

ESTRUTURAS / REGISTROS (STRUCT)

Estruturas/Registros – é o tipo de dados definido pelo usuário que permite agregar várias variáveis de tipos de dados diferentes.

Cada unidade de informação contida em uma estrutura é chamada de campo.

Os campos podem ser de diferentes tipos primitivos, ou ainda, podem representar outras estruturas.

Desse modo, as estruturas (registros) são conhecidos como variáveis compostas heterogêneas.

Estruturas são usadas para armazenar um conjunto de informações sobre uma entidade (objeto).

CRIAÇÃO DE UMA ESTRUTURA

Para criar uma estrutura deve-se colocar uma palavra-chave **struct**. Essa palavra-chave define um novo tipo de dados com mais de um campo.

Exemplo:

Formato geral	Exemplo
<pre>struct Nome { member definition; member definition; ... member definition; }</pre>	<pre>struct Books { char title[50]; char author[50]; int book_id; }</pre>

Nesse exemplo é criada uma estrutura chamada **Books** com 3 campos:

- **title** – é um vetor de 50 caracteres
- **autor** – é um vetor de 50 caracteres
- **book_id** – é um número inteiro

	Books
title	
author	
book_id	

DECLARAÇÃO DE UMA VARIÁVEL DO TIPO ESTRUTURA

A declaração de variáveis desse tipo pode ser feita logo depois da declaração de uma estrutura:

Exemplo:

```
struct Books
{
    char title[50];
    char author[50];
    int book_id;
} book;
```

Nesse exemplo a declaração de uma variável **book** do tipo **Books** é feita logo depois da declaração da própria estrutura.

As variáveis também poderão ser declarados ao longo do programa usando a palavra-chave **struct** junto com o **nome** da estrutura previamente criada pelo usuário.

Exemplo:

```
struct Books Book1;
struct Books Book2;
```

Nesse exemplo são criadas (declaradas) duas variáveis **Book1** e **Book2** do tipo **Books**.

ACESSO AOS CAMPOS DE UMA ESTRUTURA

Para acessar os campos (ou membros) de uma estrutura é usado operador (.) que é colocado entre o nome da variável e nome do campo:

Exemplo:

```
Book1.book_id = 123;
```

Nesse exemplo é atribuído o valor **123** para campo **book_id** da variável **Book1**.

No próximo exemplo será usada uma função específica para processar as sequências de caracteres. Essas funções fazem parte da biblioteca **string.h** e algumas delas estão apresentados na tabela a seguir:

strcpy(s1, s2);	copia string s2 em s1
strcat(s1, s2);	concatena (adiciona) string s2 no final do string s1
strlen(s1);	retorna o tamanho do string s1
strcmp(s1, s2);	retorna 0 se s1 é igual a s2 ; menor que 0 se s1<s2 ; maior que 0, se s1>s2
strchr(s1, ch);	retorna o ponteiro para primeira ocorrência do caractere ch em string s1
strstr(s1, s2);	retorna o ponteiro para primeira ocorrência da string s2 em string s1

Exemplo 1: Criação de uma estrutura Books

```
1  #include <stdio.h>
2  #include <string.h>
3
4  struct Books
5  {
6      int book_id;
7      char title[50];
8      char author[50];
9  };
10 //=====
11 int main( )
12 {
13     struct Books Book1; /* Declare Book1 of type Book */
14     struct Books Book2; /* Declare Book2 of type Book */
15
16     /* book 1 specification */
17     strcpy( Book1.title, "Logica de Programação");
18     strcpy( Book1.author, "FORBELLONE");
19     Book1.book_id = 101;
20
21     /* book 2 specification */
22     strcpy( Book2.title, "Treinamento em Linguagem C");
23     strcpy( Book2.author, "MIZRAHI");
24     Book2.book_id = 102;
25
26     /* print Book1 info */
27     printf( "\n\n_____ Livro 1 (Book1) _____\n");
28     printf( "Codigo (book_id) : %d\n", Book1.book_id);
29     printf( "Titulo (title)   : %s\n", Book1.title);
30     printf( "Autor  (author)   : %s\n", Book1.author);
31
32     /* print Book2 info */
33     printf( "\n\n_____ Livro 2 (Book2) _____\n");
34     printf( "Codigo (book_id) : %d\n", Book2.book_id);
35     printf( "Titulo (title)   : %s\n", Book2.title);
36     printf( "Autor  (author)   : %s\n", Book2.author);
37
38     return 0;
39 }
```

_____ Livro 1 (Book1) _____
Codigo (book_id) : 101
Titulo (title) : Logica de Programação
Autor (author) : FORBELLONE

_____ Livro 2 (Book2) _____
Codigo (book_id) : 102
Titulo (title) : Treinamento em Linguagem C
Autor (author) : MIZRAHI

CRIAÇÃO DE UM VETOR DE ESTRUTURAS

Em C/C++ é possível criar um vetor de estruturas. Neste caso cada elemento do vetor será uma estrutura.

O exemplo a seguir cria um vetor **Lib**, que contem 5 elementos de estrutura **Books**

```
const int lib_size = 5;  
struct Books Lib[lib_size];
```

Lib

	0	1	2	3	4
title					
author					
book_id					

Acesso a um campo específico de uma estrutura que faz parte de um vetor será feito dessa forma:

```
Lib[0].book_id = 100;
```