

Programación 1

Tema 11

Registros



Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza





Índice

- Registros y campos
 - Sintaxis
 - Ejemplos
 - Dominio de valores
 - Representación externa
 - Operaciones
- Problemas y ejemplos

Problema

- Gestionar información relativa a una persona:
 - Nombre
 - Apellidos
 - NIF
 - Fecha de nacimiento
 - Estado civil (casado, no casado)

Registro

- **Registro o tupla**
 - Agrupan datos de igual o diferente naturaleza relacionados entre sí.
- Para utilizarlos en C++, hay que definir antes un **tipo registro**.

Ejemplos

Definición de un nuevo tipo
de datos: Fecha

```
struct Fecha {  
    unsigned dia, mes, agno;  
};
```

Tipo

Nombre de
la variable

```
Fecha hoy;
```

Declaración de una
variable de tipo Fecha

```
Fecha cumpleClase[80];
```

Tipo base

Nombre de
la variable de
tipo vector

Dimensión

Declaración de un vector
de 80 componentes
de tipo Fecha



Representación gráfica

Tipo Fecha

dia	<input type="text"/>
mes	<input type="text"/>
agno	<input type="text"/>

Variable hoy

dia	6
mes	11
agno	2024

Vector
cumpleClase

<table><tr><td>dia</td><td>1</td></tr><tr><td>mes</td><td>10</td></tr><tr><td>agno</td><td>2006</td></tr></table>	dia	1	mes	10	agno	2006	<table><tr><td>dia</td><td>21</td></tr><tr><td>mes</td><td>5</td></tr><tr><td>agno</td><td>2006</td></tr></table>	dia	21	mes	5	agno	2006	<table><tr><td>dia</td><td>23</td></tr><tr><td>mes</td><td>1</td></tr><tr><td>agno</td><td>2006</td></tr></table>	dia	23	mes	1	agno	2006	<table><tr><td>dia</td><td>26</td></tr><tr><td>mes</td><td>10</td></tr><tr><td>agno</td><td>2005</td></tr></table>	dia	26	mes	10	agno	2005	<table><tr><td>dia</td><td>18</td></tr><tr><td>mes</td><td>5</td></tr><tr><td>agno</td><td>2006</td></tr></table>	dia	18	mes	5	agno	2006	<table><tr><td>dia</td><td>11</td></tr><tr><td>mes</td><td>6</td></tr><tr><td>agno</td><td>2006</td></tr></table>	dia	11	mes	6	agno	2006	<table><tr><td>dia</td><td>20</td></tr><tr><td>mes</td><td>9</td></tr><tr><td>agno</td><td>2006</td></tr></table>	dia	20	mes	9	agno	2006
dia	1																																															
mes	10																																															
agno	2006																																															
dia	21																																															
mes	5																																															
agno	2006																																															
dia	23																																															
mes	1																																															
agno	2006																																															
dia	26																																															
mes	10																																															
agno	2005																																															
dia	18																																															
mes	5																																															
agno	2006																																															
dia	11																																															
mes	6																																															
agno	2006																																															
dia	20																																															
mes	9																																															
agno	2006																																															
0	1	2	3	4	5	...																																										

Ejemplos

```
struct Nif {  
    unsigned dni;    // número del DNI  
    char letra;      // letra asociada al DNI  
};
```

Tipo Nif

Sintaxis

```
<definiciónTipoRegistro>
    ::= “struct” <identificador> “{”
        { <definiciónCampo> }
        “}” “;”
<definiciónCampo>
    ::= <tipo> <declaraciónSimple>
        { “,” <declaraciónSimple> } “;”
<declaraciónSimple>
    ::= <nombreCampo> [ “=” <expresión> ]
```


Registros

- Dominio de valores
 - Producto cartesiano de los dominios de valores de los campos
- Representación externa
 - Listas
 - **Fecha** hoy = {30, 10, 2023};
 - **Nif** rey = {15, 'S'};

Registros

- Operadores:
 - ".": operador de selección de campo
 - hoy.dia = 27;
 - hoy.dia++;
 - cout << rey.dni << "-" << rey.lettra;
 - cumpleClase[0].mes = 8;
 - Asignación
- No hay operaciones disponibles de:
 - Comparación
 - Lectura de teclado o escritura en la pantalla



Metodología de trabajo con registros en módulos

□ Fichero de **interfaz** del módulo

- Definición del tipo registro
- Declaraciones de funciones adicionales para trabajar con el tipo

□ Fichero de **implementación** del módulo

- Definiciones de las funciones declaradas en el fichero de interfaz
- Definiciones de otros elementos auxiliares

nif.hpp

```
/*  
 * Definición del tipo de dato Nif que representa la  
 * información del NIF (Número de Identificación  
 * Fiscal) de una persona.  
 */  
struct Nif {  
    unsigned dni;           // número del DNI  
    char letra;            // letra asociada al DNI  
};  
  
...
```

nif.hpp

```
...  
  
/*  
 * Pre: ---  
 * Post: Devuelve «true» si y solo si  
 * «nifAValidar» define un NIF válido,  
 * es decir, su letra es la que le corresponde  
 * a su DNI.  
 */  
bool esValido(const Nif unNifAValidar);  
  
...
```

nif.hpp

```
...  
  
/*  
 * Pre: El valor del parámetro «nifAEscribir»  
 *       representa un NIF válido.  
 * Post: Escribe «nifAEscribir» en la pantalla,  
 *       con un formato como «01234567-L».  
 */  
void mostrar(const Nif nifAEscribir);  
  
...
```

nif.hpp

```
...  
  
/*  
 * Pre: ---  
 * Post: Devuelve la letra del número  
 * de identificación fiscal que  
 * corresponde a un número de  
 * documento nacional de identidad  
 * igual a «dni».  
 */  
char letra(const unsigned dni);
```

nif.cpp

```
#include "nif.hpp"
#include <cctype>
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

const unsigned NUM_LETRAS = 23;
const string TABLA_NIF
    = "TRWAGMYFPDXBNJZSQVHLCKE";
```


nif.cpp

```
/*  
 * Pre: ---  
 * Post: Devuelve la letra del número  
 * de identificación fiscal que  
 * corresponde a un número de  
 * documento nacional de identidad  
 * igual a «dni».  
 */  
char letra(const unsigned dni) {  
    return TABLA_NIF.at(dni % NUM_LETRAS);  
}
```

nif.cpp

```
/*  
 * Pre: ---  
 * Post: Devuelve «true» si y solo si  
 * «nifAValidar» define un NIF  
 * válido, es decir, su letra es la  
 * que le corresponde a su DNI.  
 */  
bool esValido(const Nif nifAValidar) {  
    return letra(nifAValidar.dni)  
        == toupper(nifAValidar.letra);  
}
```

nif.cpp

```
/*  
 * Pre: El valor del parámetro «nifAEscribir» representa  
 * un NIF válido.  
 * Post: Escribe «nifAEscribir» en pantalla, con un  
 * formato como «01234567-L». También modifica  
 * el carácter de relleno que utiliza el manipulador  
 * «setw», estableciendo el espacio en blanco.  
 */  
void mostrar(const Nif nifAEscribir) {  
    cout << setfill('0');  
    cout << setw(8) << nifAEscribir.dni << "-"  
        << nifAEscribir.letra;  
    cout << setfill(' ');  
}
```



Número de identificación fiscal.

Ejemplo de uso

```
Nif n;  
n.dni = 1234567;  
n.letra = 'L';  
if (esValido(n)) {  
    mostrar(n);  
    cout << endl;  
}
```



Número de identificación fiscal.

Ejemplo de uso

```
Nif n = {1234567, 'L'};  
if (esValido(n)) {  
    mostrar(n);  
    cout << endl;  
}
```

Personas

- Gestionar información relativa a personas:
 - Nombre
 - Apellidos
 - NIF
 - Fecha de nacimiento
 - Estado civil (casado, no casado)

persona.hpp

```
#include <string>
#include "nif.hpp"
#include "fecha.hpp"
using namespace std;

/*
 * Definición del tipo de dato Persona que representa información
 * de una persona: nombre y apellidos, número de identificación fiscal, fecha
 * de nacimiento, estado civil y sexo
 */
struct Persona {
    string nombre, apellidos;
    Nif nif;
    Fecha nacimiento;
    bool estaCasada;
};
```

Representación gráfica

Tipo Persona

nombre	<input type="text"/>						
apellidos	<input type="text"/>						
nif	<table><tr><td>dni</td><td><input type="text"/></td></tr><tr><td>letra</td><td><input type="text"/></td></tr></table>	dni	<input type="text"/>	letra	<input type="text"/>		
dni	<input type="text"/>						
letra	<input type="text"/>						
nacimiento	<table><tr><td>dia</td><td><input type="text"/></td></tr><tr><td>mes</td><td><input type="text"/></td></tr><tr><td>agno</td><td><input type="text"/></td></tr></table>	dia	<input type="text"/>	mes	<input type="text"/>	agno	<input type="text"/>
dia	<input type="text"/>						
mes	<input type="text"/>						
agno	<input type="text"/>						
estaCasada	<input type="checkbox"/>						

persona.hpp

```
/*  
 * Pre: ---  
 * Post: Devuelve una cadena que  
 * representa el nombre completo de  
 * la persona «p».  
 */  
string nombreCompleto(const Persona p);  
  
/*  
 * Pre: ---  
 * Post: Muestra los datos de la  
 * persona «p» en la pantalla.  
 */  
void mostrar(const Persona p);
```

persona.hpp

```
/*  
 * Pre: ---  
 * Post: Devuelve «true» si y  
 * solo si la fecha de  
 * nacimiento de «persona1»  
 * es estrictamente anterior a  
 * la fecha de nacimiento de  
 * «persona2».  
 */  
bool esMayorQue(  
    const Persona persona1,  
    const Persona persona2);
```

persona.cpp

```
#include <iostream>
#include "persona.hpp"
using namespace std;

/*
 * Pre: ---
 * Post: Devuelve una cadena que representa el
 *       nombre completo de la persona «p».
 */
string nombreCompleto(const Persona p) {
    return p.nombre + " " + p.apellidos;
}
```

persona.cpp

```
/*  
 * Pre: ---  
 * Post: Muestra los datos de la persona «p» en la  
 * pantalla.  
 */  
void mostrar(const Persona p) {  
    cout << "Persona: " << nombreCompleto(p) << endl;  
    cout << "NIF: "; mostrar(p.nif); cout << endl;  
    cout << "Nacida/o el ";  
    mostrar(p.nacimiento); cout << endl;  
    if (p.estaCasada) {  
        cout << "Casada/o" << endl;  
    } else {  
        cout << "Soltera/o" << endl;  
    }  
}
```

persona.cpp

```
/*  
 * Pre: ---  
 * Post: Devuelve «true» si y solo si la  
 *       fecha de nacimiento de «persona1»  
 *       es estrictamente anterior a la fecha de  
 *       nacimiento de «persona2».  
 */  
bool esMayorQue(const Persona persona1,  
                const Persona persona2) {  
    return esAnterior(persona1.nacimiento,  
                      persona2.nacimiento);  
}
```

Persona. Ejemplos de utilización

```
Persona rey;  
rey.nombre = "Felipe";  
rey.apellidos = "Borbón Grecia";  
rey.nif.dni = 15;  
rey.nif.letra = 'S';  
rey.nacimiento.dia = 30;  
rey.nacimiento.mes = 1;  
rey.nacimiento.agno = 1968;  
rey.estaCasada = true;  
mostrar(rey);  
cout << endl;
```

Persona. Ejemplos de utilización

```
Persona reinaEmerita
    = { "Sofía", "Grecia Dinamarca",
        {11, 'B'}, {2, 11, 1938},
        true
    };
mostrar(reinaEmerita);
cout << endl;
```

Persona. Ejemplos de utilización

```
Persona reinaEmerita
    = { "Sofía",           // nombre
        "Grecia Dinamarca", // apellidos
        {11, 'B'},         // NIF
        {2, 11, 1938},     // fecha nacimiento
        true               // estaCasada
    };
mostrar(reinaEmerita);
cout << endl;
```


Persona. Ejemplos de utilización

```
Persona princesa = {"Leonor",  
                    "Borbón Ortiz"};  
princesa.nif = {16, 'Q'};  
princesa.nacimiento = {31, 10, 2005};  
princesa.estaCasada = false;  
mostrar(princesa);  
cout << endl;
```



Persona. Ejemplos de utilización

Representación gráfica

nombre	"Felipe"
apellidos	"Borbón Grecia"
nif	dni 15 letra 'S'
nacimiento	dia 30 mes 1 agno 1968
estaCasada	true

nombre	"Sofía"
apellidos	"Grecia Dinamarca"
nif	dni 11 letra 'B'
nacimiento	dia 2 mes 11 agno 1938
estaCasada	true

nombre	"Leonor"
apellidos	"Borbón Ortiz"
nif	dni 16 letra 'Q'
nacimiento	dia 31 mes 10 agno 2005
estaCasada	false

¿Cómo se puede estudiar este tema?

- Repasando estas transparencias
- Trabajando con el código de estas transparencias
 - <https://github.com/prog1-eina/tema-11-registros>
- Leyendo
 - «Data structures». *Cplusplus.com*. 2000–2017
 - **Excepto la sección «Pointers to structures»**
 - <http://www.cplusplus.com/doc/tutorial/structures/>
 - Capítulo 11 adaptado de los apuntes del profesor Martínez
- Resolviendo los problemas de la clases de problemas
- Programando las soluciones a los problemas de la práctica 5