ESTRUTURAS

As estruturas em C permitem colocar, numa única entidade, elementos de tipos diferentes (corresponde aos *registos* na generalidade das outras linguagens).

Uma **estrutura** é um conjunto de uma ou mais variáveis (chamadas de ***campos*** ou *membros*) agrupadas sobre um único nome, facilitando a sua referência.

Estes elementos ou campos podem ser de qualquer tipo de dados válidos em C (básicos (char, int, float ou double), vetores, strings, apontadores ou mesmo outras estruturas).

DECLARAÇÃO DE ESTRUTURAS:

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

A declaração de uma estrutura corresponde unicamente à definição de um novo tipo e não à declaração de variáveis do tipo estrutura.

Exemplo: Definição de uma estrutura que suporta datas.

```
struct  DATA
{
   int dia, ano;
   char Mes[12];
} ;
```

A definição da estrutura DATA indica que o compilador passa a conhecer um outro tipo, chamado struct DATA, que é composto por 2 inteiros (dia e ano) e um vetor com 12 carateres (Mes).

Declaração de variáveis do tipo estrutura fora da definição da estrutura

Basta indicar o tipo seguido do nome das variáveis:

Sintaxe:

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

Exemplo:

```
struct Data  d, datas[100], *ptr_data ;
```

Em que:

- **d** é uma variável do tipo struct DATA;
- **datas** é um vetor de 100 elementos, sendo cada um deles uma estrutura do tipo struct DATA;
- **ptr_data** é um apontador para o tipo struct DATA (a lecionar posteriormente).

Declaração de variáveis do tipo estrutura, aquando da declaração da própria estrutura

Sintaxe:

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

Exemplo:

```
struct  DATA
{
   int dia, ano;
   char Mes[12];
} d, datas[100], *ptr_data;
```

ACESSO AOS CAMPOS DE UMA ESTRUTURA:

Para aceder ao campo XXX de uma estrutura YYY, cuja variável do tipo YYY é ZZZ, usa-se o operador (.) ponto, fazendo:

```
ZZZ . XXX ;      //  ou seja nome_da_variavel_estrutura . nome_do_campo ;
```

Exemplo 1: Sabendo-se que a variável, tipo estrutura, Ficha, em um dado instante, contivesse os valores seguintes:

Nome: António Ajuizado

Endereco: Rua das Virtudes, s/n

Idade: 20

Salario: 1000

Ficha.Idade refere-se ao conteúdo do campo **idade** da estrutura de nome **Ficha**, isto é, **20**.

Exemplo 2 (com leitura e exibição de dados):

```
#include <stdio.h>
struct  DATA
{
    int dia, mes, ano;
}D;

main()
{
printf("Sistema de registo de datas\n");
 scanf("%d%d%d", &D.dia, &D.mes, &D.ano);
 printf("Data inserida:\n");
printf("DATA: %d/ %d/ %d\n", D.dia, D.mes, D.ano);
}
```

Exemplo 3 (com inicialização e exibição de dados):

```
#include <stdio.h>
#include <string.h>
struct      DATA
{
    int dia, ano;
    char mes[12];
} dt_nasc ;
```

```
main()
{
dt_nasc . dia = 23 ;
strcpy (dt_nasc . mes, " Janeiro" );
dt_nasc . ano = 1966 ;
printf(" Data Nascimento: %d / %s / %d \n", dt_nasc . dia, dt_nasc . mes, dt_nasc . ano );
}
```

Exemplo 4 (com leitura e exibição de dados):

```
#include <stdio.h>
struct  DATA
{
    int dia, mes, ano;
} datas[10];
main()
{ int i;
for(i=0; i<10; i++)
{
        printf("Data %d:\n", i+1);
        scanf("%d%d%d", &datas[i].dia, &datas[i].mes, &datas[i].ano);
}
printf("\nDatas inseridas:\n");
for (i=0; i<10;i++)
        printf("DATA %d : %d/ %d%/ %d\n", i+1, datas[i].dia, datas[i].mes, datas[i].ano);
}
```

Inicialização automática de estruturas:

Sintaxe: **struct** [nome_estrutura] [nome_varável] **=** **{**valor1, …, valorN **} ;**

Colocamos entre chavetas os valores a colocar nos campos da estrutura, pela ordem em que estes foram escritos na definição.

Exemplo 1:

```
/* Declaração da estrutura */
struct  Data
{
   int  dia, ano;
   char mes[12] ;
} dt = {23, 1985, "JAN"};   //Decl. da variável
…
 printf("Data:%d/%s/%d\n",dt.dia,dt.mes,dt.ano);
```

ou

```
/* Declaração da estrutura */
struct  Data
{
   int  dia, ano;
   char mes[12] ;
};
…
//Decl. da variável
struct Data dt = {23, 1985, "JAN"};
 printf("Data:%d/%s/%d\n",dt.dia,dt.mes, dt.ano);
```

Se a variável a inicializar for um vetor, a inicialização faz-se do mesmo modo colocando cada um dos elementos entre chavetas.

Exemplo 2:

```
struct  Data { int dia, ano; char mes[12] } ;
struct Data v [ ] = { {1, 1900, "JAN"}, {2, 1920, "FEV"}, {31, 1950, "DEZ"}} ;
```

OU

```
struct  Data { int dia, ano; char mes[12] } ;
struct Data v [ 3 ] = { {1, 1900, "JAN"}, {2, 1920, "FEV"}, {31, 1950, "DEZ"} ;
```

A definição de uma estrutura pode ser realizada sem indicar qual o seu nome, no entanto todas as variáveis têm que ser inicializadas no momento da sua definição.

Exemplo: struct { int dia, ano; char mes[12] } dt ;

Estas estruturas (sem nome) não podem ser enviadas ou recebidas como parâmetros de funções, pois não possuem um nome que as identifique na definição de parâmetros da função.

EXERCÍCIO RESOLVIDO

R1. Considerando o Registo de uma mercadoria de uma loja contendo as seguintes informações: código, nome, preço e stock, fazer um programa que, dado o Registo de 50 mercadorias, leia um código exiba o nome, preço e o stock da respectiva mercadoria.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define N 50

struct  MERC
{
    char COD[6];
    char NOME [15];
    float PRECO;
    int STOCK;
} TAB[N];

main()
{
     int i;
     char k[6], RESP;
    // Leitura dos dados
    for(i=0; i<N; i++)
    {
        printf("Código: ");  gets(TAB[i].COD);
        fflush(stdin);
        printf("Nome: ");  gets(TAB[i].NOME);
        printf("Preço: ");  scanf("%f", &TAB[i].PRECO);
        printf("Stock: ");  scanf(" %d", &TAB[i].STOCK);
        fflush(stdin);
    }
    do
    {
      fflush(stdin);

       //leitura da chave de pesquisa
      printf("entre com o código desejado: ");
      gets(k); fflush(stdin);

    //testa em cada Registo se o código é igual a chave pesquisada
       for(i=0; i<N; i++)
            if (strcmp(k,TAB[i].COD)==0)
                printf("REGISTO DESEJADO: %s, %s, %.1f, %d \n",TAB[i].COD,TAB[i].NOME,
    TAB[i].PRECO, TAB[i].STOCK);

     //verifica se o utilizador deseja pesquisar outro código
        printf("Repetir(S/N)?");
        RESP=getchar();
    }while (toupper(RESP) != 'N');
}
```

EXERCÍCIOS

1. Escreve, na linguagem C, as declarações para novas estruturas (PESSOA E ALUNO) declarando para cada estrutura um variável desse mesmo tipo, com os seguintes campos:

   a) NOME, ENDERECO, ESTADOCIVIL, GENERO;

   b) MATRÍCULA, NOTA1, NOTA2, NOTA3, MEDIA.

2. Escreve a instrução para declarar a variável AL do tipo ALUNO (alínea 1.b) fora da definição da estrutura ALUNO.

3. Escreve um programa que armazene os dados dos alunos da tua turma: nome, número e notas (Port, Mat e PSI) num registo e que permita, dado o número do aluno, imprimir o seu nome, notas e média.

4. Uma indústria faz a folha mensal de pagamentos de seus 10 empregados baseada no seguinte:

⇒ Existe uma tabela com os dados de cada funcionário (matrícula, nome e salário bruto);

⇒ Escreve um programa que leia e processe a tabela e emita, para cada funcionário, seu salário, cujo formato é dado a seguir:

```
Matrícula:
    Nome:
    Salário Bruto:
    Dedução SS:
    Salário Liquido:
```

⇒ O desconto da SS é de 12% do salário bruto.

⇒ O salário líquido é a diferença entre o salário bruto e a dedução da SS.

5. Rescreve o programa do exercício 4, de modo a que seja processada a tabela anterior, para todos os funcionários da empresa, mas apresentando os registos ordenados alfabeticamente por nome de funcionário.