Práctica 8. Tipos de datos estructurados. Registros

Objetivos:

- Conocer la importancia que tiene el tipo de dato registro (struct).
- Declarar, definir y saber utilizar el tipo struct.
- Diseñar e implementar tipos compuestos mediante anidación de arrays y registros.

1. Registros

El término registro es un concepto que se utiliza continuamente en la vida diaria. Bajo el punto de vista de la programación también es un concepto de gran utilidad.

Un **registro** es una estructura compuesta por un número fijo de componentes, llamados campos, donde cada campo viene definido por su tipo y su identificador (nombre).

Un registro guarda información relativa a un solo objeto. Así, por ejemplo, podemos considerar que la fecha tiene tres componentes, día, mes y año. Igualmente, la información relativa a una persona puede contener su nombre, apellidos, fecha de nacimiento, estado, teléfono, etc.

Es muy importante distinguir entre registro y array.

Los campos de un registro pueden tener diferentes tipos, mientras que los elementos de un array deben ser todos del mismo tipo.

Además, los campos de un registro se deben seleccionar por nombre, mientras que los elmentos de un array se seleccionan con subíndices.

En C se define una estructura o registro de la siguiente manera:

```
typedef struct nombre_struct{
      tipo_elemento nombre_campo1;
      tipo_elemento nombre_campo2;
      …
};
```

- **Para acceder a los campos del registro y poder operar con ellos, tendremos que utilizar el operador "." separando el nombre de la estructura y el nombre del campo.**

Por ejemplo, la fecha indicada arriba la podríamos definir como:

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

Y posteriormente definir la variable como:

```
TFecha fecha;
```

2. Ejercicio resuelto 1. Repasa

- **Define las estructuras de datos necesarias para crear un programa donde se almacenen los datos de 76 alumnos. De cada alumno se almacenarán los siguientes datos:**
  - **Nombre (45 caracteres)**
  - **Apellidos (45 caracteres)**
  - **Dirección (45 caracteres)**
  - **Asignatura 1 (3 caracteres)**
  - **Nota Asignatura 1 (real)**
  - **Asignatura 2 (3 caracteres)**
  - **Nota Asignatura 2 (real)**
  - **….**
  - **Asignatura 10 (3 caracteres)**
  - **Nota Asignatura 10 (real)**

2.1 Solución

```
#include<iostream>
using namespace std;

//En primer lugar definimos las constantes necesarias.
//Constantes para indicar la longitud de las cadenas
const int kTAMCAD = 45;    //Para el nombre, apellidos y dirección
const int kTAMASIG = 3;   //Para el código de las asignaturas

//Constantes para el tamaño de los arrays
const int kNUMASIG = 10;   //Número de asignaturas de cada alumno
const int kNUMALU = 76;   //Número de alumnos

//Definimos las estructuras necesarias
//Cada alumno tiene 10 asignaturas compuesto de código de la asignatura y nota
typedef struct {
      char codigoAsignatura[kTAMASIG];
      float nota;
} TAsignatura;

//Estructura con la información de los alumnos
typedef struct {
      char nombre[kTAMCAD];
      char apellidos[kTAMCAD];
      char direccion[kTAMCAD];
      TAsignatura asig[kNUMASIG];
} TFichaAlumno;

//Por último queda definir el array que contendrá los datos de todos los alumnos
typedef TFichaAlumno TAlumnos [kNUMALU];
```

3. Ejercicio resuelto 2. Repasa

- **Realizar un programa en C que guarde información de 30 alumnos. De cada alumno leeremos su número de expediente, nombre, fecha de nacimiento, fecha de ingreso y su nota media. El programa debe permitir dar de alta un alumno y mostrar todos los alumnos.**

3.1 Solución

```
#include<iostream>
using namespace std;

//Definición de constantes
const int kTAMCAD = 45;  //Cadena de caracteres
const int kNUMALU =30;   //Número de alumnos

//Registro que define la fecha
typedef struct {
      int dia;
      int mes;
      int anyo;
} TFecha;
//Registro que define la información que se desea almacenar de los alumnos
typedef struct {
      int exp;
      char nombre[kTAMCAD];
      TFecha fechaNac;
      TFecha fechaIng;
      float notaMedia;
} TFichaAlumno;
//Tipo de datos que es un array de alumnos
typedef TFichaAlumno TAlumnos[kNUMALU];

//Definición cabecera funciones
void mostrarFecha(TFecha);
void mostrarAlumnos(TAlumnos, int);
void darDeAltaAlumno(TAlumnos, int &);
void pedirFecha(TFecha &);
```

```
//Programa principal
main() {
   TAlumnos alumnos;
   int opcion, numAlumnos;

   numAlumnos = 0;

   do{
    cout<<"******Gestión de alumnos******"<<endl;
    cout<<"Opciones disponibles: "<<endl;
    cout<<"1. Dar de alta un alumno"<<endl;
    cout<<"2. Mostrar listado alumnos"<<endl;
    cout<<"3. Salir"<<endl;
    cout<<"Seleccione una opción: ";
    cin>>opcion;

    switch(opcion){
      case 1: if(numAlumnos<30)
                 darDeAltaAlumno(alumnos, numAlumnos);
           else
              cout<<”No es posible introducir más alumnos”<<endl;
            cin.get();
            cin.get();
            break;
      case 2: mostrarAlumnos(alumnos, numAlumnos);
            cin.get();
            cin.get();
            break;
      case 3: cout<<"Terminado"<<endl;
            break;
      default: cout<<"Opción incorrecta"<<endl;
    };
  }while(opcion!=3);
}
//Muestra la fecha separada por /
void mostrarFecha(TFecha fecha) {
   cout<<fecha.dia<<"/"<<fecha.mes<<"/"<<fecha.anyo;
```

```
}

//Muestra los datos de los alumnos
void mostrarAlumnos(TAlumnos alumnos, int numAlumnos) {
    int i;

    for(i = 0; i<numAlumnos;i++){
       cout<<endl;
       cout<<"Alumno: "<<alumnos[i].exp<<endl;
       cout<<"Nombre: "<<alumnos[i].nombre<<endl;
       cout<<"Fecha de nacimiento: ";
       mostrarFecha(alumnos[i].fechaNac);
       cout<<endl;
       cout<<"Fecha de ingreso: ";
       mostrarFecha(alumnos[i].fechaIng);
       cout<<endl;
       cout<<"Nota media: "<<alumnos[i].notaMedia<<endl;
    }
}

//Solicita la fecha
void pedirFecha(TFecha &fecha) {
     cout<<"Introduce día:";
     cin>>fecha.dia;
     cout<<"Introduce mes: ";
     cin>>fecha.mes;
     cout<<"Introduce año: ";
     cin>>fecha.anyo;
}

//Solicita los datos de los alumnos
void darDeAltaAlumno(TAlumnos alumnos, int &numAlumnos) {
   int i;

   i = numAlumnos;

   cin.get();
```

```
  cout<<"Introduce el número de expediente: ";
  cin>>alumnos[i].exp;
  cin.get();
  cout<<"Introduce el nombre:";
  cin.getline(alumnos[i].nombre, kTAMCAD);
  cout<<"Introduce la fecha de nacimiento: "<<endl;
  pedirFecha(alumnos[i].fechaNac);
  cout<<"Introduce la fecha de ingreso: "<<endl;
  pedirFecha(alumnos[i].fechaIng);
  cout<<"Introduce la nota media: ";
  cin>>alumnos[i].notaMedia;

  numAlumnos++;
}
```

4. Ejercicio resuelto 3. Repasa

- **Crea un procedimiento que permita ordenar la estructura del ejercicio anterior por orden alfabético, utilizando las funciones de cadenas vistas en la práctica anterior.**

4.1 Solución

Para resolver este ejercicio vamos a utilizar el algoritmo de selección directa.

```
void ordenarAlumnosPorNombre(TAlumnos alumnos, int numAlumnos){
  int i, j, posicion_minimo;
  TFichaAlumno fichaAlumno;

  for(i=0; i<numAlumnos;i++) {
    posicion_minimo = i;
    for(j =i+1; j<numAlumnos; j++) {

if(strcmp(alumnos[j].nombre,alumnos[posicion_minimo].nombre) < 0)
                posicion_minimo = j;
    }
    fichaAlumno = alumnos[posicion_minimo];
```

```
        alumnos[posicion_minimo] = alumnos[i];
        alumnos[i] = fichaAlumno;
    }
}
```

5. Ejercicio resuelto 4. Repasa

▶ **Disponemos de los siguientes datos de vehículos:**

Matrícula	Marca	Propietario	Precio
A-1111-AA	Renault Megane	11111111-A	22.500,00€
2222 BBB	Ford Kuga	22222222-B	24.000,95€
C-3333-CC	Ferrari	33333333-C	99.999,99€
4444 DDD	BMW X5	44444444-D	34.000,00€
5555 EEE	Mercedes SLK	55555555-E	45.454,45€

Define en C una estructura adecuada para almacenar los datos anteriores y los módulos que nos permitan leer y escribir datos de la estructura creada.

5.1 Solución

```
#include<iostream>
using namespace std;

//Definición de constantes
const int kTAMMAT = 11;    //Cadena de caracteres para la matrícula y el NIF
const int kTAMMARCA = 40; //Cadena de caracteres para la marca
const int kNUMCOCHES =5;   //Número de vehículos

//Registro que define la fecha
typedef struct {
      int dia;
      int mes;
```

```
      int anyo;
} TFecha;

//Registro que define la información que se desea almacenar de
los vehículos
typedef struct {
      char matricula[kTAMMAT];
      char marca[kTAMMARCA];
      char nif[kTAMMAT];
      float precio;
} TFichaCoche;

//Tipo de datos que es un array de vehículos
typedef TFichaCoche TCoches[kNUMCOCHES];


//Definición cabecera funciones
void darDeAltaVehiculo(TCoches, int &);
void mostrarVehiculos(TCoches, int);

//Programa principal
main(){
   TCoches vehiculos;
   int opcion, numCoches;

   numCoches = 0;

   do{
     cout<<"******Gestión de vehículos******"<<endl;
     cout<<"Opciones disponibles: "<<endl;
     cout<<"1. Dar de alta un vehículo"<<endl;
     cout<<"2. Mostrar listado vehículos"<<endl;
     cout<<"3. Salir"<<endl;
     cout<<"Seleccione una opción: ";
     cin>>opcion;

     switch(opcion){
       case 1: if(numCoches<kNUMCOCHES)
```

```
            darDeAltaVehiculo(vehiculos, numCoches);
          else
            cout<<"No puedes introducir más coches, ya has
introducido el máximo"<<endl;
          cin.get();
          cin.get();
          break;
      case 2: mostrarVehiculos(vehiculos, numCoches);
          cin.get();
          cin.get();
          break;
      case 3: cout<<"Terminado"<<endl;
          break;
      default: cout<<"Opción incorrecta"<<endl;
     };
  }while(opcion!=3);
}

void darDeAltaVehiculo(TCoches vehiculos, int &coches) {
  int i;

  i = coches;

  cout<<"Tamaño del array: "<<i<<endl;

  cin.get();
  cout<<"Introduce la matricula: ";
  cin.getline(vehiculos[i].matricula,kTAMMAT);
  cout<<"Introduce la marca: ";
  cin.getline(vehiculos[i].marca, kTAMMARCA);
  cout<<"Introduce el nif del propietario: ";
  cin.getline(vehiculos[i].nif, kTAMMAT);
  cout<<"Introduce el precio: ";
  cin>>vehiculos[i].precio;

  coches ++;
}
```

```
void mostrarVehiculos(TCoches vehiculos, int coches) {
  int i;

   for(i = 0; i<coches;i++) {
      cout<<endl;
      cout<<"Datos del vehículo: "<<i+1<<endl;
      cout<<"Matricula: "<<vehiculos[i].matricula<<endl;
      cout<<"Marca: "<<vehiculos[i].marca<<endl;
      cout<<"Propietario: "<<vehiculos[i].nif<<endl;
      cout<<"Precio: "<<vehiculos[i].precio<<endl;
   }
}
```

6. Ejercicio propuesto 1. Resuelve

► **Completa el ejercicio anterior añadiendo dos opciones más en su menú. Debes incluir las opciones:**

- **Buscar un vehículo. Esta opción preguntará la matrícula del coche a buscar, buscará el coche en la estructura y mostrará sus datos por pantalla.**
- **Eliminar vehículo. Esta opción preguntará la matrícula del coche a eliminar y lo eliminará de la estructura.**

Ejemplo de menú:

****** Gestión de vehículos ******

Opciones disponibles:

1. Dar de alta un vehículo
2. Mostrar listado de vehículos
3. Buscar un vehículo
4. Eliminar un vehículo
5. Salir

7. Ejercicio propuesto 2. Resuelve

▶ La FIFA se está planteando gestionar la información de las selecciones y jugadores del próximo mundial de fútbol a celebrar en el 2014. De las selecciones le interesa disponer de la siguiente información:

- **Nombre del país**
- **Número de convocados**
- **Seleccionador**
- **Si ha sido campeona del mundo**

En el caso de los jugadores le interesa almacenar los siguientes datos:

- **Nombre**
- **Selección a la que pertenece**
- **Goles marcados**
- **Posición en el campo (portero, defensa, mediocampo o delantero)**

Teniendo en cuenta estos requerimientos, debes diseñar e implementar un programa que contenga un menú con las siguientes opciones:

1. **Leer desde teclado la información de una nueva selección.**
2. **Leer desde teclado la información de un nuevo jugador.**
3. **Mostrar un listado con los datos de todas las selecciones.**
4. **Mostrar un listado con los datos de todos los jugadores de una selección (se pedirá el nombre de la selección por teclado).**
5. **Mostrar el "pichichi" del mundial (aquel jugador (o jugadores) que hayan marcado más goles). El listado debe mostrar la siguiente información:**

Nombre jugador, Selección a la que pertenece y número de goles marcados

8. Ejercicio propuesto 3. Resuelve

▶ Un supermercado quiere gestionar la información acerca de sus productos y sus proveedores. De cada producto le interesa conservar los siguientes datos:

- **Código**
- **Nombre**
- **Proveedor**

- **Precio**
- **Stock actual**
- **Stock mínimo**
- **Stock máximo**

En el caso de los proveedores le interesa almacenar la siguiente información:

- **CIF**
- **Nombre**
- **Teléfono**
- **Dirección**
- **Sector (cárnico, frutas, verduras, percadería o droguería)**

Un proveedor podría suministrar diversos tipos de productos.

Teniendo en cuenta estos requerimientos, debes diseñar e implementar un programa que contenga un menú con las siguientes opciones:

1. **Dar de alta un nuevo producto. Se debe comprobar que el proveedor existe (ya está dado de alta).**
2. **Dar de alta un nuevo proveedor.**
3. **Mostrar un listado con los productos de un mismo proveedor (se debe pedir el nombre del proveedor).**
4. **Mostrar un listado con los datos de todos los proveedores.**
5. **Mostrar un listado de pedidos a realizar. Los productos que se deben pedir son aquellos en el que el stock actual es menor que el stock mínimo. Se deberá mostrar: código del producto, proveedor y cantidad a pedir. La cantidad a pedir será la suficiente para que el stock del producto sea igual al stock máximo.**

9. Ejercicio propuesto 4. Resuelve

Para poder gestionar los datos de sus empleados, una empresa nos ha pedido un programa que deberá almacenar para cada trabajador la siguiente información:

- **nombre**
- **teléfono,**
- **edad,**
- **número de hijos,**
- **fecha de ingreso en la empresa,**

- **Volumen de ventas de los últimos 4 meses (vector).**

Nuestro programa tendrá un menú con una serie de opciones:

1. **Introducir un nuevo empleado (tendremos que guardar cuantos empleados hemos introducido hasta ahora)**
2. **Listar los datos de todos los empleados cuya edad esté comprendida entre dos edades.**
3. **Obtener el empleado del mes, que será aquel cuyo volumen de ventas en los 4 últimos meses sea el mayor.**
4. **Ordenar los empleados por el volumen total de los 4 últimos meses.**
5. **Listar por pantalla las nóminas calculadas de todos los empleados. El listado debe tener el siguiente formato:**

Nombre	*Fijo*	*C. Trienios*	*IRPF*	*Nomina*
Pepito Grillo	900€	160€	15%	901€
Juanito González	900€	240€	11%	1014€
…	…	…	…	…

Para calcular la nómina hay que tener en cuenta la siguiente información:

- **Todos los empleados perciben un salario fijo de 900€.**
- **Además cobran una cantidad de 80€ por cada trienio (tres años de antigüedad en la empresa).**
- **A cada empleado se le aplica una retención sobre el salario que depende del nº de hijos que tiene. Se aplican los siguientes porcentajes:**

Nº hijos	**Retención**
0	**18%**
1	**15%**
2	**11%**
3 ó más	**10%**

- **El salario final se calcula como: (s. fijo + 80*trienios)*(1-IRPF).**
- **Para simplificar los cálculos, ten en cuenta simplemente el año de la fecha de alta en la empresa y el año actual con el fin de calcular el número de trienios.**

10. Ejercicio propuesto 5. Resuelve

▶ **Implementa un programa en lenguaje C que permita gestionar las pociones de un taller de magia. ¡Los magos están desesperados con tantas pociones! Debes ayudarlos para que no se equivoquen. Se desea almacenar para cada poción:**

- Número de la poción: se rellenará automáticamente empezando en 0.
- Utilidad de la poción: "amor", "exámenes", "salud", etc.
- Ingredientes que lleva: puede contener un número variable de ingredientes, como máximo 10 (pueden estar repetidos), los cuales se codificarán mediante un identificador entero, de la siguiente forma:

Identificador	Ingrediente
1	Bigote de gato
2	Escama de dragón
3	Diente de diablo
4	Veneno de serpiente

- Cantidad de la poción que hay en el frasco (medida en ml).
- Si se ha terminado la poción del frasco o no.

El programa tendrá un menú con las siguientes opciones:

1. **Introducir una nueva poción**. El almacén de los brujos es reducido, por lo que como máximo podrán almacenar 100 pociones. Características:
    a. El número de poción no se pedirá al usuario, se escribirá automáticamente (basándote en la posición del array), empezando en 0.
    b. Inicialmente una poción está como "no terminada", como es lógico.
2. **Utilizar poción**, sabiendo su número, basándonos en:
    a. Ten en cuenta que el número de poción está relacionado con la posición en el array, no es necesario que la busques por todo el array.
    b. Se pedirá la cantidad que se desea utilizar (siempre que sea menor que la cantidad que queda en el frasco) y se decrementará del frasco. Si se llegase a terminar la poción, marcar una poción como "terminada".
3. **Listar las pociones que todavía quedan** (las que no se han terminado).
4. **Finalizar la ejecución del programa.**

La ejecución del programa consistirá en ir seleccionando cualquiera de las 3 primeras opciones del menú, en cualquier orden y número de veces, hasta que se elija la opción 4, en cuyo caso el programa finalizará.

EJEMPLO DE EJECUCIÓN

```
... se elije la opción 1 del menú
Poción número 0
Utilidad: exámenes
Número de ingredientes: 2
Ingrediente 1: 4
Ingrediente 2: 6
```

Cantidad (ml): **500**

... se vuelve a elegir la opción 1
Poción número 1
Utilidad: **salud**
Número de ingredientes: 1
Ingrediente 1: **2**
Cantidad (ml): **200**

... se elije la opción 2 del menú
Número de poción: **0**
Cantidad a utilizar en ml(max 500): **300**

... se vuelve a elegir la opción 2
Número de poción: **0**
Cantidad a utilizar en ml(max 200): 2**00**
¡la poción 0 se ha terminado!

... se vuelve a elegir la opción 1
Poción número 2
Utilidad: **amor**
Número de ingredientes: **2**
Ingrediente 1: **1**
Ingrediente 2: **4**
Cantidad (ml): **250**

... se elije la opción 3 del menú (se listan sólo las pociones no terminadas)

****
Numero: 1
Utilidad: salud
Ingredientes: Escama de dragón
Cantidad: 200 ml
****
****
Numero: 2
Utilidad: amorIngredientes: Bigote de gato Veneno de serpiente
Cantidad: 250 ml
****

10. Errores de programación comunes

Confundir la definición de un tipo de dato y una variable. Cuando escribimos en un programa:

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

Estamos definiendo un tipo de datos, no una variable. Si quisiéramos definir una variable se haría como:

```
TFecha fecha;
```

En este caso fecha, sería una variable del tipo TFecha que se compone de tres valores.

- **Acceso erróneo a los valores de una variable de tipo struct. Cuando tenemos una variable de tipo struct accedemos a los datos con el operador ".". Por ejemplo, para la variable fecha anterior, accederemos a sus datos con: fecha.dia, fecha.mes y fecha.anyo.**