
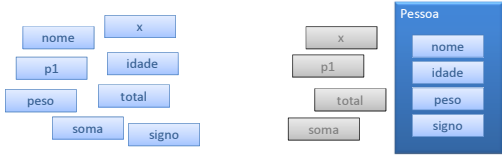





Estrutura (struct) 

- Conjunto de variáveis, de mesmos ou diferentes tipos
- Dados variados, porém relacionados entre si com um mesmo propósito
- Estrutura vs. OO



The diagram illustrates the difference between a struct and an object-oriented approach. On the left, under the 'Estrutura' (struct) approach, variables are organized into a single container labeled 'Pessoa'. Inside this container, there are boxes for 'nome', 'idade', 'peso', 'total', 'soma', and 'signo'. On the right, under the 'OO' (object-oriented) approach, the same variables are shown as separate, ungrouped boxes: 'nome', 'idade', 'peso', 'total', 'soma', and 'signo'.

Estrutura (struct) 

- Instrução `struct`

Sintaxe

```
struct nome {
    tipo nome;
    ...
    tipo nome;
} {variáveis};
```

Exemplo

```
struct pessoa {
    char nome[30];
    unsigned int idade;
    double peso;
} p1;
```

- Tipo de dado: `struct nome_struct`

```
struct pessoa p2, p3;
```

- Acesso (.)

```
p1.idade = 30;
```

Matrizes de estruturas



- **struct** com []

Sintaxe

```
struct nome {
    tipo nome;
    ...
    tipo nome;
} variáveis[ ];
```

Exemplo

```
struct pessoa {
    char nome[30];
    unsigned int idade;
    double peso;
} turma[5];

strcpy(turma[0].nome, "Andre");
turma[0].idade = 30;
turma[0].peso = 68.5;
```

Estruturas em funções



- Utilizar como um tipo de dado simples
- Declarar o parâmetro da função como o tipo de dado:

struct nome_struct nome_parâmetro

```
void imprimePessoa(struct pessoa p)
{
    printf("%s tem %d anos e pesa %.2f kg.", p.nome, p.idade, p.peso);
}
```

Ponteiros de estruturas



- Similar aos demais usos de ponteiro

```
struct pessoa p1;
struct pessoa *p2;
p2 = &p1;
```

- Acesso via operador seta (->)

```
printf("Nome: %s", p2->nome);
```

Estruturas dentro de estruturas



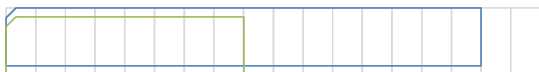
- Estrutura comporta qualquer tipo de dado
- Se uma estrutura é um tipo de dado, então comporta uma estrutura
- Comporta ponteiros, matrizes

```
struct geral {  
    int x;  
    int *y;  
    int z[15];  
    struct pessoa p1;  
    struct pessoa galera[30];  
    struct pessoa *p2;  
};
```

Unões



- Instrução union
- Compartilhamento de memória entre variáveis



Tipos definidos pelo usuário



- Instrução typedef
- Permite criar apelidos para tipos de dados