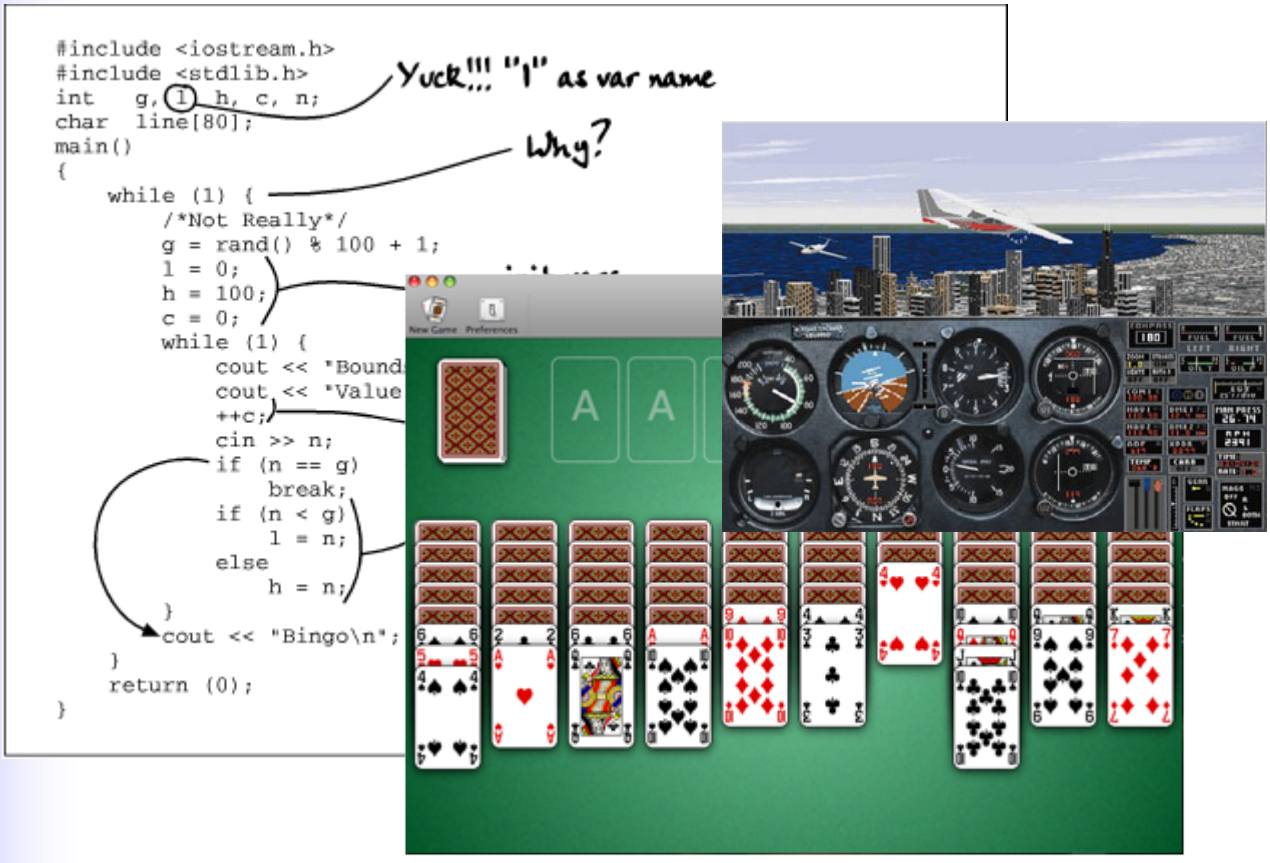
UCA
Universidad de Cádiz

Informática II

GRADO EN MATEMÁTICAS

2014 - 2015



Prácticas de la asignatura

maria.guerrero@uca.es

TEMA 0 : ANTECEDENTES

1. Escribir una función que busque en un vector de números el máximo valor, e indique su valor y posición. Se supone que el vector está desordenado.

2. Escribir una función que busque en un vector de números los valores máximo y mínimo. Se supone que el vector está desordenado.

3. Escribir una función que genere 6 números aleatorios entre 1 y 49, de la misma forma que en la primitiva. Obviamente, los números no pueden ser repetidos.

4. Realizar un programa que calcule los números primos entre 1 y N mediante la *criba de Eratóstenes*. El sistema consiste en crear una tabla con los números del 2 a N. El primer número no tachado (el 2) es primo. Tachar todos los múltiplos del 2. El primer número no tachado (el 3) es primo. Tachar todos los múltiplos del 3. El primer número no tachado (el 5) es primo. Tachar todos los múltiplos del 5. Continuar así hasta que todos los números estén tachados. Por ejemplo, la situación tras tachar todos los múltiplos del 5 es la siguiente :

2	3	4	5	6	7	8	9	10	11	12	13	14	15	*	*	*	N

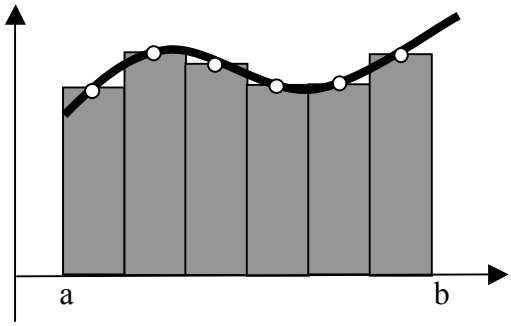
Este sistema se puede mejorar de múltiples formas. Pensar cómo, y aplicarlas.

5. Realizar un programa que almacene en un vector los 100 primeros números primos, y luego los muestre de mayor a menor.

6. Realizar un programa que calcule la integral de una curva f(x) en el intervalo a y b.

$$S = \int_a^b \sin(x)\ dx$$

Para ello se romperá el segmento a y b en N partes, y S se calculará como la suma de las áreas de N rectángulos delimitados por la curva dada y el eje x, tal y como se muestra en el gráfico para N=6. Probar el programa con la función seno, entre 0 y pi/2, con N=1000, y comprobar si el área es 1.



7. Realizar un procedimiento que permita encontrar un número existente en un vector ordenado de números (o determinar que no existe) mediante la búsqueda binaria. Dicha técnica consiste en comparar el número a buscar con el número situado en la mitad del vector. Si el número buscado es menor quiere decir que debemos buscar únicamente en

la primera mitad del vector. En caso contrario, el número estará en la segunda mitad. Ello convierte el problema anterior en un problema similar, pero con un vector de tamaño inferior (en concreto, la mitad).

8. Realizar un procedimiento que permita ordenar vector de números.

9.- Realice una función *longitud(cadena)* que nos devuelva la longitud de una cadena de caracteres. (Sin utilizar la librería *string.h*).

10.- Implemente una función *concatena(s, t)* que añada la cadena *t* al final de la cadena *s*. Se supone que hay espacio suficiente para contener las dos cadenas. (Sin utilizar la librería *string.h*).

11.- Escriba un programa que lea cadenas de caracteres hasta que se introduzca la cadena nula. Para cada una de ellas el programa debe devolver la longitud y su concatenación con la cadena "HOLA".

12.- Implemente un programa que lea dos cadenas de caracteres y muestre un menú con las siguientes opciones (Utilice las funciones adecuadas de la librería *string.h*):

1. Comparar cadenas.
2. Copiar la primera en la segunda.
3. Concatenar ambas cadenas.
4. Determinar la longitud de las cadenas.

13.- Escriba una función que cuente el número de veces que aparece un determinado carácter en una cadena. La función aceptará la cadena y el carácter a buscar devolviendo el número de ocurrencias del mismo en la cadena.

14.- Desarrolle una función que devuelva cuál es el carácter que más se repite en una determinada cadena y cuántas veces se repite.

15.- Escriba una función *inversa(cadena)* que invierta la cadena de caracteres. Es decir, si al llamar a *inversa(cadena)* 'cadena' era "Hola colega", al salir de la función 'cadena' debe ser "ageloc aloH".

16.- Utilice la función anterior para escribir un programa que invierta la entrada línea a línea.

17.- Implemente una función que acepte una cadena de caracteres y devuelva si dicha cadena es o no palíndromo. Una cadena será un palíndromo si se lee igual de izquierda a derecha que de derecha a izquierda, ignorando los espacios en blanco. Por ejemplo: "dábale arroz a la zorra el abad" o "reconocer". Para ignorar la diferencia entre mayúsculas y minúsculas puede utilizar las funciones *toupper(c)* o *tolower(c)* que devuelven el carácter en mayúsculas y minúsculas respectivamente.

18.- Escriba una función que reciba una cadena de caracteres que represente el nombre de alguna entidad y devuelva como resultado en otra cadena, el acrónimo de la entidad. Por ejemplo: "Consejo Superior de Investigaciones Científicas" → "C.S.I.C".

19.- Escriba una función *atoi(cadena)* que convierta una cadena de caracteres del 0 al 9 de longitud máxima 6 a su valor entero correspondiente.

20.- Implemente una función *htoi(cadena)* que convierta una cadena de dígitos hexadecimales en su valor entero correspondiente. La longitud máxima de la cadena será de 4 caracteres.

21.- Escriba una función *comprimir(cadena1, cadena2)* en la que se borre todo carácter de 'cadena1' que exista en 'cadena2'.

22.- Implemente una función *cualquier(cadena1, cadena2)* que devuelva como resultado la primera posición de 'cadena1' en que aparezca cualquier carácter de 'cadena2' (o -1 so 'cadena1' no contiene ningún carácter de 'cadena2').

TEMA 1: TIPOS DE DATOS ESTRUCTURADOS (II)

1. Escribir un programa que pida al usuario los datos de una matriz de tamaño 3x3, y luego la muestre por pantalla.

2. Completar el programa anterior para que muestre la suma de sus filas, y la suma de sus columnas.

3. Escribir un programa que pida al usuario los datos de una matriz de tamaño 3x3, la trasponga, y luego muestre la muestre por pantalla.

4. Escribir un programa que pida al usuario los datos de una matriz de tamaño 3x3, y un número N, reste este valor a cada elemento de la matriz, y muestre la matriz por pantalla.

5. Escribir un programa que pida al usuario los datos de una matriz de tamaño 3x3, y guarde en un vector la diagonal principal, y en otro la otra diagonal.

6. Escribir una función que busque en una matriz de números el máximo valor, e indique su valor y posición. Se supone que la matriz está desordenada.

7. Escribir una función que busque en una matriz de números los valores máximo y mínimo. Se supone que la matriz está desordenada.

8. Escribir una función que reciba dos matrices, y determine si son iguales.

9. Escribir un programa que almacene en una matriz de NxN los valores:

```
1 1 1 1 1 1
2 2 2 2 2 2
3 3 3 3 3 3
4 4 4 4 4 4
5 5 5 5 5 5
6 6 6 6 6 6
```

10. Escribir un programa que almacene en una matriz de NxN los valores:

```
0 1 2 3 4 5
1 2 3 4 5 6
2 3 4 5 6 7
3 4 5 6 7 8
4 5 6 7 8 9
5 6 7 8 9 10
```

11. Escribir un programa que almacene en una matriz de NxN los valores:

```
1 2 3 4 5 6
2 1 2 3 4 5
3 2 1 2 3 4
4 3 2 1 2 3
5 4 3 2 1 2
6 5 4 3 2 1
```

12. Realizar un programa que pida al usuario una matriz A y un vector v y muestre por pantalla el producto: A x v

13. Realizar un programa que pida al usuario dos matrices A y B y muestre por pantalla el producto de ambas: A x B

14. Realizar un programa que gire 90º en sentido inverso al de las agujas del reloj los valores de una matriz cuadrada:

```
1 2 3 2          2 4 2 3
3 4 5 4   ⇨      3 5 1 6
2 3 1 2          2 4 3 5
4 5 6 3          1 3 2 4
```

15. Hacer una función que sume los elementos por encima de la diagonal principal de una matriz. La matriz no tiene necesariamente que ser cuadrada.

16. Escribir un programa que calcule el determinante de una matriz cuadrada.

17. Escribir un programa que resuelva un sistema de ecuaciones lineales de N ecuaciones con N incógnitas por el método de Gauss.

18.- Supongamos la siguiente estructura de datos:

```
struct dir
{
    char nombre[30];
    char calle[40];
    char ciudad[20];
    char provincia[30];
    int código;
```

}info_dir[MAX]

Realice un programa que sea capaz de manejar un vector de estas estructuras con las siguientes funciones:

1. Añadir una dirección.
2. Borrar una dirección.
3. Listar las direcciones que tenemos.
4. Modificar algunos de los registros.

19.- Almacene en un vector, leyendo desde teclado una lista de registros de tipo persona, con dos campos:

1. Nombre: alfanumérico de longitud máxima 30.
2. Edad: entero.

El programa recibirá un nombre y deberá imprimir la edad de esa persona utilizando una función cupo prototipo sea: *int busca(persona x[], char* nombre)*

20.- Tenemos 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 cada uno de los 12 meses.

Determine las estructuras de datos adecuadas para almacenar toda esa información, y suponer que ya contienen esa información.

Determine en qué punto llovió más y en cual menos durante el año pasado. Calcule la cantidad de agua recogida en las estaciones meteorológicas durante cada mes

Además determine el número de estaciones que hay dentro de cada uno de los siguientes grupos:

Grupo	**Intervalo l/m^2**	**Grupo**	**Intervalo l/m^2**
1	0 – 99	6	500 – 599
2	100 – 199	7	600 – 699
3	200 – 299	8	700 – 799
4	300 – 399	9	800 – 899
5	400 – 499	10	900 – 999

Se supone que la cantidad máxima que se puede recoger es de 999 litros/m^2.

21.- Una compañía tiene 25 vendedores. Para cada venta conocemos el número de vendedor, el nombre y el importe. No se conoce el número total de ventas, y además el número de ventas puede variar de un vendedor a otro.

Diseñe un algoritmo que lea los nombres de los 25 vendedores y todas las ventas que ha hecho la compañía (debe finalizar cuando se introduzca en una venta un número de vendedor igual a 0), y que calcule las ventas totales de cada vendedor por separado, las ventas totales de la compañía, y el nombre (e importe) de los dos vendedores que más han facturado.

22.- Se necesita tener información almacenada de los pacientes de una consulta. De cada paciente se necesita conocer su DNI, su nombre, sus apellidos, un teléfono de contacto y una serie de medidas realizadas durante un mes (30 días). Las medidas son las siguientes: su temperatura corporal, la tensión máxima y mínima y el número de pulsaciones. Implemente un programa utilizando estructuras que permita lo siguiente:

a) Inicializar todos estos datos.

b) Proporcione un resumen por cada paciente indicando el número medio de pulsaciones, las tensiones medias y la temperatura media.

c) Muestre una lista con aquellos pacientes que tengan una tensión media comprendida en un intervalo de medidas proporcionado por el usuario.

23.- Se desea tener información almacenada de direcciones de lugares. De cada lugar se necesita saber el nombre, la calle, la ciudad, la provincia y el código postal. Implemente utilizando estructuras y registros un programa que permita:

a) Añadir un elemento.

c) Borrar un elemento.

d) Buscar un elemento.

e) Listar todos los datos.

24.- Una tienda de productos de limpieza necesita almacenar información sobre sus productos. De cada producto se desea saber su nombre, la cantidad existente en el almacén, el precio de compra y el precio de venta del mismo. Implemente un programa que muestre un menú que permita:

a) Inicializar estos datos.

b) Modificar la información.

c) Muestre la cantidad de existencias de un producto determinado.

d) Muestre el producto con mayor margen de beneficio.

e) Liste aquellos productos cuyo precio de venta es superior al precio de venta medio.

25.- Un videoclub desea almacenar información acerca de sus películas. De cada película se necesita almacenar su título, un identificador, el precio y el número de veces que ha sido alquilada. Implemente un programa que permita introducir los datos de las películas, que muestre la película más solicitada (indicando el beneficio obtenido) y la que menos.

TEMA 2: RECURSIVIDAD

1.- Cree una función recursiva que realice el siguiente cálculo:

$$\left(1+\frac{1}{2}+\ \ldots\ +\frac{1}{n}\right)=\left(1+\frac{1}{2}+\ \ldots\ +\frac{1}{n-1}\right)+\frac{1}{n}$$

2.- Realice una función recursiva que calcule la potencia de un número real elevado a un entero positivo. La definición de dicha función se define de la siguiente forma:

$x^0 = 1$

$x^n = (x * x)^{\frac{n}{2}}, si\ n > 0\ \ y\ es\ par$

$x^n = x * x^{n-1}, si\ n > 0\ y\ es\ impar$

3.- Implemente una función recursiva que calcule la cifra i – ésima de un entero n, es decir:

- La última, si i = 0
- La cifra (i – 1) de n /10, en otro caso

4.- Cree una función recursiva que calcule el coeficiente binomial definido de la siguiente forma:

$\binom{n}{0} = \binom{n}{n} = 1$

$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$

5.- Sabiendo que para *inf* y *sup* ϵ Z, tales que *inf* ≤ *sup* , se tiene:

$$\sum_{i=inf}^{sup} a_i = f(x) = \begin{cases} a_i, & \inf = sup \\ \sum_{i=inf}^{med} a_i + \sum_{i=med+1}^{sup} a_i, & \inf < sup \end{cases}$$

(siendo med = (inf + sup) / 2), defina una función recursiva para $\sum_{i=sup}^{sup} \frac{1}{i^2}$, e incluya en un programa que halle $\sum_{i=1}^{100} \frac{1}{i^2}$

6.- Use el hecho de que

$$\int_b^a f(x)dx = \int_a^m f(x)dx + \int_m^b f(x)dx$$

(siendo $m = \frac{a+b}{2}$) para desarrollar una función recursiva que halle aproximadamente la integral definida de la función *sin(x)* a base de dividir el intervalo [a ; b] en dos hasta que sea lo bastante pequeño ($|b-a| < \varepsilon$), en cuyo caso aceptamos que $\int_b^a f(x) \cong (b-a) * f(m)$

7.- Sabiendo que 0 es par y que la paridad de cualquier otro entero positivo es la opuesta que la del entero anterior, desarrolle las funciones lógicas, mutuamente recursivas, **EsPar** y **EsImpar**, que se complementen a la hora de averiguar la paridad de un entero positivo.

8.- Implemente el máximo común divisor mediante una función recursiva.

9.- Implemente mediante recursividad la función factorial.

10.- Escriba utilizando una función recursiva la sucesión de Fibonacci.

11.- Implemente una función recursiva que calcule el cociente entero de dos números mediante restas.

12.- Escriba una función recursiva que calcule la suma de los dígitos de un número.

13.- Implemente una función recursiva para calcular el término enésimo de la secuencia de Lucas: 1, 3, 4, 7, 11, 18, 29, 47, …

14.- Escriba una función recursiva que invierta las cifras de un número.

15.- Implemente una función recursiva que permita hacer una multiplicación, utilizando el método Ruso.

16.- Escriba una función recursiva que sume los elementos de un vector.

17.- Escriba una función recursiva que multiplique los elementos de un vector.

18.- Implemente una función recursiva que determine si un número es positivo o negativo.

19.- Implemente una función recursiva que sume los elementos de una matriz.

20.- Escriba una función recursiva que resuelva el cuadro latino, por ejemplo:

0 0 0 0 1
0 0 0 1 2
0 0 1 2 3
0 1 2 3 4
1 2 3 4 5

21.- Implemente una función recursiva que muestre por pantalla la matriz de cubo mágico.

22.- Escriba una función recursiva que busque el mínimo de un vector.

23.- Escriba una función recursiva que busque el máximo de un vector.

TEMA 3: PUNTEROS

1.- Dadas las siguientes definiciones y asignaciones:

```
int a[10] = {1,5,8,4,9,11,12,7,6,5};
int *p, *q;
int i;

p = &a[0];
q =&a[1];
```

- Ilustre como serían las asignaciones de memoria a las variables e indique con una flecha las posiciones a las que apuntarían las variables punteros.
- Indique gráficamente el estado de la memoria luego de ejecutar cada una de las siguientes sentencias en forma secuencial:

- i = *p + *q;
- *p += *q; /* o lo que es lo mismo: *p = *p + *q; */
- *(p+1) = i+10;
- i = *(q-1) / *(p+9);
- *p = *(p+10-3);
- q = p+5;

2.- Escribir una función que reemplace en la cadena de caracteres "s" todas las apariciones del carácter "viejo" por el carácter "nuevo". Además, elimine los espacios en blanco, desplazando los caracteres útiles hacia la izquierda.

Nota: El encabezado de la función deberá ser el siguiente:
void reemplazar (char *s, char nuevo, char viejo);

3.- Realizar un programa que lea una palabra y ordene en forma alfabética los caracteres que la forman. El ordenamiento debe realizarse sobre la misma estructura de almacenamiento donde se encuentra la palabra original. Los caracteres repetidos deberán ser eliminados. Solo se pueden utilizar variables simples como auxilio de programación. La salida del programa deberá mostrar la palabra ingresada y luego los caracteres ordenados en minúsculas.

4.- Realizar un programa que lea 25 números enteros positivos, determine el mayor número par y el menor número impar de todos. Se deben utilizar funciones que calculen el menor, el mayor y la determinación de si es par o no. Las funciones definidas deberán retornar el resultado correspondiente.

5.- Escribir un programa que lea dos cadenas de caracteres alfabéticos y un número. El mismo debe insertar la segunda cadena en la primera, a partir del r de la primera cadena que está en la posición indicada por el número.
Nota: deberá realizar todos los controles necesarios para poder realizar dicha operación.

6.- Escriba un programa que ingrese dos números A(real) y B(entero). Luego, en una función deberá sustituir el contenido de A por el sin(A) y el contenido de B por A^B. Los dos números ingresados deben ser almacenados en variables locales a la función *main*.

7.- Escribir un programa que acepte un número seguido de un espacio y luego una letra. Si la letra que sigue al número es una ‘f’, el programa deberá manejar el número introducido como una temperatura en grados Fahrenheit, convertirla en grados Celsius e imprimir un mensaje adecuado de salida. Si la letra que sigue al número es una ‘c’, el programa deberá tratar al número como una temperatura en grados Celsius, convertirla a grados Fahrenheit, e imprimir un mensaje adecuado de salida. Si la letra no es ni una ‘f’ ni una ‘c’, el programa deberá imprimir un je que diga que los datos son incorrectos y terminar.
Nota: para la conversión de grados deberá utilizar la fórmula usada en el práctico anterior.

8.- Escribir un programa que lea dos cadenas de caracteres denominadas "s1" y "s2" respectivamente, y verifique la existencia de la cadena “s2” como subcadena integrante de la “s1”, imprimiendo como salida el cartel que corresponda.

9.- Realice un programa que permita almacenar para 5 usuarios los tiempos de ejecución de sus respectivos programas. Cada usuario puede ejecutar HASTA 10 programas. Se pide calcular el tiempo promedio de uso de la computadora para cada usuario. Como salida deberá mostrar los tiempos y su promedio. Las variables a usar deberán ser locales a la función *main*. Tanto el ingreso de los datos como el cálculo del promedio e impresión de los datos y resultados deberán ser desarrollados en sus respectivas funciones.
10.- Realice un programa que contenga una función llamada **copiarVector()** que reciba dos vectores y el tamaño de los mismos (deben de ser del mismo tamaño) y que consiga copia en el segundo array el contenido del primero. Hágalo utilizando punteros.

11.- Cree un programa llamado **paresImpares** que cree un vector de 100 números aleatorios del 1 al 1000. Una vez creado, mostrar el contenido y después organizarlo de forma que estén juntos los elementos pares y los impares. Después, volver a mostrar el vector.

12.- Cree un programa llamado **vendedores** que cree una matriz de 18 X 10 indicando que poseemos una empresa de 18 vendedores cada uno de los cuales vende 10 productos.

El vector almacena los ingresos obtenidos por cada vendedor en cada producto, de modo que un menú permite almacenar los ingresos, revisar el total de cada vendedor y obtener los ingresos totales.

13.- Escriba un programa que mediante un menú admita reservar o cancelar asientos de un avión, así como mostrar qué asientos están ocupados y libreas actualmente.

La matriz tendrá 25 filas y 4 columnas.

14.- Crear un programa que cree un array con 1000 letras mayúsculas aleatorias y que cuenta cuántas veces aparece cada letra en el vector.

15.- Escriba un programa que pida una serie de números al usuario y halle el máximo, el mínimo y la media aritmética de ellos. Para ello se debe crear una variable puntero tipo ***float*** , pedir al usuario que introduzca el número de datos, y después los datos a calcular. Recordar que se debe reservar memoria de forma dinámica para almacenar el vector de datos. La salida del programa debe ser algo así:
Numero de datos: 10
Máximo: 25
Mínimo: 4

Media Aritmética: 14.6

16.- Implemente un programa que evalúe un ***polinomio de cualquier grado***:

$$P(X)=a0+a1x+a2x2+a3x3+...+anxn$$

Como en un principio no se sabe de qué grado va a ser este polinomio, se debe crear una variable puntero del tipo double:

double *coeficientes;

A continuación se debe pedir el ***grado*** del polinomio y reservar memoria para el array de coeficientes. Luego, a través de un bucle se empezará a pedir los coeficientes del polinomio. Finalmente se ***debe pedir el punto (x)*** en el que se va a evaluar el polinomio, cuyo resultado se sacará por la consola. También se debe sacar por la consola el polinomio que se ha evaluado.

La entrada y salida de datos por la consola podría ser:
>> Introduce el número de coeficientes: **2**
>> Introduce el coeficiente 1: **2.5**
>> Introduce el coeficiente 2: **3.5**
>> Introduce el valor de x: **2.0**
>> El polinomio es: **2.5*X^0 + 3.5*X^1**
>> El valor del polinomio en el punto **2.0** es **9.5**

17.- Este ejercicio consiste en implementar un algoritmo para el cálculo de la integral definida de un polinomio entre dos puntos (A y B) dados. El ***polinomio debe ser de cualquier grado***:

$$P(X)=a0+a1x+a2x2+a3x3+...+anxn$$

Los pasos a seguir son:

1. Pedir el grado del polinomio.
2. Crear un ***array*** de tipo ***double*** para albergar el polinomio (utilizar ***new***).
3. Pedir los coeficientes del polinomio.
4. Pedir los puntos A y B para evaluar la integral (A < B).
5. Pedir el número de subintervalos "***n***".
6. Mostrar el valor de la integral por consola.

El algoritmo se basa en la división del intervalo (A, B de la figura) en ***n*** subintervalos, sumando las áreas de éstos. Si los subintervalos son lo suficientemente pequeños el área de cada subintervalo se puede aproximar al área de un trapecio. En definitiva, la integral se puede aproximar como:

$$\int_b^a f(x) \approx \Sigma ((f(xi)+f(xi+1)/2)*(xi+1-xi))$$

TEMA 4: FICHEROS

1.- Escriba un programa que reciba de la línea de comandos del Sistema Operativo el nombre de un fichero e imprima por pantalla su contenido.

2.- Realice un programa que lea de la entrada estándar y almacene todas las pulsaciones en un fichero un fichero, cuyo nombre debemos pasarle como parámetro en la línea de comandos del Sistema Operativo. El programa leerá caracteres, almacenándolos en el fichero hasta que se introduzca el carácter '$'.

3.- Realice una función que lea una línea de un fichero de entrada y la almacene en un vector.

4.- Realice una función que lea una palabra de un fichero de entrada y la almacene en un vector.

5.- Implemente un programa que lea un fichero y copie su contenido en otro fichero de salida. Los nombres de los ficheros deben ser introducidos a través de la línea de comandos del Sistema Operativo.

6.- Implemente un programa que borre un fichero especificado por el usuario.

7.- Escriba un programa que concatene dos ficheros de texto en un fichero "salida.txt".

8.- Realice un programa que cambie un carácter por otro en un fichero y almacene el resultado en otro llamado "salida.txt". El programa recibirá el nombre del fichero original, el carácter a modificar y el carácter que lo sustituirá.

9.- Escriba un programa que lea de la línea de comandos los nombres de dos ficheros, los compare, e imprima por pantalla las líneas donde encuentre diferencias.

10.- Realice un programa que reciba un entero *n*, una palabra *p* y el nombre de un fichero de texto. El programa debe copiar el fichero a un fichero de salida insertando una línea con la palabra *p* (a modo de separador) cada *n* líneas.

11.- Suponemos un fichero que contiene en cada línea las coordenadas de un punto en el espacio. Almacene el contenido de este fichero en un vector, suponiendo que cada elemento del vector debe contener un registro con tres campos.

12.- Realice una función que nos devuelva el tamaño de un fichero ya abierto que se le pasa como parámetro.

13.- Implemente un programa que lea de su entrada estándar un entero *n* y un nombre de fichero. Debe leer del fichero las últimas *n* líneas y almacenarlas en un fichero de salida.

14.- Escriba un programa que reciba el nombre de un fichero de texto. El programa debe contar las líneas no vacías que hay en el fichero e imprimir este número por pantalla. Se entiende por línea vacía aquella que tiene solo los caracteres de final de línea, ningún otro.

15.- Realice una función que invierta un fichero devolviendo este resultado en otro fichero. La función recibirá los ficheros ya abiertos.

16.- Implemente un programa que simule el juego del ahorcado donde el usuario tiene que adivinar las palabras que le propone el ordenador. Las palabras con las que se juegan deben estar almacenadas en un fichero "palabras.dat" separadas por espacios en blanco. En todo momento se debe mostrar al usuario las letras que ha dicho y que no están en la palabra y el número de intentos que le quedan antes de perder.

17.- Implemente un programa que lea el contenido de un fichero y muestre por pantalla dicho contenido escrito en código Morse. La traducción es la siguiente:

Letra	Transcripción
A	.-
B	-…
C	-.-.
CH	----
D	-..
E	.
F	..-.
G	--.
H	….
I	..
J	.---
K	-.-
L	.-..
M	--
N	-.
Ñ	--.--
O	---
P	.--.
Q	--.-
R	.-.
S	…
T	-
U	..-
V	…-
W	.--
X	-..-
Y	-.--
Z	--..
0	-----
1	.----
2	..---
3	…--
4	….-
5	…..
6	-….
7	--…
8	---..
9	----.

18.- Las notas de los alumnos de una clase de Bachillerato se encuentran en el fichero “alumnos.dat”. El formato del fichero es el siguiente: en cada línea se encuentra el nombre del alumno, las notas obtenidas en cada uno de los trimestres y la asignatura a la que corresponden dichas notas. Implemente un programa a utilizando estructuras que calcule la nota global de cada asignatura por cada alumno y que indique cual es el alumno que ha sacado mayor nota por cada asignatura.

TEMA 5: DISEÑO DE PROGRAMAS

1.- Escriba un programa que acepte una cadena de caracteres introducida por teclado y determine:

- El número total de letras.
- El número de letras minúsculas.
- EL número de letras mayúsculas.
- El número de dígitos pares.
- El número de signos de puntuación (‘,’, ‘;’. ‘:’, ‘.’, ‘?’, ‘¿’, ‘!’, ‘¡’).
- El número de espacios en blanco.
- El número de caracteres de la cadena que aparezcan también en la cadena constante “La asignatura de Informática es maravillosa”.

El programa debe tener la siguiente estructura:

- Un fichero .c que contenga la función principal.
- Un fichero .c donde aparezca la implementación de todas las funciones que sean necesarias para diseñar el programa.
- Un fichero .h donde se incluyan las bibliotecas que se necesiten, así como las declaraciones de las funciones que se vayan a utilizar.

2.- Implemente el juego de “Los Barquitos” (el usuario juega contra el ordenador). El programa debe tener la siguiente estructura:

- Un fichero .c que contenga la función principal.
- Un fichero .c donde aparezca la implementación de todas las funciones que sean necesarias para diseñar el programa.
- Un fichero .h donde se incluyan las bibliotecas que se necesiten, así como las declaraciones de las funciones que se vayan a utilizar.

3.- En un tablero 4x3, con tres caballos blancos situados en las 3 casillas superiores y 3 negros en las 3 casillas inferiores, diseñe un programa para intercambiar las posiciones de los caballos blancos por las de los negros, en un número mínimo de jugadas. El programa debe tener la siguiente estructura:

- Un fichero .c que contenga la función principal.
- Un fichero .c donde aparezca la implementación de todas las funciones que sean necesarias para diseñar el programa.
- Un fichero .h donde se incluyan las bibliotecas que se necesiten, así como las declaraciones de las funciones que se vayan a utilizar.

4.- Implemente el juego “Tres en Raya” (el usuario juega contra el ordenador). Para ello, puede utilizar ficheros, matrices bidimensionales y vectores. El programa debe tener la siguiente estructura:

- Un fichero .c que contenga la función principal.

- Un fichero .c donde aparezca la implementación de todas las funciones que sean necesarias para diseñar el programa.
- Un fichero .h donde se incluyan las bibliotecas que se necesiten, así como las declaraciones de las funciones que se vayan a utilizar.

5.- Uno de los juegos con monedas más antiguos consiste en colocar en fila 8 monedas, y tratar de agruparlas, en 4 movimientos, en 4 montones de 2 monedas cada uno. En cada movimiento, cada moneda debe saltar exactamente sobre otras dos (independientes o apiladas). El programa debe tener la siguiente estructura:

- Un fichero .c que contenga la función principal.
- Un fichero .c donde aparezca la implementación de todas las funciones que sean necesarias para diseñar el programa.
- Un fichero .h donde se incluyan las bibliotecas que se necesiten, así como las declaraciones de las funciones que se vayan a utilizar.

6.- A principios de siglo fue comercializado el siguiente juego, consistente en un caja de madera con 15 bloques que podían ser deslizados por la caja (sin sacarlos de ella). Los bloques venían en la posición inicial:

1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	

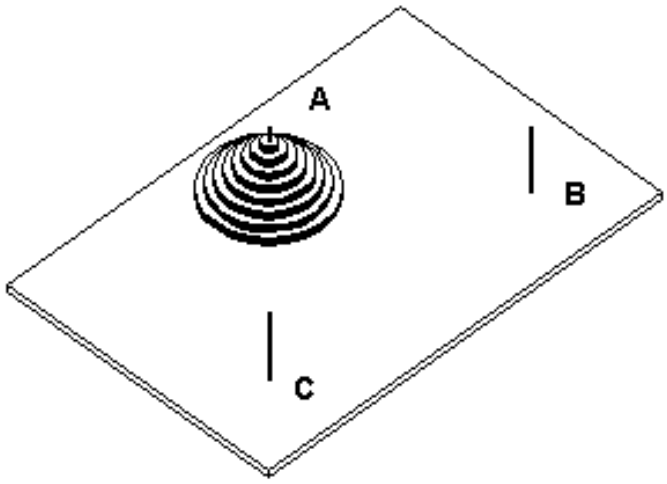
El problema consiste en llevarlos a su posición natural:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

El programa debe tener la siguiente estructura:

- Un fichero .c que contenga la función principal.
- Un fichero .c donde aparezca la implementación de todas las funciones que sean necesarias para diseñar el programa.
- Un fichero .h donde se incluyan las bibliotecas que se necesiten, así como las declaraciones de las funciones que se vayan a utilizar.

7.- El rompecabezas de las torres de Hanoi consta de tres varillas verticales fijas en un tablero. En una de las varillas se encuentra ensartada una pila de n discos de tamaño decreciente, con el disco mayor en la base. El problema consiste en trasladar la pila a otra varilla, moviendo un disco cada vez, de manera que en ningún momento un disco descanse sobre otro de menor tamaño. Inicialmente sólo es posible mover el disco de menor tamaño. El segundo movimiento también está forzado. A partir del tercer movimiento, la elección ya no es única.



El programa debe tener la siguiente estructura:

- Un fichero .c que contenga la función principal.
- Un fichero .c donde aparezca la implementación de todas las funciones que sean necesarias para diseñar el programa.
- Un fichero .h donde se incluyan las bibliotecas que se necesiten, así como las declaraciones de las funciones que se vayan a utilizar.

8.- Tres tiradores A, B y C acuerdan batirse en un duelo triangular con las siguientes reglas: Situados en los vértices de un triángulo equilátero y después de acordar el orden, dispararán por turno un disparo a aquel de los otros dos que prefiera. El duelo continuará en este orden hasta que únicamente quede uno de los duelistas. Se sabe que A acierta siempre, B un 80% y C un 50%. Suponiendo que todos escogen la mejor estrategia y que nadie resulta herido por un disparo no dirigido a él, ¿quién tiene más probabilidades de sobrevivir? El programa debe tener la siguiente estructura:

- Un fichero .c que contenga la función principal.
- Un fichero .c donde aparezca la implementación de todas las funciones que sean necesarias para diseñar el programa.
- Un fichero .h donde se incluyan las bibliotecas que se necesiten, así como las declaraciones de las funciones que se vayan a utilizar

9.- Disponemos de tres medias nueces y de una canica, que escondemos debajo de una de las tres nueces. El usuario debe intentar adivinar debajo de qué nuez ha puesto la canica el ordenador. Una vez que has elegido una de las nueces el PC destapará una de las otras dos que no esconda la canica. Llegado este punto, hay dos nueces boca abajo, una de las cuales esconde la canica y el ordenador volverá a preguntar al usuario donde está la canica. Este proceso se repetirá hasta que el usuario no quiera cambiar más de lugar, entonces terminará el juego mostrando donde está la canica y si el usuario ha acertado. El programa debe tener la siguiente estructura:

- Un fichero .c que contenga la función principal.
- Un fichero .c donde aparezca la implementación de todas las funciones que sean necesarias para diseñar el programa.
- Un fichero .h donde se incluyan las bibliotecas que se necesiten, así como las declaraciones de las funciones que se vayan a utilizar

10.- Supongamos que te has hecho una prueba, para una cierta enfermedad, que tiene una fiabilidad del 98%, esto es, si alguien tiene esa enfermedad, la prueba dará resultado positivo el 98% de los casos, mientras que si no la tiene, será negativo el 98% de los casos. Supongamos también que 1 de cada 200 personas que se realizan la prueba realmente padece la enfermedad. Supongamos finalmente que te has hecho la prueba y ha dado positivo. ¿Cómo de preocupado (o preocupada) deberías estar? El programa debe tener la siguiente estructura:

- Un fichero .c que contenga la función principal.
- Un fichero .c donde aparezca la implementación de todas las funciones que sean necesarias para diseñar el programa.
- Un fichero .h donde se incluyan las bibliotecas que se necesiten, así como las declaraciones de las funciones que se vayan a utilizar

TEMA 6: CONCEPTOS AVANZADOS DE C

1.- Escribir una función que muestre un número de 8 bits (unsigned char) en formato binario.

2.- Escribir una función `ponerbits(x,p,n,y)` que devuelva `x,` con `n` bits que empiezan en la posición `p` y están a la derecha de una variable y unsigned char, dejándolos en `x` en la posición `p` y a la izquierda dejando los otros bits sin cambio. Por ejemplo, si `x = 10101010 (170 decimal), y = 10100111 (167 decimal), n = 3 y p = 6`, entonces se necesita tomar 3 bits de `y` (111) y ponerlos en `x` en la posición 10*xxx*010 para obtener la solución 10111010. La solución deberá mostrarse en binario, sin embargo la entrada puede ser en forma decimal. Una posible salida podría ser la siguiente:

```
x = 10101010 (binario)
y = 10100111 (binario)
ponerbits n = 3, p = 6 da x = 10111010 (binario)
```

3.- Escribir una función que invierta los bits de x (unsigned char) y guarde el resultado en y. La salida deberá mostrarse en binario, aunque la entrada puede estar en forma decimal. Una posible salida podría ser:

```
x = 10101010 (binario)
x invertida = 01010101 (binario)
```

4.- Escribir una función que rote (no desplace) a la derecha *n* posiciones de bits de *x* del tipo unsigned char. La salida deberá mostrarse en binario y la entrada puede ser en forma decimal. Una posible salida podría ser:

```
x = 10100111 (binario)
x rotada por 3 = 11110100 (binario)
```

5.- Implementa un programa que permita mostrar por pantalla un fichero cuyo nombre de le pasa al programa como argumento mediante la línea de comandos.

6.- Escribe un programa que permita copiar el contenido de un fichero en otro. Los nombres de los ficheros se le pasan como argumento mediante la línea de comandos.

7.- Escribe un programa que lea mediante la línea de comandos un fichero organizado en 5 columnas. El programa debe sumar cada columna por separado y cuando llegue al final de éste calculará e imprimirá la media aritmética de cada columna. Por ejemplo la entrada podría ser:

```
12 34 45 7  8
8  7  54 32 45
12 34 43 5  5
```

Y el programa imprimiría: `10 25 47 14 19`

8.- Implemente un programa que lea mediante la línea de comandos un fichero organizado en 10 columnas. El programa debe mostrar por pantalla la diagonal mayor y la menor de la matriz que contiene el fichero.

9.- Definir una macro `min(a,b)` para determinar el entero más pequeño. Definir otra macro `min3(a,b,c)` en términos de `min(a,b)`. Incorporar las macros en un programa demostrativo en donde se pida al usuario tres números y se muestre el más pequeño.

10.- Definir un nombre de macro de preprocesador para seleccionar los bits menos significativos de un tipo de dato `unsigned char` y el *n*-ésimo bit (asumiendo que el menos significativo es 0) de un tipo de dato `unsigned char`.