Ficha de Trabalho N.º 5

Versão 2024/25

Objetivos: Utilização de Estruturas e de Arrays de Estruturas.

Conceitos Necessários à Resolução da Ficha

Introdução: declaração, inicialização, acesso aos campos

As estruturas em C permitem colocar, numa única entidade, elementos de diferentes tipos.

As componentes armazenadas dentro de uma estrutura são vulgarmente denominadas campos ou membros da estrutura

Declaração de estruturas

SINTAXE

struct [nome da estrutura]
{
    $tipo_1$ $campo_{11}$, $campo_{21}$, ... , $campo_{m1}$;
    ...
    $tipo_n$ $campo_{1n}$, $campo_{2n}$, ... , $campo_{kn}$;
};

A declaração de uma estrutura corresponde à declaração de um novo tipo e não à declaração de variáveis estruturadas

A declaração de variáveis pode também ser realizada quando se define a própria estrutura, especificando os identificadores depois da chaveta } que a fecha.

Declaração de estruturas

SINTAXE

struct [nome da estrutura]
{
    $tipo_1$ $campo_{11}$, $campo_{21}$, ... , $campo_{m1}$;
    ...
    $tipo_n$ $campo_{1n}$, $campo_{2n}$, ... , $campo_{kn}$;
} v_1, v_2 , ... , v_k;

Assim, a declaração anterior poderia tomar a forma:

```
struct data
{       int dia, ano;
        char mes[12];
}d, datas[50], *ptrData;
```

Mas é preferível separá-las!!

Exemplos

```
struct data
{       int dia, ano;
        char mes[10];
};

A partir desta declaração fica disponível um novo tipo (struct data),
composto por dois inteiros e um vector com 10 caracteres.
```

```
Na declaração de variáveis

struct data d, datas[50], *ptrData;

▪ d é uma variável do tipo struct data
▪ datas é um vector de 50 elementos, cada um deles do tipo struct data
▪ prtData é um apontador para o tipo struct data
```

Para aceder ao membro *mmm* de uma estrutura *eee* usa-se o operador ponto (.), fazendo *eee.mmm*

Exemplo

```
struct data{ int dia, ano; char mes[12];}dn;
dn.dia = 25;
dn.ano = 2009;
strcpy(dn.mes, "Novembro");
printf("\nDATA: %d-%s-%d\n", dn.ano, dn.mes, dn.dia);
```

É possível inicializar uma estrutura quando é declarada, usando a sintaxe:

struct estrutura variável = {$valor_1$,..., $valor_n$}

```
struct data{ int dia, ano; char mes[10];};
struct data dn = {25, 2009, "Novembro"};
printf("\nDATA: %d-%s-%d\n", dn.ano, dn.mes, dn.dia);
```

Declaração de tipos

Uma das desvantagens inerentes à utilização de estruturas consiste no facto de a declaração das variáveis ter de utilizar sempre a palavra reservada *struct* seguida do nome da estrutura.

A palavra reservada **typedef** permite que um determinado tipo possa ser denominado de modo diferente, de acordo com o interesse do utilizador.

TANTO PODE SER USADO COM ESTRUTURAS COMO COM UM TIPO QUALQUER DA LINGUAGEM.

SINTAXE

typedef tipo existente sinónimo;

Exemplo

```
typedef int inteiro;
typedef float real;
```

Exemplo

Considerando a estrutura data anteriormente definida,

```
typedef struct  pessoa
        {       char nome[60];
                struct data dataNascimento;
                int idade;
                char estadoCivil;
                float salario;
        }PESSOA; //PESSOA é um tipo estruturado
```

A partir desta declaração, podemos declarar variáveis com esta estrutura de forma muito simples:

```
PESSOA Tiago, Nuno, Filipe;
```

NOTA: na definição de um *typedef* NÃO podem ser declaradas variáveis (apenas tipos).

Problemas Propostos

1 - Calcular a área e o comprimento da diagonal de um retângulo definido através dois pontos, correspondentes aos vértices opostos da diagonal. Cada ponto é definido através de uma estrutura composta por duas coordenadas x e y.

2 - Defina o tipo de dados FRACAO (numerador/denominador) e implemente as operações de soma, subtração, divisão e multiplicação de frações. Os numeradores, denominadores e operação são pedidos ao utilizador.

3 - Pretendem-se registar as notas de um máximo de 50 estudantes. Para cada estudante é necessário registar o número, nome e a nota respetiva.

a) Elabore um programa que solicite ao utilizador N nomes, respetivos números e notas (sendo N <=50);
b) Acrescente ao programa anterior o código que permita, dado o número de um estudante, mostre o respetivo nome e nota;
c) Crie um novo programa ou um novo projeto (copie e altere o criado para responder às duas alíneas anteriores), mas, desta vez, estruturando-o, usando funções e use ponteiros para aceder aos elementos do vetor.

4 - Elabore um programa que permita armazenar em memória a informação relativa a um novo livro que chega a uma biblioteca. Considere que para cada livro deve ser guardada a seguinte informação:

- título (máximo 30 caracteres);
- autor (máximo 30 caracteres);
- área (máximo 20 caracteres);
- ano.

O programa deve permitir a inserção de novos livros, bem como listar e retirar livros da lista.

5 - Elaborar um programa que faça a gestão de reservas de uma sala de espectáculos, atribuindo automaticamente os respectivos lugares (dispostos numa matriz de F Filas por C Cadeiras). Cada reserva deve guardar o nome de quem a fez e se está ou não paga. A aplicação deve permitir as seguintes funcionalidades:

1 - Criar uma nova reserva num lugar vago;
2 - Eliminar uma reserva;
3 - Apresentar um mapa da ocupação da sala;
4 - Listar as reservas já realizadas apresentando o nome do cliente, o respectivo lugar e a indicação se está pago ou não.