Ejercicio1

Crea tres arrays de cinco elementos: el primera de enteros, el segundo de double y el tercero de booleanos. Muestra las referencias en las que se encuentra cada uno de los arrays anteriores.

Ejercicio 2

Construye un array de 10 elementos del tipo que desees. Declara diferentes variables de array que referenciarán al array creado. Comprueba, imprimiendo por pantalla, que todas las variables contienen la misma referencia.

Ejercicio 3

Construye un array de 10 elementos, inserta valores en cada uno de ellos y redimensiónalo a un array de 11 elementos. Cuando lo redimensionamos, no queremos insertar los valores nuevos en el array de uno en uno.

Ejercicio 4

Crea un array de 10 elementos de enteros, dándole valor a cada uno de ellos, llamamos a una función que incremente todos los elementos en 5, llamamos a una función que imprima por pantalla el contenido del array (en ambas funciones, pasamos el array por parámetro).

Ejercicio 5

Crea un array de longitud 10 que se inicializará con números aleatorios comprendidos entre 1 y 100. Mostrar la suma de todos los números aleatorios que se guardan en el array.

Ejercicio 6

Introduce por teclado un número n; a continuación, solicita al usuario que teclee n números. Realiza la media de los números positivos, la media de los negativos y cuenta el número de ceros introducidos.

Ejercicio 7

Diseñar un programa que solicite al usuario que introduzca por teclado 5 números decimales. A continuación, mostrar los números en el mismo orden que se han introducido.

Ejercicio 8

Escribir una aplicación que solicite el usuario cuántos números desea introducir. A continuación, introducir por teclado esa cantidad de números enteros, y por último, mostrar el orden inverso al introducido.

Ejercicio 9

Diseñar la función: **int máximo(int t[])**, que devuelve el máximo valor contenido en un array t.

Ejercicio 10

Escribir la función **int [] rellenaPares(int longitud, int fin)**, que crea y devuelve un array ordenado de la longitud especificada, que se encuentra rellenado con números pares aleatorios comprendidos en el rango desde 2 hasta **fin** (inclusive).

Ejercicio 11

Escribe la función: **int buscar(int t [], int clave)**, que busca de forma secuencial en el array **t** el valor **clave**. En caso de encontrarlo, devuelve en qué posición lo encuentra; y en caso contrario, devolverá -1.

Ejercicio 12

Definir una función que tome como parámetros dos arrays, el primero con los 6 números de una apuesta de la primitiva, y el segundo (ordenado) con los 6 números de la combinación ganadora. La función devolverá el número de aciertos.

Ejercicio 13

Implementar la función: **int [] sinRepetidos(int t[])**, que construye y devuelve un array de la longitud apropiada, con los elementos de **t**, donde se han eliminado los datos repetidos.

Ejercicio 14

Leer y almacenar n números enteros en un array, a partir del cual se construirán otros dos arrays con los elementos con valores pares e impares del primero, respectivamente. Los arrays pares e impares deben mostrarse ordenados.

Ejercicio 15

Escribe en una función el comportamiento de la inserción ordenada.

Ejercicio 16

Diseñar una aplicación para gestionar un campeonato de programación, donde se introduce la puntuación (enteros) obtenida por 5 programadores, conforme van terminando su prueba. La aplicación debe mostrar las puntuaciones ordenadas de los 5 participantes. En ocasiones, cuando finalizan los 5 participantes anteriores, se suman al campeonato programadores de exhibición, cuyos puntos se incluyen con el resto. La forma de especificar que no intervienen más programadores de exhibición es introducir como puntuación un -1. La aplicación debe mostrar, finalmente, los puntos ordenados de todos los participantes.

Ejercicio 17

Escribir la función: **int[] eliminarMayores(int t[], int valor)**, que crea y devuelve una copia del array **t** donde se han eliminado todos los elementos que son mayores que **valor**.

Ejercicio 18

Crea una función que realice el borrado de un elemento de un array ordenado.

Ejercicio 19

El “número de la suerte” de una persona puede calcularse a partir de sus números favoritos. De entre estos, se seleccionan dos diferentes al azar, que se eliminarán de la lista, pero en su lugar se añade la media aritmética de los dos eliminados de la lista de números favoritos. El proceso se repite hasta que solo quede un número, que resultará el número de la suerte para esa persona. Para calcular bien el número de la suerte es imprescindible que la lista de números se encuentre siempre ordenada. Escribe una aplicación que solicite al usuario sus números favoritos y calcula su número de la suerte.

Ejercicio 20

Comprueba qué produce la comparación con el operador **==** de dos arrays del mismo tipo, la misma longitud y los mismos valores.

Ejercicio 21

Desarrollar el juego “la cámara secreta”, que consiste en abrir una cámara mediante su combinación secreta, que está formado por una combinación de dígitos del 1 al 5. El jugador especificará cuál es la longitud de la combinación; a mayor longitud, mayor será la dificultad del juego. La aplicación genera, de forma aleatoria, una combinación secreta que el usuario tendrá que acertar. En cada intento se muestra como pista, para cada dígito de la combinación introducido por el jugador, si es mayor, menor o igual que el correspondiente en la combinación secreta.

Ejercicio 22

Crear un array bidimensional de longitud 5x5 y rellenarlo de la siguiente forma: el elemento de la posición [n][m] debe contener el valor 10 x n + m. Después se debe mostrar su contenido.