# Tipos compostos

# Programação I 2020.2021

Teresa Gonçalves tcg@uevora.pt

Departamento de Informática, ECT-UÉ

# Como programar?



## Sumário

Tipo composto (ou registo)

Declaração, inicialização e utilização

Vetores e estruturas

# Tipo composto

# Tipo composto (ou registo)

### O que é?

Tipo estruturado, construido a partir de outros tipos Corresponde ao produto cartesiano de conjuntos

### Para que serve?

Para guardar informação relacionada entre si, de tipos diferentes

# Exemplo

### Registo de um livro numa biblioteca

Título → sequência de caracteres

Autores → sequência de caracteres

Editora → sequência de caracteres

Edição → número

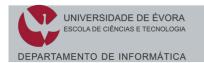
Reimpressão → número

Ano de publicação → número

ISBN → sequência de caracteres

Localização → sequência de caracteres

. . . .



# **Declaração e acesso**

### Declaração

```
struct <nome>{
    tipo membro1;
    tipo membro2;
    <...>
} <variaveis>;
```

### **Acesso aos elementos**

```
Utiliza-se o carácter '.'
```

```
variavel.membro1
```

variavel.membro2

# Declaração

#### Declaração do registo

```
struct data {
    int ano, mes, dia;
};
```

#### Declaração do registo e variável

```
struct data {
    int ano, mes, dia;
} hoje;
```

#### Declaração do registo e variável e inicialização

```
struct data {
    int ano, mes, dia;
} hoje = {2013, 3, 1};
```



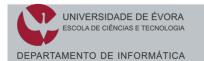
# Inicialização e utilização

### Declaração e inicialização

```
struct data hoje = {2013,1,4};
```

### Utilização

```
struct data ontem;
ontem.ano = 2012;
ontem.mes = 1;
ontem.dia = 3;
```



```
#include <stdio.h>
struct data {
    int ano;
    int mes;
    int dia;
};
main() {
    struct data hoje;
    hoje.ano = 2013;
    hoje.mes = 1;
    hoje.dia = 4;
    printf("Hoje e'%d/%d/%d.\n", hoje.ano, hoje.mes, hoje.dia );
```

# Atribuição, passagem de parâmetros, devolução de valores

Comportamento idêntico a qualquer outro tipo de dados pré-definido

```
struct abc {
   int a;
};
struct abc func(struct abc x) {
   struct abc r = x;
   return r;
}
int main(void){
   struct abc x = { 12 };
   struct abc y = func(x);
   printf("%d\n", y.a);
   return 0;
}
```



### Vetor numa estrutura

```
struct mes {
    int num_dias;
    int nome;
};
struct mes este_mes;
este_mes.numero_de_dias = 31;
este_mes.nome = 1;
printf("O mes %d tem %d dias\n", este_mes.nome,
este_mes.num_dias);
```

### Vetor de estruturas

```
struct data {
  int ano, mes, dia;
};
struct data feriados[12];
feriados[0].ano = 2013;
feriados[0].mes = 1;
feriados[0].dia = 1;
```

### **Estruturas em estruturas**

```
struct data {
    int ano, mes, dia;
};
struct hora{
    int horas, mins, segs;
};
struct data_hora{
    struct data sdata;
    struct hora shora;
};
struct data_hora hoje = { {2013, 1, 4 }, { 10, 40, 43 } };
hoje.shora.segs ++;
if( hoje.shora.segs == 60 ) {
    hoje.shora.mins ++;
    hoje.shora.segs = 0;
}
```

# Registo de funcionários numa empresa

```
struct data {
    int dia, mes, ano;
};
struct id {
    int ccidadao;
    int ncontribuinte;
};
struct funcionario {
    struct id ident;
    int idade;
    struct data entrada;
    float salario;
};
```

```
void escreve data(struct data d) {
    printf("%2d/%2d/%2d\n", d.dia, d.mes, d.ano);
}
void escreve_id(struct id identificador) {
    printf("ccidadao: %d; contribuinte: %d\n", identificador.ccidadao,
identicador.ncontribuinte);
}
void escreve_funcionario(struct funcionario f) {
    escreve_id(f.ident);
    printf("Idade: %2d\n", f.idade);
    printf("Data de entrada: "); escreve data(f.entrada);
    printf("Ordenado: %6.1f\n", f.salario);
}
main() {
    struct funcionario f = \{\{111223, 66623661\}, 35, \{10, 2, 2000\}, 1100\};
    escreve funcionario(f);
}
```