

# PROGRAMACIÓ 1

Grado en Ingeniería Informática e I2ADE

## Tema 7

### Tipos de datos estructurados: Registros



Dept. de Ciència de la Computació i Intel·ligència *a*rtificial  
Dpto. de Ciencia de la Computación e Inteligencia *a*rtificial



Universitat d'Alacant  
Universidad de Alicante

# Índice

2

1. Declaración de tipos de datos:  
**typedef**
2. El tipo *registro*
3. Arrays de registros

# 1. Declaración de tipos de datos: typedef

3

- En el lenguaje C se pueden definir tipos de datos personalizados utilizando la palabra reservada **typedef**.
- Es útil crear nuevos tipos de datos para mejorar la legibilidad de los programas.
- Sintaxis:

```
typedef declaración;
```

donde *declaración* tiene la forma de la declaración de una variable, con la diferencia de que la variable *creada* será en realidad un nuevo tipo de dato.

# 1. Declaración de tipos de datos: typedef

4

## Ejemplos de creación de nuevos tipos de datos mediante el uso de typedef

- Creación de un nuevo tipo de dato punto2D:

```
typedef int punto2D[2];
```

Ahora se pueden crear variables del nuevo tipo de dato **punto2D**:

```
punto2D punto;  
punto[0] = 3;  
punto[1] = -1;
```

Lo cual es equivalente a:

```
int punto[2];  
punto[0] = 3;  
punto[1] = -1;
```

# 1. Declaración de tipos de datos: typedef

5

## Ejemplos de creación de nuevos tipos de datos mediante el uso de typedef

```
// Definición de tipos de datos
typedef int    TVector[20];
typedef float TNotas[50];
typedef char   TCadena[30];
typedef int    TMatriz[3][3];

// Declaración de variables
TNotas  notas_P1, notas_P2;
TCadena nom_alumno1, nom_alumno2;
TMatriz matriz1, matriz2;
```

En la asignatura de Programación 1, por convención, nombraremos los nuevos tipos de datos con el prefijo T, para indicar que es un tipo de dato de usuario, y la siguiente (primera) letra del nombre del nuevo tipo en mayúscula

Lo que es equivalente a:

```
float notas_P1[50];
float notas_P2[50];
char  nom_alumno1[30];
char  nom_alumno2[30];
int   matriz1[3][3];
int   matriz2[3][3];
```

## 2. El tipo registro

6

- El **tipo registro** es una estructura de datos en la que se almacena una colección finita de elementos, no necesariamente del mismo tipo de dato, que están relacionados entre sí.
- Por lo general, en un registro se agrupan atributos (propiedades) de una entidad como, por ejemplo, una persona, un vehículo, etc.
- Cada uno de los elementos (atributos) de un registro se denomina **campo**.

## 2. El tipo registro

7

### Ejemplos de registros:

Dirección	
calle	array de char
código postal	array de char
ciudad	array de char

Libro	
autor	array de char
título	array de char
prestado	booleano

Fecha	
día	entero
mes	entero
año	entero

Empleado	
nombre	array de char
nº seguridad social	array de char
suelo	float
dirección	Registro
fecha nacimiento	Registro

**Nota:** Es conveniente identificar bien los registros y sus atributos (campos) antes de definirlos en el lenguaje de programación.

## 2. El tipo registro

8

### Declaración de un registro en C

- Para definir un nuevo tipo de dato *registro* se debe utilizar la palabra reservada **struct** acompañada del identificador (nombre) que se le asigna y del conjunto de atributos (campos) del registro delimitados por llaves (**{}**):

```
struct [nombre]{  
    tipo_campo1 nombre_campo1;  
    tipo_campo2 nombre_campo2;  
    ...  
}[var1, var2, ...]; // el punto y coma es obligatorio
```

**Nota:** Los corchetes indican optatividad

Los campos del registro se definen con la misma sintaxis que las variables

- **nombre**: nombre del tipo registro definido. Puede ser cualquier identificador válido. Se puede obviar si sólo se define la estructura para declarar variables (*var1*, *var2*, ...) de ese tipo.
- **tipo\_campoX**: tipo de cada uno de los campos del registro.
- **nombre\_campoX**: nombre de cada campo del registro. Puede haber tantos campos como sea necesario.
- *var1*, *var2*, ...: Se pueden declarar variables.



## 2. El tipo registro

9

### Ejemplos de declaración de registros:

```
struct TProducto {  
    int    codigo;  
    float  precio;  
} p1, p2;
```

p1 y p2 son variables del  
tipo de dato (registro)  
**struct TProducto**

```
struct TControl{  
    int cod;  
    bool testigo[10];  
};  
...  
struct TControl c1;
```

c1 es una variable del tipo de dato (registro) **struct TControl**.  
Este registro tiene un array de booleanos como campo.

Como no se ha declarado la estructura como tipo  
de dato, para declarar variables de este tipo de  
registro hay que utilizar **struct TControl**

```
typedef struct {  
    int numero;  
    char letra;  
} TDni;
```

```
typedef struct {  
    TDni nif;  
    char nombre[30];  
} TSocio;  
...  
TSocio socio1;
```

Utilizando **typedef** se declara un  
nuevo tipo de dato **TDni** que,  
posteriormente, podemos utilizar para  
declarar variables de ese tipo, incluso  
dentro de otros registros (anidación  
de registros)

## 2. El tipo registro

10

### Acceso a los campos de un registro en C

- Para acceder a los campos de un registro se debe utilizar el operador punto “.”.
- Sintaxis:

*identificador\_registro.nombre\_campo*

#### Ejemplo:

```
typedef struct {  
    int numero;  
    char letra;  
} TDni;  
  
...  
TDni miDni;
```

```
// Inicialización de la variable miDni  
miDni.numero = 12345678;  
miDni.letra = 'A';
```

identificador  
del registro

operador “.” de acceso a  
los campos del registro

identificador del campo del  
registro al que se quiere acceder

## 2. El tipo registro

11

### Asignación de registros en C

- El operador de asignación “=” funciona de forma similar a como lo hace entre tipos de datos simples.

#### Ejemplo:

```
#include<stdio.h>
#include<string.h> // para poder utilizar la función strcpy()

#define MAX_CAD 30
```

```
typedef char TCadena[MAX_CAD];
```

```
typedef struct {
    int numero;
    char letra;
} TDni;
```

```
typedef struct {
    TCadena nombre;
    int edad;
    TDni dni;
} TPersona;
```

```
int main(){
    TPersona p1, p2;

    strcpy(p1.nombre, "Juan Pérez");
    p1.edad = 43;
    p1.dni.numero = 12345678; // acceso a datos en estructuras
    p1.dni.letra = 'A';      // (registros) anidadas

    p2 = p1;
    ...
    return 0;
}
```

Tras esta sentencia, ambas variables tienen la misma información en sus campos.

# 3. Arrays de registros

12

## Arrays de registros

- Se declaran como cualquier otro array utilizando como tipo base del array el tipo de dato registro.
- En cada posición del array se almacena una variable del tipo de dato del registro.
- Para acceder a la información de un registro concreto del array, se accede primero a la posición del array y luego se accede al campo concreto.

### Ejemplo:

```
typedef struct {  
    int codigo;  
    float precio;  
} TProducto;  
  
TProducto listaProductos[100];
```

Una vez declarado el tipo de dato del registro se declara el array de registros

Ejemplo en el que se accede al noveno registro del array para asignar valores a sus dos campos.

```
...  
listaProductos[8].codigo = 456;  
listaProductos[8].precio = 30.49;  
...
```

### 3. Arrays de registros

13

#### **Ejemplo 1:**

Definir las estructuras de datos necesarias para procesar la siguiente información:

- Una empresa de alquiler de vehículos desea gestionar la información acerca de los vehículos que tiene (no más de 200). Concretamente: matrícula, marca, modelo, fecha de compra y kms mensuales realizados para todo el año, con la finalidad de obtener los vehículos que realizan más kilómetros de media al año (podrá ser uno solo o muchos con la misma media).

### 3. Arrays de registros

14

#### Ejemplo 1 (y II): Posibles estructuras de datos

- Una empresa de alquiler de **vehículos** desea gestionar la información acerca de los vehículos que tiene (no más de 200). Concretamente: matrícula, marca, modelo, fecha de compra y kms mensuales realizados para todo el año, con la finalidad de obtener los vehículos que realizan más kilómetros de media al año (podrá ser uno solo o muchos con la misma media).

registro: **vehículo**

Se necesita un array  
con 200 registros del  
tipo vehículo

Campo	Tipo de dato
matrícula	cadena de caracteres
marca	cadena de caracteres
modelo	cadena de caracteres
fecha de compra	registro: día, mes, año
kms/mes x 12 meses	array de 12 enteros

### 3. Arrays de registros

15

#### **Ejemplo 1 (y III):** Posibles estructuras de datos

- Una empresa de alquiler de vehículos desea gestionar la información acerca de los vehículos que tiene (no más de 200). Concretamente: matrícula, marca, modelo, fecha de compra y km mensuales realizados para todo el año, con la finalidad de obtener los vehículos que realizan más kilómetros de media al año (podrá ser uno solo o muchos con la misma media)

**Objetivo:** Obtener un array con los índices de los vehículos con media más alta (como mucho serán todos, es decir 200).

# 3. Arrays de registros

16

## Ejemplo 1 (y IV): Diseño de los datos

```
#define NUM_COCHES 200

typedef char TCadena[30];
typedef char TMatricula[9];
typedef int TKms_mes[12];
```

```
typedef struct {
    int dia;
    int mes;
    int anyo;
} TFecha;
```

```
typedef struct {
    TMatricula matricula;
    TCadena      marca;
    TCadena      modelo;
    TFecha       fech_comp;
    TKms_mes     kms_mes;
} TVehiculo;
```

```
typedef TVehiculo TListaVehiculos[NUM_COCHES];
TListaVehiculos vehiculos;
```

```
typedef int TVehiculosMasKms[NUM_COCHES];

// array de posiciones en el array vehiculos
TVehiculosMasKms vehiculos_mas_kms;

// número de vehículos con la media más alta
int num_vehiculos_mas_kms;
```

Estructura del registro para fecha

Estructura del registro para vehículo

Estructuras para guardar la lista de todos los índices de los coches con la media de kms al año más alta

Estructuras para guardar la lista de vehículos



# 3. Arrays de registros

17

## Ejemplo 2:

Definir las estructuras de datos necesarias para procesar la siguiente información:

- En una planta de fabricación de lavadoras quieren establecer un control de calidad informatizado de sus prototipos. Cada electrodoméstico viene caracterizado por un código numérico y una serie de características: capacidad (en kilos), modelo, tipo de carga (superior/frontal) y el resultado de los 10 controles a los que ha sido sometido. El control sólo tiene dos posibilidades: se ha pasado o no. Además, hay que saber qué revisor ha efectuado cada control. Un revisor puede realizar varios controles sobre el mismo aparato. De cada revisor se tiene la siguiente información: código numérico, nombre y departamento al que pertenece. La planta fabrica 25 prototipos al año.

### 3. Arrays de registros

18

#### Ejemplo 2 (y II): Posibles estructuras de datos

- En una planta de fabricación de **lavadoras** quieren establecer un control de calidad informatizado de sus prototipos. Cada electrodoméstico viene caracterizado por un código numérico y una serie de características: capacidad (en kilos), modelo, tipo de carga (superior/frontal) y el resultado de los 10 **controles** a los que ha sido sometido. El control sólo tiene dos posibilidades: se ha pasado o no. Además, hay que saber qué **revisor** ha efectuado cada control. Un revisor puede realizar varios controles sobre el mismo aparato. De cada revisor se tiene la siguiente información: código numérico, nombre y departamento al que pertenece. La planta fabrica 25 prototipos al año.

registro: **lavadora**

Se necesita un array  
con 25 registros del  
tipo lavadora

Campo		Tipo de dato	
código		entero	
capacidad		entero	
modelo		cadena	
carga		carácter o enumerado	
controles	ok	booleano	
	revisor	código	entero
		nombre	cadena
		departamento	cadena

# 3. Arrays de registros

19

## Ejemplo 2 (y III): Diseño de los datos

```
#define NUM_LAVADORAS 25
#define NUM_CONTROLES 10

typedef char TCadena[30];
typedef struct {
    bool ok;
    int  codRevisor;
} TControl;

typedef struct {
    int      codigo;
    TCadena  nombre;
    TCadena  departamento;
} TRevisor;
```

Estructura del  
registro para control

Estructura del  
registro para revisor

Estructura del  
registro para  
lavadora

Estructuras para  
guardar la lista de  
lavadoras

```
typedef struct {
    int      codigo;
    int      capacidad;
    TCadena  modelo;
    char     carga; // s - superior; f - frontal
    TControl controles[NUM_CONTROLES];
} TLavadora;

typedef TLavadora TListaLavadoras[NUM_LAVADORAS];
TListaLavadoras listaLavadoras;
```