

INF01202

Algoritmos e Programação

Modalidade Ensino a Distância

Linguagem C

Tópico 16: Estruturas - introdução

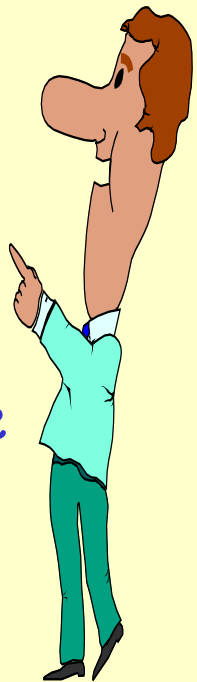
Até agora ...

➤ Armazenamos dados em **variáveis simples**:

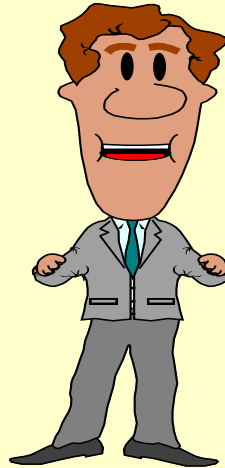
- char
- int
- float
- double

➤ Armazenamos conjuntos de valores relacionados em **arranjos**:

- Vários elementos de mesmo tipo referenciados por um mesmo identificador

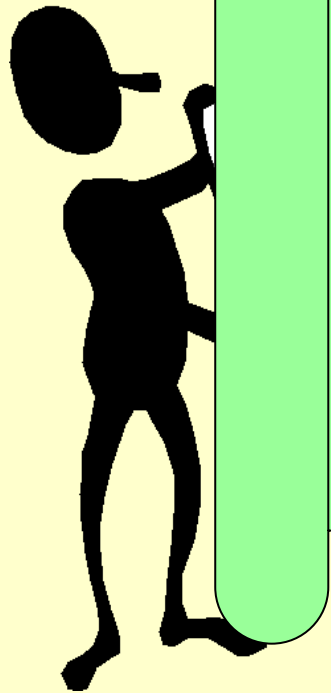


Por que estruturas ?



- As estruturas em C permitem agrupar em uma única entidade elementos de tipos diferentes!

O que é uma estrutura ?



Conjunto de uma ou mais variáveis (campos) relacionadas, agrupadas sob um único nome, de forma a facilitar a sua referência!

Podem conter elementos com qualquer tipo de dado válido em C:

- Básicos
- Arranjos
- Ponteiros
- Ou mesmo outras estruturas!

Exemplo 1:

funcionario

Nome da
Estrutura




cod

nome

cargo

depto

salario



Campos	Tipo
cod	int
nome	char[]
cargo	char []
depto	int
salario	float

Diferença entre arranjo e estrutura:

Arranjo (elementos homogêneos)

- Todos os elementos do mesmo tipo.
- Um só nome, com índice para identificar cada elemento.

Estrutura (elementos heterogêneos)

- Elementos podem ser de tipos diferentes.
- Cada elemento é identificado através de um nome diferente (campo).

Arranjo

nota

0	1	2	3	4	5	6	7	8

Índices
Identificando cada elemento

Nome
Comum para todos os elementos

Estrutura

funcionario

--	--	--	--	--

cod

nome

cargo

depto

salário

Nome
Comum para todos os elementos

Campos
Um nome diferente para cada campo

Declaração de Estruturas

→ Palavra reservada que informa ao compilador o *gabarito* do novo tipo de dado

→ Especificador de tipo

```
struct nome_da_estrutura
{
    tipo1    campo1, campo2;
    ...
    tipon    campo;
};
```

Exemplo 1:

```
struct pessoa
{
    int idade;
    char sexo, est_civil;
    char nome[40];
    float salario;
};
```

O código do exemplo acima apenas definiu o formato (ou gabarito) de um novo tipo de dados.

pessoa não é uma variável, mas sim o nome pelo qual será reconhecido este novo tipo de dados.

Declaração de variáveis tipo estrutura...

- Para se trabalhar com os dados de uma **estrutura** é necessário, primeiramente, declarar uma ou mais **variáveis** com o tipo da estrutura que desejamos. Existem duas formas p/ essa declaração:

Junto com a declaração da estrutura

```
struct nome_tipo_estrutura
{
    tipo1  campo1, campo2;
    ...
    tipon  campo;
} v1, v2, ..., vn;
```

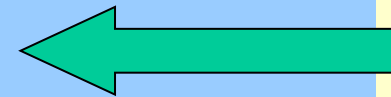
Após a declaração da estrutura

```
struct nome_tipo_estrutura v1, v2, ..., vn;
```

Exemplo 2:

```
struct pessoa
{
    int idade,
    char sexo, est_civil;
    char nome[40];
    float salario;
} professor, funcionario;
```

Variáveis tipo
pessoa



Neste caso, a estrutura tanto de *professor*, quanto de *funcionário* tem o mesmo gabarito, embora o seu conteúdo, i.e., os valores dos membros possa ser diferente.

Exemplo 3:

```
struct pessoa prof, func[100], *ptr_pessoa;
```

- `prof` é uma variável do tipo **struct pessoa**.
- `func` é um vetor de 100 elementos, sendo cada um deles uma estrutura do tipo **struct pessoa**.
- `*ptr_pessoa` é um ponteiro para o tipo **struct pessoa**.

Definição de tipos-*typedef*

- Ao se usar estruturas deve-se declarar as variáveis deste tipo SEMPRE precedidas pela palavra reservada **struct**, seguida do *nome_da_estrutura*
- Ideal seria usar uma estrutura unicamente através de uma palavra, como se faz com os tipos básicos de C (int, float,...). Esta idéia pode ser implementada através da palavra reservada **typedef**, cuja sintaxe:

typedef *tipo_existente_em_C* *sinônimo*

Onde

Tipo_existente_em_C pode ser {int, float, double, char,...}

e

Sinônimo um identificador qualquer

Declaração de Estruturas com *typedef*

A palavra typedef não cria um novo tipo. Permite apenas que um determinado tipo possa ser denominado de forma diferente, de acordo com as necessidades de programador.

Ex:

```
typedef    char    caractere
```

Esta declaração afirma que a palavra caractere será usada para o tipo char também, como sinônimo.

Declaração de Estruturas com *typedef*

EX: declare um novo tipo chamado **carro** que permita definir a estrutura **automovel**

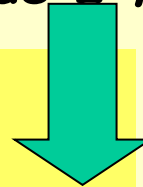
```
typedef struct automovel  
{  
    int chassis;  
    char placa[9];  
    int ano;  
    char marca[8];  
} carro;
```

Então, se pode declarar variáveis de 2 formas diferentes:

```
struct automovel Fiesta, Gol;
```

ou

```
carro Fiesta, Gol;
```



Mais um exemplo:

```
typedef struct pessoa
{
    int idade,
    char sexo, est_civil;
    char nome[40];
    float salario;
} pessoas;
```

➤ Sem typedef:

```
struct pessoa prof, func;
```

➤ Com typedef:

```
pessoas prof, func;
```

Este recurso torna o código mais "limpo"!

Declaração de estrutura sem indicação de nome...

➤ Quando usar?

- Quando você precisar de apenas uma variável de estrutura.
- Por exemplo, você declara uma variável do tipo **pessoa**, a qual servirá para: professor, funcionario, etc.

Exemplo:
struct
{
 int idade;
 char sexo, est_civil;
 char nome[40];
 float salario;
} **pessoa_info**;

Sem nome!

É possível OMITIR ou *nome_tipo_estrutura* ou *variáveis_estrutura*, mas NÃO ambos.

Neste caso, é necessário declarar uma variável.

Acesso aos dados de uma estrutura

- É possível através do operador • (ponto)

nome_variável.nome_campo

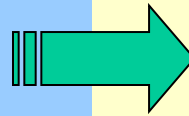


- Cada campo deve ser referenciado individualmente, indicando a qual variável tipo estrutura se refere

Acesso aos dados de uma estrutura

➤ Exemplo:

```
struct pessoa
{
    int idade,
    char sexo, est_civil;
    char nome[40];
    float salario;
} prof, func;
```



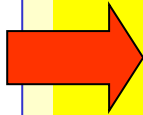
```
...
prof.idade = 32;
prof.sexo = 'm';
prof.est_civil = 's';
...
func.idade = 21;
func.sexo = prof.sexo;
...
```

prof identifica a estrutura

idade, sexo, est_civil identificam o campo da estrutura

Inicialização de estrutura na declaração

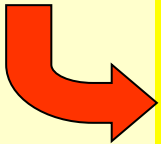
Junto com a declaração da estrutura



```
struct nome_da_estrutura
{
    tipo1    campo1, campo2;
    ...
    tipon    campo;
} v1 = {valor1, valor2, ..., valorn};
```

Os valores entre chaves correspondem aos valores da estrutura, pela ordem em que foram escritos na sua definição

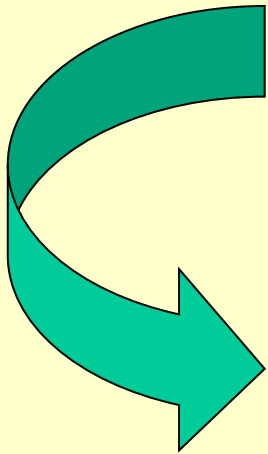
Após a declaração da estrutura



```
struct nome_da_estrutura v1 = {valor1, valor2, ..., valorn};
```

Exemplo 4:

- Inicialização junto com a declaração da estrutura:



```
struct pessoa
{
    int idade,
    char sexo, est_civil;
    char nome[40];
    float salario;
} prof = {32, 'f', 'c', "Maria", 2645.52};
```

idade ←

→ nome

→ salário

→ estado civil (s-solt.; c-casado; ...)

→ sexo (m-masc.; f-fem.)

Exemplo 5:

- Inicialização após a declaração da estrutura:

```
struct pessoa func = {32, 'f', 'c', "Maria", 2645.52};
```

Inicialização de estrutura por leitura e atribuição de dados

```
typedef struct tipofunc { //declaracao de tipo estrutura
    int cod;
    char nome[30];
    char cargo[10];
    int depto;
    float salario;    };

main ( )
{
    tipofunc funcionario; // declaracao de variavel tipo estrutura

    scanf ("%d", &funcionario.cod); // inicializacao por leitura e atribuição
    strcpy ( funcionario.nome, "Jose");
    strcpy (funcionario.cargo, "Gerente");
    scanf ("%d", &funcionario.depto);
    scanf ("%f", &funcionario.salario);

    ....
}
```

Usando estrutura dentro de estrutura (cont.)

```
typedef struct tipoendereco //declaracao de tipo estrutura
{
    char rua [50];
    int numero;
    char bairro [20];
    char cidade [30];
    char sigla_estado [3];
    longint cep;
};

typedef struct fichapessoal //declaracao de tipo estrutura
{
    char nome [50];
    long int telefone;
    tipoendereco endereco;
};

main (void)
{
    fichapessoal ficha;
    strcpy (ficha.endereco.rua,"Rua das Flores");
    ...
```



Usando estrutura dentro de estrutura

```
typedef struct tipoendereco
{
    char rua [50];
    int numero;
    char bairro [20];
    char cidade [30];
    char sigla_estado [3];
    long int CEP;
};
```

```
typedef struct fichapessoal
{
    char nome [50];
    long int telefone;
    tipoendereco endereco;
};
```

```
fichapessoal ficha;
strcpy (ficha.endereco.rua,"Rua das Flores");
scanf ("%d", &ficha.endereco.numero, num);
strcpy (ficha.endereco.bairro,"Floresta");
strcpy (ficha.endereco.cidade,"Bage");
...
```

Primeiramente é acessado o campo **endereco** da **struct ficha** e, dentro deste campo, o campo **rua** (ou numero, ou bairro, ou cidade, ou etc.).

Exercício (cont.):

1. Defina em C um novo tipo denominado **Bilhete**, conforme layout do bilhete de passagem de ônibus mostrado abaixo.

Viação VIAJE BEM

Cod. Empresa: _____

Nome do Passageiro: _____

Origem: _____

Destino: _____

Data: ____/____/____

Horário: ____:____

Assento: _____

Valor do bilhete: _____ .

```

/*
  Name: viajebemV02.cpp
  Author: Cida Souto.
  Date: 20/05/08 00:07
  Description: Definir um novo tipo bilhete e as respectivas funções que
  permitem ler e mostrar os dados de um bilhete.
*/
#include <stdio.h>
#include <stdlib.h>
typedef struct tipohorario
{
    int hora, min;
};
typedef struct tipodata
{
    int dia, mes, ano;
};
typedef struct tipobilhete
{
    int cod_e;
    char nome_p[20];
    char origem[20], destino[20];
    tipodata data_b;
    tipohorario horario_b;
    int assento;
    float valor;
};

```

Exercício:

2. A partir da estrutura definida no item 1, leia e imprima todos os dados relativos a um determinado bilhete.

```
int main ( )
{
    tipobilhete bilhete;
    system ("color 9f");
    printf ("\n\n>>> VIAJE BEM LTDA. <<<\n\n");
    printf ("\n>>> Lendo Dados <<<\n\n");
    printf ("Cod. da Empresa: \t\t");
    scanf ("%d", &bilhete.cod_e);
    fflush(stdin);
    printf ("Nome do Passageiro: \t\t");
    gets (bilhete.nome_p);
    printf ("Origem: \t\t\t");
    gets (bilhete.origem);
    printf ("Destino: \t\t\t");
    gets (bilhete.destino);
    printf ("Data do Bilhete (dd mm aa): \t");
    scanf ("%d%d%d", &bilhete.data_b.dia, &bilhete.data_b.mes, &bilhete.data_b.ano);
    printf ("Horario do Bilhete (hh mm): \t");
    scanf ("%d%d", &bilhete.horario_b.hora, &bilhete.horario_b.min);
    printf ("Assento: \t\t\t");
    scanf ("%d", &bilhete.assento);
    printf ("Valor do Bilhete: \t\t");
    scanf ("%f", &bilhete.valor);
    printf ("\n>>> Imprimindo Dados <<<\n\n");
    system ("pause"); // continua ...
}
```

```

system ("cls");
printf ("\n-----\n");
printf (" VIACAO VIAJEBEM \n");
printf (" \n");
printf (" Cod. Empresa: %4d Nome do Passageiro: %s \n", bilhete.cod_e,
        bilhete.nome_p);
printf (" Origem: %s Destino: %s \n", bilhete.origem,
        bilhete.destino);
printf (" Data: %2d/%2d/%2d Horario: %2d:%2d \n",
        bilhete.data_b.dia, bilhete.data_b.mes, bilhete.data_b.ano, bilhete.horario_b.hora,
        bilhete.horario_b.min);
printf (" Assento: %2d Valor do Bilhete: %5.2f \n", bilhete.assento,
        bilhete.valor);
printf ("-----\n\n");
system ("pause");
} // fim de main

```