# Grado en Ingeniería Informática
# Introducción a la Programación- Curso 13-14

## Práctica 4 - Tipos de datos

**TIEMPO ASIGNADO: 8** Horas de laboratorio

**OBJETIVOS:**

- Que el alumno conozca y sea capaz de implementar y manejar en C el tipo de datos **array** (vectores y matrices).
- Que el alumno conozca y sea capaz de aplicar los conceptos de aritmética de punteros al usar **arrays**
- Que el alumno conozca y sea capaz de implementar y manejar en C cadenas de caracteres en C, conociendo además el uso de las principales funciones de la biblioteca **string.h**.
- Que el alumno conozca y sea capaz de implementar y manejar en C el tipo **struct** (registros o estructuras).
- Que el alumno conozca y sea capaz de implementar y manejar en C los tipos **union** y **enum** (uniones y tipos enumerados)
- Que el alumno sea capaz de definir y manejar en C sus propios tipos de datos usando la cláusula **typedef**.
- Que el alumno sea capaz de trabajar con ficheros en C (de texto y binarios) haciendo uso de las funciones básicas para manejarlos (abrir, leer, escribir, posicionamiento etc).
- Que el alumno conozca las funciones de manejo de memoria dinámica y sea capaz de definir vectores y matrices de forma dinámica.
- Que el alumno sea capaz de implementar usando el IDE Code::Blocks 12.11 programas en lenguaje C que hagan uso de los diferentes tipos de datos vistos

**EJERCICIOS PROPUESTOS**

**1.-** Realiza un programa que declare un vector de 10 enteros y a través de funciones, lo inicialice y lo visualice por pantalla.

**2.-** Escribe un programa que lea diez enteros comprendidos entre 1 y 10, los almacene en un vector y escriba por pantalla la cantidad de elementos de cada número que contiene.

Ejemplo: Dado el siguiente vector [2, 2, 2, 3, 3, 4, 4, 4, 7, 7]

La salida producida sería:

Existen 3 elementos del número 2
Existen 2 elementos del número 3
Existen 3 elementos del número 4
Existen 2 elementos del número 7

**3.-** Escribe un programa que normalice los 20 números reales que están almacenados en un vector *estadisticas*. Para llevar a cabo esta normalización, se debe dividir cada número por el máximo valor del vector de forma que los valores resultantes estén comprendidos entre 0 y 1. Realiza una versión devolviendo el resultado en el mismo vector y otra que construya un nuevo vector normalizado.

**4.-** Realiza un programa que escriba por pantalla el máximo de los elementos de un vector. Para ello se deben usar las funciones cuyos prototipos se describen a continuación y utilizar aritmética de punteros para acceder a los elementos del vector.

- *void leer_vector(int *vector)*
- *void maximo(int *vector, int *max)*

**5.-** Dada la siguiente definición:

*int vector[4]={1, 7, 3, 4};*

Indica que valor resulta de la evaluación de las siguientes expresiones:

1.- *vector + (* (vector + 1))
2.- *(vector + 3)

**6.-** Realiza un programa que, dado un vector con 15 números enteros distintos, identifique si los valores que el usuario introduzca a continuación, están o no contenidos en el vector indicando la posición del elemento en caso de existir. El programa finalizará cuando se introduzca un valor 0.

**7.-** Implementa un programa que realice el producto escalar de dos vectores.

**8.-** Realiza un programa que calcule la desviación típica de una serie de números reales almacenados en un vector.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i = \bar{x})^2}{N}}$$

**9.-** Realiza un programa que dada una matriz cuadrada de números reales, obtenga una segunda matriz traspuesta de la primera, escribiendo el resultado por pantalla.

**10.-** Realiza un programa que dadas dos matrices cuadradas de enteros, obtenga una matriz producto de las anteriores e imprima ese resultado por pantalla.

**11.-** Implementa una función que permita copiar todos los elementos de una matriz cuadrada de enteros en otra matriz.

**12.-** Realiza una función *int compara(int A[N][N], int B[N][N])* que determine si dos matrices cuadradas de enteros son iguales. La función devolverá 1 si las matrices son iguales y 0 en caso contrario.

**13.-** Realiza una función que compruebe si una matriz es simétrica. La función devolverá 1 si la matriz es simétrica y 0 en caso contrario.

**14.-** Dada una matriz **A** de enteros de dimensión nxn que satisface la siguiente propiedad: $\forall i$: $1 \leq i \leq n \quad \forall j$: $1 \leq j \leq n \quad a_{i,j} \leq a_{i,j+1}$, implementa un programa que determine si existen dos elementos ubicados en la diagonal principal (cada uno de ellos en una posición distinta) cuyo producto sea igual a algún otro elemento que ocupe una posición distinta a las de la diagonal principal. Es decir debe determinar si se cumple: $\exists\ i,j,k,l \quad 1 \leq i,j,k,l \leq n \quad i \neq j \quad k \neq l$ tal que $a_{k,k} * a_{l,l} = a_{i,j}$.

**15.-** Un **cuadrado latino** de orden N es una matriz cuadrada que en su primera fila contiene los N primeros números naturales y en cada una de las siguientes N-1 filas contiene la rotación de la fila anterior un lugar a la derecha. Realiza un programa que reciba como parámetro la dimensión del cuadrado y genere la matriz correspondiente a su cuadrado latino

**16.-** Realiza una función *longitud* que devuelva la longitud de una cadena de caracteres sin utilizar el fichero de cabecera <string.h>

**17.-** Escribe una función *void concatena(char * c1,char *c2)* que añada la cadena c1 al final de la cadena c2, sin utilizar el fichero de cabecera <string.h>

**18.-** Implementa un programa que dadas dos cadenas de caracteres, a través de un menú de opciones, realice las siguientes operaciones:

1.- Comparación cadenas.
2.- Copia de la primera sobre la segunda
3.- Concatenación de ambas cadenas
4.- Cálculo de la longitud de las cadenas

**19.-** Realiza una función *int cuenta( char * cad, char c)* que devuelva el número de veces que aparece el carácter *c* en la cadena *cad*.

**20.-** Escribe una función que determine cuál es el carácter que más veces se repite en una determinada cadena y cuántas veces se repite.

**21.-** Implementa una función *void inversa(char * cadena)* que invierta la cadena de caracteres *cadena*. Realiza una versión iterativa y otra recursiva

**22.-** Escribe una función que acepte una cadena de caracteres y determine si dicha cadena es o no un palíndromo. Se dice que una cadena es un palíndromo si se puede leer igual de izquierda a derecha que de derecha a izquierda, ignorando los espacios en blanco.

**23.-** Escribe una función que convierta una cadena de caracteres numéricos de longitud máxima 6 a su valor entero correspondiente.

**24.-** Implementa una función *void comprimir(cadena1, cadena2)* que elimine todo carácter de *cadena1* que exista en *cadena2*.

**25.-** Escribe una función *void primer_caracter(cadena1, cadena2)* que devuelva como resultado la primera posición de *cadena1* en que aparezca cualquier carácter de *cadena2* (o –1 si *cadena1* no contiene ningún carácter de *cadena2*

**26.-** Se desea almacenar la información relativa a 20 personas. De cada persona se almacena su nombre, apellidos, dni, edad. Implementa la estructura necesaria para almacenar dicha información y realiza las funciones necesarias para la inicialización de la estructura.
El programa deberá además recibir por teclado el dni de una persona e imprimir su edad utilizando una función cuyo prototipo sea: *int busca_persona(personas x[], char *dni)*

**27.-** Se dispone de la información relativa a 100 estaciones meteorológicas diferentes repartidas por una determinada área geográfica. Conocemos el nombre de las estaciones y la cantidad de lluvia en litros/$m^2$ que recogieron durante el año pasado en cada uno de los 12 meses. Implementa la estructura adecuada para almacenar dicha información. Realiza las funciones necesarias para determinar en qué punto llovió más y en cual menos y la cantidad media de agua recogida en las estaciones meteorológicas durante cada mes.

**28.-** Se desea almacenar la información relativa a las ventas semanales de los 25 vendedores de una compañía. De cada vendedor se almacena su nombre, apellidos y dni. Realiza un programa que inicialice la estructura y calcule las ventas totales de cada vendedor, las ventas totales de la compañía, y el nombre (e importe) de los dos vendedores que más han facturado.

**29.-** Se desea realizar un programa en C que implemente un diccionario Inglés/Español. El programa debe leer palabras en Inglés y traducirlas a Español hasta que el usuario introduzca la palabra "Fin". Diseña la estructura de datos adecuada para almacenar las parejas de palabras Inglés-Español introducidas por el usuario. Inicializa dicha estructura y realiza el proceso de traducción descrito. El número de parejas de palabras es variable, pero limitado a un máximo de 100, siendo la longitud de cada palabra de un máximo de 40 caracteres.

**Ejemplo:** Si se introducen las siguientes parejas de palabras:
**book** *libro* **green** *verde*

**mouse** *ratón* **door** *puerta*
Cuando el usuario introduzca la palabra **green** la respuesta será *verde*.

**30.-** Un instituto desea almacenar los datos de los N alumnos de un curso. De cada alumno se almacenarán nombre, apellidos, dni, edad y las calificaciones que ha obtenido en cada uno de los tres trimestres de cada una de las 7 asignaturas del curso; junto con las calificaciones deberá almacenarse también el nombre de la asignatura. Diseña la estructura necesaria para almacenar dicha información e implementa las siguientes funciones:

- void **nota_global** (alumno * al)
  Recibe como parámetro exclusivamente los datos de un alumno y debe escribir por pantalla la nota global para cada asignatura.

- void **mayor_nota** (char *asig, alumno *clase)
  Recibe como parámetro los datos de todos los alumnos y el nombre de una asignatura y deberá escribir por pantalla el nombre y apellidos del alumno que más nota haya obtenido en la asignatura especificada por *asig* durante el primer trimestre.

**31.-** Una empresa desea realizar un programa para controlar la permanencia de sus trabajadores en el lugar de trabajo. Para ello debe almacenar la hora de llegada y salida de cada trabajador en horas, minutos y segundos de cada uno de los cinco días laborables de la semana. Además se desea almacenar nombre, apellidos, dni, y salario por hora de cada trabajador. Elige la estructura adecuada para almacenar esta información y realiza las funciones básicas de inicialización e inserción de información.
Suponiendo que cada trabajador debe contar con una permanencia de 35 horas semanales, se considerarán el resto como horas extra. El programa debe obtener el nombre, apellidos y el salario total de cada uno de los trabajadores considerando un incremento de un 10% del salario por hora para las horas extra. Así como, los nombres y apellidos de todos aquellos trabajadores que contabilicen una permanencia semanal menor a la exigida por la empresa.

**32.-** Se desea almacenar la siguiente información de los empleados de una empresa: nombre y apellidos, edad, DNI, ventas realizadas en cada uno de los días de la semana. Diseña la estructura necesaria para almacenar esa información teniendo en cuenta que las ventas diarias de un empleado pueden ser de hasta 100.000 pesetas. Se debe crear un menú para manejar esta estructura implementando las siguientes funciones y respetando sus prototipos:

- void *dia_mas_ventas* (empleado * emp, int * dia);

Recibe la información de un empleado y devuelve en *día* el día de la semana en el que ese empleado ha tenido más ventas.

- char * *menor_empleado* (empleado emp[]);

Recibe la información de todos los empleados de la empresa y devuelve el nombre del empleado más joven de la empresa (si hay más de uno con la misma edad devolverá el primero que encuentre).

- int *total_ventas* (empleado emp);

Recibe un empleado y devuelve el total de las ventas semanales de ese empleado

- int *mismo_nombre*(empleado * emp);

Recibe la información de todos los empleados de la empresa y devuelve 1 si hay dos empleados con el mismo nombre y 0 en caso contrario.

**33.-** Realiza un programa que permita a través de un menú realizar las siguientes operaciones sobre un fichero de texto: crear un fichero, escribir en el fichero sin eliminar su contenido, leer el contenido y escribirlo en pantalla.

**34.-** Realiza un programa que copie un el contenido de un fichero de texto existente en otro y devuelva el número de caracteres, palabras y líneas que contiene el mismo.

**35.-** Realiza un programa que almacene la estructura diseñada en el ejercicio 26 en un fichero y posteriormente copie el contenido de este fichero en una estructura del mismo tipo.

**36.-** Escribe un programa que dado un vector dinámico desordenado de *n* enteros, elimine los valores duplicados, desplazando los elementos y obtenga el número de elementos restantes.

**37.-** Dado un vector dinámico de n elementos inicializados con valores 0 y 1. Escribe una función que reciba dicho vector y el tamaño n y añada un elemento más que represente el bit de paridad, es decir que contenga 0 o 1 para que la paridad sea par, es decir que la cantidad total de 1 sea par.

**38.-** Escribe un programa que cree una matriz de forma dinámica pidiendo al usuario la dimensión de las filas y columnas de la matriz. Posteriormente debe introducir valores en la matriz e imprimir su resultado.

**39.-** Realiza el ejercicio anterior usando una función que reserve memoria para la matriz en lugar de hacerlo en la función main.