```

```
Imprimindo dados do cliente 311:  
Nome: Universidade Federal do ABC  
Endereco: R. Oratorio, 1234
```

### Notas:

- O código do cliente deve ser um número aleatório entre 1 e 1000.
- Os demais campos (nome, logradouro e número) devem ser informados pelo usuário.

# Exercícios de Aprendizagem

## Exercício 2

Modifique o programa anterior para armazenar os dados de um veículo com base na seguinte estrutura:

```
#define MAX_NOME 80
#define MAX_MODELO 50

typedef struct {
    char nome[MAX_NOME];
    char cpf[15];
} proprietario_t;

typedef struct {
    char modelo[MAX_MODELO];
    int ano;
    char placa[10];
    proprietario_t proprietario;
} veiculo_t;
```

Note que, neste caso, a estrutura foi declarada como um tipo.

# Exercícios de Aprendizagem

## Exercício 2

### Exemplo de Funcionamento

VEICULO:

Modelo: Corsa

Ano: 2010

Placa: HLP-3587

PROPRIETARIO:

Nome: Alan Turing

CPF: 320.451.237-48

### Nota:

- Preencha os campos com valores fixos (sem usar `scanf` ou `fgets`).

# Exercícios de Aprendizagem

## Exercício 3

Faça um programa para armazenar o peso ( $p$ ) e a altura ( $h$ ) de  $n$  pessoas em uma **struct**. Em seguida, calcule o Índice de Massa Corporal (IMC) de cada pessoa, onde:

$$IMC = \frac{p}{h^2}$$

### Notas:

- O número de pessoas ( $n$ ) deve ser informado pelo usuário, assim como o peso e a altura de cada pessoa.
- Utilize um vetor de estruturas para realizar esta tarefa (alocação estática de memória).



# Exercícios de Aprendizagem

## Exercício 3

### Exemplo de Funcionamento

Nr de pessoas: 2

Entre as medidas:

Peso 1: 79.5

Altura 1: 1.87

Peso 2: 56

Altura 2: 1.57

Índice de massa corporal:

Pessoa 1: 22.73

Pessoa 2: 22.72

# Exercícios de Aprendizagem

## Exercício 4

Implemente o [Exercício 3](#) usando alocação dinâmica de memória. Declare um ponteiro para a estrutura de pesos e alturas.

### Observação:

- Não esqueça de liberar a memória no final do processo.

# Considerações Finais

- ❖ Nesta aula foram apresentados os principais conceitos relacionados a:
  - Estruturas em C.
  - Enumerações e tipos.
- ✓ *É importante rever os conceitos apresentados na aula e consultar a bibliografia sugerida sobre o assunto.*

# Referências Bibliográficas I



Aguilar, L. J. (2008).

*Programação em C++: Algoritmos, Estruturas de Dados e Objetos.*  
McGraw-Hill, São Paulo.



Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. (2002).

*Algoritmos: Teoria e Prática.*  
Elsevier, Rio de Janeiro.



Drozdek, A. (2009).

*Estrutura de Dados e Algoritmos em C++.*  
Cengage Learning, São Paulo.



Forbellone, A. L. V. e Eberspacher, H. F. (2005).

*Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados.*  
Pearson Prentice Hall, São Paulo, 3 edition.



Knuth, D. E. (2005).

*The Art of Computer Programming.*  
Addison-Wesley, Upper Saddle River, NJ, USA.



Pinheiro, F. d. A. C. (2012).

*Elementos de Programação em C.*  
Bookman, Porto Alegre.

# Referências Bibliográficas II



Sedgewick, R. (1998).

*Algorithms in C: Parts 1-4, Fundamentals, Data Structures, Sorting, Searching.*  
Addison-Wesley, Boston, 3rd edition.



Szwarcfiter, J. L. e Markenzon, L. (1994).

*Estruturas de Dados e Seus Algoritmos.*  
LTC, Rio de Janeiro.



Tenenbaum, A. A., Langsam, Y., e Augenstein, M. J. (1995).

*Estruturas de Dados Usando C.*  
Makron Books, São Paulo.



Terra, R. (2014).

Linguagem C - Notas de Aula.

Universidade Federal de Lavras (UFLA).

Disponível em:

[http://professores.dcc.ufla.br/~terra/public\\_files/2014\\_apostila\\_c\\_ansi.pdf](http://professores.dcc.ufla.br/~terra/public_files/2014_apostila_c_ansi.pdf). Acesso em: 2018-09-16.