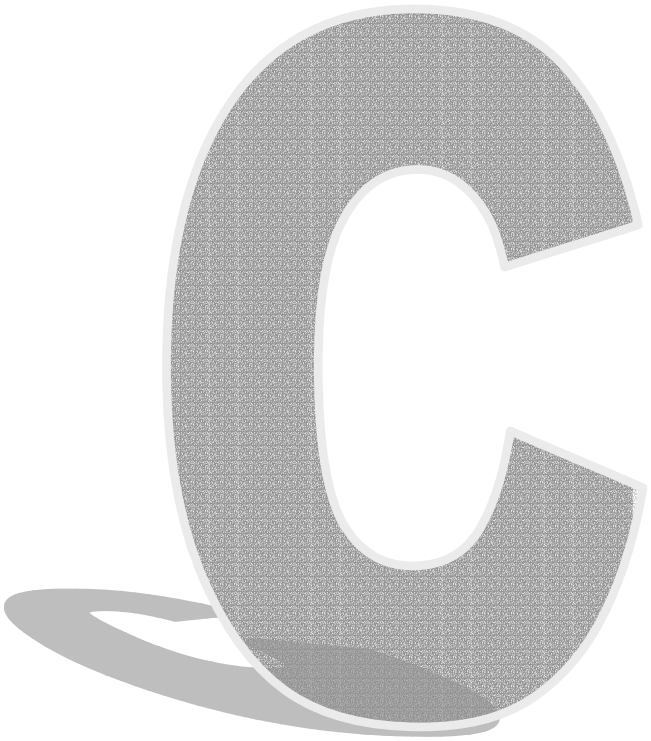
Ejercicios de programación en



Informática Industrial

Ingeniería en Automática y Electrónica Industrial

Dpto. de Ingeniería de Sistemas y Automática

Isidro Calvo Gordillo
Fabián López Valencia

Curso 2006-07

INDICE

1. Variables y operadores básicos 1
2. Sentencias condicionales 3
3. Sentencias repetitivas 6
4. Funciones 10
5. Arrays 14
6. Punteros 18
7. Cadenas 20
8. Estructuras 25
9. Ficheros 30

Variables y operadores básicos

1. Programa que pida un valor en pesetas y lo convierta en euros y otro programa que lleve a cabo la operación inversa.

   Nota 1 : La solución en euros deberá tener 2 decimales.

   Nota 2 : Utilizar `#define` para definir la relación entre euros y pesetas: 1 euro = 166.386 Pts

2. Pedir un carácter por teclado y mostrar un mensaje que muestre el carácter introducido así como su código ASCII tanto en formato decimal como hexadecimal.

3. Programa que pida la temperatura en grados Celsius y la convierta a grados Fahrenheit (y viceversa) mostrando en pantalla un mensaje del tipo "*xxx.xx grados Celsius son yyy.yy grados Fahrenheit*"

$$\frac{F-32}{9}=\frac{C}{5}$$

4. Crear un programa que pida el radio de una circunferencia y que calcule la longitud de la misma así como el área incluida dentro.

5. Crear un programa que calcule la fuerza de atracción gravitacional entre dos masas, M1 y M2 situadas a una distancia R.

$$F = G\frac{M1*M2}{R^2}(Nw)$$

   donde las masas se expresan en kilogramos y la distancia en metros y la constante de gravitación universal vale:

$$G = 6.67x10^{-11}\,Nw*m^2/Kg^2$$

   Nota: Utilizar *#define* para definir la constante G.

6. Escribir un programa que pregunte el año actual y la edad de una persona y calcule la edad de esa persona en el año 2010.

7. Escribir un programa que calcule el número de billetes de 10.000, 5.000, 1,000, así como de monedas de 500, 100, 25, 5 y 1 pesetas para desglosar una cantidad, C, de pesetas (menor de 2.147.483.647), de forma que se necesite la menor cantidad de monedas y billetes de cada tipo.

8. Crear un programa que pida un numero real y muestre la tabla de multiplicar correspondiente a dicho número de tal manera que tanto los valores de los factores como del producto se presenten encolumnados y con una precisión de 2 dígitos.

Ejemplo:

```
Programa que muestra la tabla de multiplicar de un número
Escribe un número: 5  (Valor introducido por el usuario)

TABLA DE MULTIPLICAR DEL NUMERO 5.00

        5.00  *     1     =      5.00
        5.00  *     2     =     10.00
        ...
        5.00  *     10    =     50.00
```

9. Mostrar en forma de tabla, el cuadrado y el cubo de los 5 primeros números enteros que siguen a uno introducido por teclado.

   Los datos deben aparecer ajustados a la derecha siguiendo el siguiente formato:

```
   Numero     Cuadrado         Cubo
   ------     --------         ----
      xxx         xxxx       xxxxxx
```

   Nota: Ejecutar el programa utilizando variables de tipo int e introduciendo el número base 30 y utilizando variables de tipo unsigned int e introduciendo el número base 40. ¿Qué sucede? ¿Cómo se puede explicar lo que sucede?

10. Crear un programa que muestre en pantalla el tamaño en bytes (8 caracteres) de las variables más frecuentes: *char, int*, *short int*, *long int*, *float* y *double.*
    (**Nota:** Para calcular el tamaño de una variable se puede usar el operador *sizeof*.)

11. Escribir un programa que tras preguntar el **número de almacén asociado** a un determinado tipo de pieza, **la cantidad pedida** de esa pieza y **el precio por unidad**, efectúe el cálculo del precio total de las piezas pedidas. El programa deberá escribir toda la información de la pieza en la pantalla, además del importe total del pedido.

12. Escribir un programa que lea el valor de un ángulo en radianes y muestre su valor en grados, minutos y segundos

13. Programa que tras pedir por teclado un número lo multiplique por 4 y divida por 2 utilizando los operadores de rotación.

Sentencias condicionales

1. Ejecutar el siguiente código fuente (Declarando `valor_logico` como entero y luego declarándolo como *`float`*):

```
printf("Valores logicos de las siguientes expresiones\n");
valor_logico=(3>5);
printf(" (3 > 5) es %d\n", valor_logico);
valor_logico=(5 > 3);
printf(" (5 > 3) es %d\n", valor_logico);
valor_logico=(15 > 3*5);
printf(" (15 > 3*5) es %d\n", valor_logico);
valor_logico=!(5 == 3);
printf("!(5 == 3) es %d\n", valor_logico);
```

   Nota: No confundir el operador '==' de comparación (para comprobar si dos valores son iguales) con el operador '=' de asignación que escribe un valor en una variable.

2. Escribir un programa que tras pedir 2 números por la pantalla muestra cuál es el mayor número. (Hágase con la sentencia *`if`* y con el operador condicional: expresión ? valor1 : valor2)

3. Leer tres números enteros y, si el primero de ellos es negativo, calcular el producto de los tres, en caso contrario calcular la suma de ellos.

4. Crear un programa que calcule la caída de potencial producida por una resistencia según la ley de Ohm (V = I * R) a partir de la resistencia y la intensidad que pasa a su través.

   Nota: El programa no debe aceptar resistencias negativas, dado que no tienen sentido físico, ni resistencias mayores que 1000Ω (requerimiento adicional del problema). En ambos casos el programa debe escribir un mensaje de error en pantalla diciendo que el valor de la resistencia está fuera de límites aceptables indicando la causa por la que dicho valor para la resistencia ha sido rechazado.

5. Sea un sistema de ecuaciones de la forma:

$$a\,x + b\,y = c$$
$$d\,x + e\,y = f$$

   que puede resolverse usando las siguientes fórmulas: $x = \dfrac{ce - bf}{ae - bd}$ $y = \dfrac{af - cd}{ae - bd}$

   Escribir un programa que lea los coeficientes (a, b, c, d, e, f) y resuelva el sistema. El programa deberá indicar los casos en los que el sistema de ecuaciones no tenga solución.

6. Escribir un programa que calcule las raíces, bien reales o imaginarias, de una ecuación de segundo grado.

   El programa también debe ser capaz de operar con valores nulos para el coeficiente de orden dos (es decir, deberá de ser capaz de resolver ecuaciones de primer grado)

7. Dada la función U = f (x, y) tal que:

$$\begin{cases} 10 & \text{si} \quad x * y < 1 \\ y^2 & \text{si} \quad x * y \geq 1 \end{cases}$$

   y dada la función V = f (x, y) tal que:

$$\begin{cases} 1 & \text{si} \quad x * y < 1 \\ y^2 & \text{si} \quad x * y \geq 1 \end{cases}$$

   Escribir un programa que calcule los valores de las funciones U y V, una vez conocidas las coordenadas de un punto (x, y).

8. Escribir un programa que pida un año y diga si es bisiesto o no.

   Nota: Un año es bisiesto si es múltiplo de 4 salvo el caso en que sea múltiplo de 100, que no es bisiesto, y no sea múltiplo de 400. Por ejemplo, el año 1900 no fue bisiesto, el 2000 sí y el 2100 no lo es.

9. La fecha de cualquier Domingo de Pascua se calcula de la siguiente forma:

   Sea X el año para el que se quiere calcular la fecha.

   Sea A el resto de la división de X entre 19

   Sea B el resto de la división de X entre 4

   Sea C el resto de la división de X entre 7

   Sea D el resto de la división de (19 * A + 24) entre 30

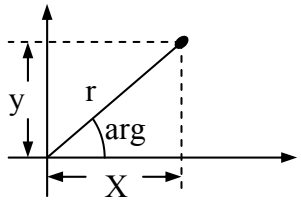
   Sea E el resto de la división de (2 * B + 4 * C + 6 * D + 5) entre 7

   La fecha para el Domingo de Pascua es el día (22 + D + E) de Marzo (obsérvese que puede dar una fecha en mes de Abril)

   Escribir un programa que pida como entrada un año y saque por pantalla la fecha del Domingo de Pascua para ese año.

10. Leer desde el teclado las coordenadas (x, y) de un punto y, mediante un menú, elegir una entre las siguientes opciones:

    - Comprobar si el punto pertenece a una circunferencia de radio 10 y centro (0,0)
    - Averiguar el cuadrante en el que se encuentra el punto.
    - Pasar las coordenadas cartesianas (x, y) a polares (r, arg).



11. Escribir un programa que permita efectuar el cálculo del área de un cuadrado, un círculo o un triángulo equilátero según la opción seleccionada por el usuario a través de un menú.

    - Crear una primera versión con la estructura de control *if ... else*
    - Y una segunda versión con la estructura de control *switch... case...*

12. Dada la *longitud* de una circunferencia, averiguar si dicha longitud corresponde, con una determinada *precisión*, a una circunferencia de *radio* R.

    El programa deberá leer la *longitud* de la circunferencia, el *radio* y la *precisión* e indicará si es cierto o no que esa circunferencia tiene ese valor del radio.

13. Escribir un programa que pida dos caracteres por pantalla, los ordene alfabéticamente, y los imprima ordenados.

Sentencias repetitivas

1. Escribir un programa que escriba los números del 1 al 100 en líneas de 10 números. Después de 100 el programa debe escribir "*Fin del programa*" en una línea nueva.

2. Programa que calcule el valor de elevar un número real, a, a un exponente entero, b, $\mathbf{a^b}$, multiplicando b veces el número a.

   Nota: Mejorar el programa para que compruebe que el exponente es mayor que 0 y si no lo es dar un mensaje de error y pedir otro exponente.

3. Programa que calcule el factorial (n!) de un número entero positivo leído por teclado.

   Nota: Probar el programa con los números 6, 7, 8, 9...

4. Crear un programa que pida un numero real y muestre la tabla de multiplicar correspondiente a dicho número de manera que tanto los valores de los factores como del producto se presenten encolumnados y con una precisión de 2 dígitos.

   Ejemplo:

```
Programa que muestra la tabla de multiplicar de un número
Escribe un número: 5  (Valor introducido por el usuario)

TABLA DE MULTIPLICAR DEL NUMERO 5.00
5.00 *     1    =     5.00
...
5.00 *    10    =    50.00
```

5. Escribir un programa que calcule el sumatorio:

$$\sum_{i=1}^{S} (-1)^i * \frac{1}{i^2}$$

   donde S es un número entero positivo introducido por teclado.

   Solución: El límite de esa expresión cuando S tiende a infinito es: -0.822467.

6. Programa que escriba en pantalla una tabla con cuadrados y cubos a partir de un número base hasta otro tope, ambos pedidos por teclado.

   Nota: El programa utilizará sólo variables de tipo ***short int*** y deberá evitar errores de overflow.

Ejemplos de ejecución:

```
Tabla de cuadrados y cubos

Escribe el número menor: 30
       el número mayor: 34

   Número  Cuadrado      Cubo
   ======  ========  ========
       30       900     27000
       31       961     29791
       32      1024
       33      1089
       34      1156
```

```
Tabla de cuadrados y cubos

Escribe el número menor: 179
       el número mayor: 183

   Número  Cuadrado      Cubo
   ======  ========  ========
      179     32041
      180     32400
      181
      182
      183
```

Nota: en **negrita** se han indicado los valores introducidos por el usuario durante la ejecución del programa.

7. Escribir un programa que ayude a aprender las tablas de multiplicar.

   Para ello se irá pidiendo la tabla de multiplicar de un número (pedido por teclado con anterioridad) y comprobando que los valores introducidos son correctos. Si es así el programa escribirá "correcto" y en caso contrario deberá escribir "Lo siento, se ha equivocado. La respuesta correcta era *número*"

   La última línea mostrará el número de aciertos.

   A continuación se muestra un ejemplo de ejecución:

   ```
   Programa para aprender las tablas de multiplicar
   Con qué número quieres practicar?  5 (Introducido por
   usuario)

   5 * 1 = 5 (Introducido por el usuario)
   Valor correcto
   5 * 2 = 11 (Introducido por el usuario)
   Lo siento se ha equivocado. La respuesta correcta era 10
   ...
   Has acertado 9 veces de 10 números.
   ```

8. Hacer un programa que lea caracteres desde teclado hasta que lea 10 veces la letra 'a'. Por cada carácter leído que no sea una 'a' debe mostrar un mensaje indicándolo. Cuando lea las 10 letras 'a' el programa terminará.

9. Hacer un programa que lea caracteres desde teclado y vaya contando las vocales que aparecen. El programa terminará cuando lea el carácter **#** y entonces mostrará un mensaje indicando cuántas vocales ha leído (cuántas de cada una de ellas).

10. Repetir el ejercicio leyendo caracteres hasta que se lea el carácter final de fichero **EOF** (**^Z**) en lugar del carácter **#**.

11. Programa que simule que se deja caer una pelota desde un edificio de X metros de altura (donde X se pide por teclado) mostrando en cada 0.1 segundos tanto la altura de la pelota como su velocidad.

    Mostrar para cada instante de tiempo (cada 0.1 segundos) una línea del estilo:

    ```
    t=xx.x distancia al suelo=xx.xx metros velocidad=xx.xx m/s
    ```

12. Programa que calcule los nºs primos del 1 al 100 y los saque por pantalla.

13. Escribir un programa que primero pida por pantalla con cuántos números se va a trabajar digamos que sean X) y luego pida los X números por pantalla.

    Después de introducir los X números se mostrará un mensaje por pantalla indicando cuál es el mayor y menor valor introducido, así como el valor medio de todos los números introducidos.

14. Programa que pida números de cuatro cifras e indique si los números son capicúas o no. El programa deberá ir pidiendo números hasta que el usuario introduce '-1' por teclado. El número –1 indicará la finalización de la ejecución del programa.

    Nota1: Un número capicúa es simétrico p.e. 1221 ó 25752

    Nota2: Cuando el número no es de cuatro cifras se deberá mostrar un mensaje de error por pantalla y se pedirá otro número menor que 10000. En caso de que el número sea menor de cuatro cifras se completará con ceros a la izquierda.

    Nota3: El único número negativo que se aceptará es '-1' que indicará la finalización del programa. Cualquier otro número negativo, se mostrará un mensaje de error.

15. Escribir un programa que calcule los números perfectos entre 1 y 10000.

    Nota: Un número perfecto es aquél tal que la suma de sus divisores menos el propio número es el propio número.

    Ejemplos:

    $6 \Rightarrow$ Divisores(6) = {1, 2, 3, 6} Suma = 1 + 2 + 3 + 6 – 6 = 6 $\Rightarrow$ N.º Perfecto

    $10 \Rightarrow$ Divisores(10) = {1, 2, 5, 10} Suma = 1 + 2 + 5 + 10 – 10 $\neq$ 10 $\Rightarrow$ No perfecto

    Solución: 6, 28, 496, 8128

16. Escribir un programa que muestre el siguiente menú y que permita pasar magnitudes de grados a radianes y de radianes a grados.

    ```
    1. Pasar de grados a radianes
    2. Pasar de radianes a grados
    3. Salir del programa
    ```

17. Escribir un programa que muestre una tabla con los caracteres ASCII mostrados en decimal, octal y hexadecimal. El programa mostrará la información con el siguiente formato:

    ```
    Dec:  xx   Octal:  xx  Hex:  xx  Car:  x
    ```

    El programa pedirá el primer carácter y los últimos caracteres que marcarán los límites de la tabla.

18. Escribir un programa que calcula el producto de los dígitos de un número entero leído desde teclado.

Funciones

1. Escribir un programa que permita convertir grados Fahrenheit a Celsius y grados Celsius a Fahrenheit.
   El programa presentará el siguiente menú:

   ```
   1.   Conversión de Celsius a Fahrenheit
   2.   Conversión de Fahrenheit a Celsius
   0.   Salir del programa.
   ```

   Nota: Cada conversión se efectuará por medio de funciones, una que convertirá de grados Celsius a grados Fahrenheit y otra que haga justo lo contrario.

2. Realizar un programa que escriba todos los números enteros menores que un cierto entero N y que a su vez sean múltiplos de dos números enteros A y B introducidos por teclado.

   Utilizar para ello una función que admita dos parámetros I, J e indique si I es múltiplo de J.

3. Escribir una función (con su correspondiente programa de prueba) que tome un valor entero y devuelva el número con sus dígitos en reversa. Por ejemplo, dado el número 7631 la función deberá devolver 1367.

4. Escribir un programa que calcule masa radioactiva de carbono 14 que queda después de **t** años. La fórmula que determina la masa restante en el tiempo **t** es:

$$M_t = m * \left(\frac{1}{2}\right)^{\frac{t}{h}}$$

   donde:

t	es el tiempo en años
M_t	es la masa que permanece en el tiempo t
m	es la masa original
h	es la vida media en años

   Para el carbono 14 la vida media es de 5700 años; si la masa original es de 300 gramos, imprimir una tabla de la masa para t = 500, 1000, 1500, 2000,... 10000 años.

   Se deberá utilizar un subprograma que permita evaluar la expresión $(^1/_2)^{t/h}$ para los diferentes valores de t.

   Nota: En TurboC existe la función *double pow(double x, double y)* que devuelve el valor x^y.

5. Escribir una función que escriba tantas líneas en blanco como se haya pedido con anterioridad al usuario en el programa principal.

6. Escribir una función que tome el tiempo introducido por el usuario en el formato (horas:minutos:segundos) y lo convierta en segundos. El programa utilizará esta función para calcular la diferencia en segundos entre dos tiempos introducidos por el usuario.

7. Escribir un programa que calcule el número combinatorio

$$\left(\frac{M}{N}\right) = \frac{M!}{N!(M-N)!}$$

   Utilizar para ello una función que calcule el factorial de un número.

   Nota: La función para calcular el factorial de un número puede ser iterativa o recursiva. (Una función recursiva es una función que se llama a sí misma)

8. Escribir una función que calcule el factorial de un número y utilizar ésta en un programa que muestre el siguiente menú.

```
1.   Factorial de un número
2.   Cálculo de e
3.   Cálculo de e^x
0.   Salir
```

   Nota 1: El cálculo de *e* debe hacerse con la siguiente expresión matemática:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$$

   Nota 2: e^x puede calcularse mediante la fórmula:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

   Nota 3: La precisión con la que se obtiene el resultado (e o e^x) depende del último valor añadido en la correspondiente serie.

9. El desarrollo en serie de Taylor de la función coseno es:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

   donde el ángulo x viene expresado en radianes.

   Escribir un programa que calcule el valor aproximado de coseno (x), utilizando para ello los N primeros términos de la serie de Taylor. El número de términos N de la serie dependerá del valor del último, es decir, se añadirán términos a la serie hasta que el valor absoluto del último término añadido sea menor que 0.0005.

10. Se desea realizar un programa que permita hacer cálculos con polinomios de segundo grado.

    El programa deberá presentar un menú en pantalla que facilite las siguientes opciones:

```
1. Leer un polinomio
2. Escribir un polinomio en su forma habitual
3. Evaluar un polinomio en un punto
4. Calcular el polinomio derivado
0. Salir
```

    Se deberán utilizar los siguientes subprogramas:

    LEER — mediante este subprograma se introducen los números enteros que representan los coeficientes del polinomio.

    ESCRIBIR — es un subprograma que permite la escritura de un polinomio en la forma: `Ax^2 + Bx + C`

    Ejemplos válidos: `5x^2+3x+5` `5x^2-3x+5` `5x^2-5`

    Ejemplos no válidos: `5x^2+ -3x+5` `5x^2-0x+5`

    EVALUAR — es un subprograma que retorna el valor de un polinomio para un número real que es introducido como parámetro.

    DERIVAR — es un subprograma que calcula el polinomio derivada de uno dado.

    Nota: No está permitido el uso de variables globales.

11. Escribir un programa que realice la descomposición en factores primos de un número introducido por teclado.

    El programa deberá ir escribiendo la tabla de los factores primos, a medida que los va calculando, tal como muestra el ejemplo siguiente:

```
Introduce un Nº entero ->    84

        Nº    |    Factores primos
        --    |    ---------------
        84    |    2
        42    |    2
        21    |    3
        7     |    7
        1     |
```

    Ayuda: Se deberá utilizar una función que nos diga si un número es primo o no.

12. Escribir un programa que cuente de un texto introducido por teclado:

    - N.º de caracteres en blanco
    - N.º de dígitos
    - N.º de letras
    - N.º de líneas
    - N.º de otros caracteres

    Nota 1: Se deben crear sendas funciones para comprobar si un carácter es numérico o alfanumérico.

    Nota 2: La función *getchar()* permite leer un carácter de teclado.

    Nota 3: Para marcar el final de lectura del texto, el usuario deberá introducir un carácter que marque fin de fichero. Este carácter es ^D en Linux y ^Z en DOS / Windows.

13. Escribir un programa que lea dos números complejos y permita realizar con ellos las siguientes operaciones aritméticas: suma, resta, multiplicación y división

    Nota 1: Se debe crear una función de permita leer un número complejo (su parte real y su parte imaginaria).

    Nota 2: Se debe crear una función de permita pasar un número complejo en forma parte real y parte imaginaria a módulo y argumento.

    Se debe crear una función que permita pasar un número complejo en forma módulo y argumento a parte real y parte imaginaria.

    Nota 3: La suma y resta de números complejos se obtiene sumando, o restando, las partes reales y las partes complejas.

    El producto de dos números complejos se obtiene multiplicando sus módulos y sumando sus argumentos.

    El cociente de dos números complejos se obtiene dividiendo sus módulos y restando sus argumentos.

Arrays

1. Escribir un programa que calcule el producto escalar y vectorial de dos vectores de 3 elementos cuyos valores se introducen por pantalla con el programa principal.

2. Realizar un programa que lea 20 números (entre el 1 y el 10) y muestre aquel o aquellos que hayan aparecido más veces.

   El programa preguntará si se quieren introducir los 20 números y en el caso en que la respuesta sea negativa rellenará el array con números aleatorios.

   Nota: Para la generación de los números aleatorios, se deberán utilizar las funciones *rand*, *srand*, *time* y la constantes definida *RAND_MAX*.

   `int rand (void):` retorna un valor pseudoaleatorio entre 0 y el valor de la constante RAND_MAX.

3. Escribir un programa que emplee un argumento de la línea de comandos para realizar una conversión decimal a hexadecimal; es decir, el número decimal se introducirá en la línea de comandos, siguiendo al nombre del programa.

```
Ej:
C:> decihex 128 111
Deci= 128  Hex= 80
Deci= 111  Hex= 6F
C:>
```

   Nota 1: Al igual que en el ejemplo, el programa deberá ser capaz de convertir varios números en una llamada.

   Nota 2: En caso de que el programa no reciba argumentos deberá devolver un mensaje de error.

   Nota 3: La función *atoi()* convierte de cadenas a enteros.

4. Escribir un programa que pida un array de caracteres por pantalla e invierta el orden de los caracteres mostrándolo por pantalla. La inversión se hará sin utilizar otro array auxiliar.

5. Escribir un programa que calcule los números primos de 0 a 100 utilizando el llamado método de la criba de Eratóstenes. Este método consiste en definir e inicializar con todos sus elementos a *True* un array de 100 elementos binarios e ir "tachando" (pasando a *False*) en pasadas sucesivas todos los múltiplos de los números primos (2, 3, 5, 7...) hasta obtener sólo los números primos. Es decir:

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

   En el ejemplo en gris claro se señalan los múltiplos de 2, mientras que en gris oscuro los múltiplos de 3 (que no son múltiplos de 2).

Nota: Aumentar el límite superior y comparar la velocidad de ejecución con el programa que calcula los números primos de *Sentencias Repetitivas*.

6. Realizar un programa que maneje un array de enteros a través de un menú con seis opciones:

   1.- Añadir un elemento al array (comprobando que el array no esté lleno)
   2.- Eliminar un elemento del array (comprobando que el array no esté vacío)
   3.- Listar el contenido del array
   4.- Contar las apariciones de un número en el array
   5.- Calcular la media y el máximo de los elementos del array
   0.- Terminar

7. Escribir un programa que permita guardar las cuentas de un banco con sus respectivos saldos. Para ello se guardará la información en un array paralelo (dos arrays unidimensionales, uno con los números de cuenta y otro los saldos)

   El programa deberá mantener las cuentas ordenadas, de menor a mayor, por número de cuenta para facilitar la búsqueda de una cuenta.

   El programa mostrará un menú con las siguientes opciones:

   1.- Dar de alta una nueva cuenta (comprobando que el array no esté lleno y colocando la cuenta en la posición correspondiente dentro del array)
   2.- Eliminar una cuenta (comprobando que el array no esté vacío y reposicionando las cuentas en el array)
   3.- Mostrar una cuenta (mostrará el número de cuenta y el saldo correspondiente)
   4.- Mostrar información (Número de cuentas dadas de alta y dinero total de todas ellas)
   5.- Calcular el saldo medio, máximo y mínimo de las cuentas del array.
   6.- Mostrar todas las cuentas (1 línea por cuenta con su número y su saldo).
   0.- Terminar

8. Escribir un programa que rellene automáticamente una matriz 4x7 con la suma de sus índices (Es decir, el elemento a_{11} = 1+1=2, a_{47} = 4+7=11). El programa mostrará la matriz resultante por pantalla.

9. Escribir un programa que pida una matriz de orden 3x3 y calcule y muestre por pantalla su matriz traspuesta.

10. Crear un programa que permita reservar asientos de una sala de cine (8 filas x 20 columnas). La posición de cada asiento se definirá con una letra (A-H) para la fila y un número (1-20) para la columna. El programa deberá visualizar qué sitios están disponibles para que el cliente pueda decidir dónde sentarse. Antes de hacer la

reserva, el programa deberá comprobar que el asiento está libre, en caso contrario devolverá un mensaje de error.

Ej. de visualización:

```
     1...5....0 1...5....0
     ____________________

A          **            A
B        **** ***        B
C    ** ****  ** ****    C
D   ***  * ** *** ** *** D
E  ***** **** **    **   E
F    ******** *** *   ** F
G  **    **** ****    ** G
H     ******* ***        H

     1...5....0 1...5....0
```

11. Un *histograma* es un gráfico que muestra la frecuencia con que aparecen en una array dado valores dentro de subintervalos especificados de su intervalo. Por ejemplo, si un array unidimensional de enteros tiene elementos de tipo 0..9 y contiene los siguientes valores:

```
6  4  4  1  9  7  5  6  4  2  3  9  5  6  4
```

Su histograma sería:

```
              4              *
Frecuencia    3              *     *
              2              *  *  *        *
              1     *  *  *  *  *  *  *     *
Valor            0  1  2  3  4  5  6  7  8  9
```

Esto indica que los valores 0 y 8 no aparecen en el array, los valores 1, 2, 3 y 7 aparecen una vez, el valor 5 aparece dos veces, el valor 6 tres veces y el valor 4 aparece 4 veces.

Escribir un programa que, tras leer las notas de los alumnos en una asignatura, **genere** y **visualice** el histograma de las notas redondeadas a valores enteros: 0, 1, 2,...10

Las notas leídas desde el teclado son valores de tipo real y el número de alumnos no es fijo aunque siempre es menor de 300.

Nota: Para la realización del histograma se tendrán en cuenta las dimensiones de la pantalla no permitiéndose más de 22 filas, por tanto si la máxima frecuencia es mayor de 22 se deberá mostrar el histograma a escala.

12. Escribir un programa que rellene un array con números aleatorios de tipo float. El programa, primero pedirá la cantidad de números aleatorios (máximo 50) y el rango de los números aleatorios (valor_min .. valor_max) y posteriormente mostrará todos los números aleatorios en el orden en que se han generado y finalmente en orden de menor a mayor.

Nota: Para la generación de los números aleatorios, se deberán utilizar las funciones *rand*, *srand*, *time* y la constantes definida *RAND_MAX*.

Variación 1: Repetir el programa generando números de tipo entero.

Variación 2: Repetir el programa recibiendo los valores (cantidad de números, valor_min y valor_max) como argumentos de la línea de comandos.

13. Escribir un programa que calcule las suma de dos matrices A y B.

    El programa preguntará por las dimensiones de las matrices (dimensiones máximas 5 x 5) y a continuación, si se puede efectuar la suma, introducirá los elementos de ambas matrices y realizará la suma, mostrando finalmente el resultado.

14. Escribir un programa que calcule el producto de dos matrices A y B: $\underset{(m\,x\,n)}{\overline{\overline{C}}} = \underset{(m\,x\,p)}{\overline{\overline{A}}} \; x \; \underset{(p\,x\,n)}{\overline{\overline{B}}}$

    El programa leerá las dimensiones y los elementos de cada una de las matrices (dimensiones máximas 5 x 5) y a continuación, si se puede efectuar el producto, realizará lo calculará y mostrará el resultado en forma matricial.

    Nota: Los elementos de la matriz productos se obtienen de la siguiente forma:

$$c_{ij} = \sum_{k=1}^{p} a_{ik} * b_{kj}$$

Punteros

1. Escribir un programa que efectúe las siguientes operaciones.

   a) Declarar las variables enteras largas **value1** y **value2** e inicializar **value1** a 200000

   b) Declarar la variable **lPtr** como apuntador a un objeto de tipo **long**.

   c) Asignar la dirección de la variable **value1** a la variable de apuntador **lPtr**.

   d) Imprima el valor del objeto al que apunta **lPtr**.

   e) Asígnele a la variable **value2** el valor del objeto al que apunta **lPtr**.

   f) Imprima el valor de **value2**.

   g) Imprima la dirección de **value1**.

   h) Imprima la dirección almacenada en **lPtr.** ¿Es igual el valor impreso que la dirección de **value1**?

   Nota: %p muestra el valor de la variable como puntero

2. Crear un programa que calcule el valor de la intensidad que pasa a través de una resistencia dada, cuando se le aplica un voltaje determinado.

   El programa deberá estar dividido en las siguientes funciones:

   - `explicar_programa ()`

     Esta función mostrará una introducción del programa por la pantalla.

   - `obtener_valores ()`

     Esta función pedirá los valores para la resistencia y voltaje los cuales se pasarán *por referencia* al programa principal.

   - `calcular ()`

     Esta función efectuará el cálculo de la intensidad a partir de la resistencia y el voltaje aplicado.

   - `imprimir_respuesta ()`

     Esta función se encargará de mostrar un mensaje con los resultados.

3. Crear una función que intercambie el contenido de dos variables. Para ello se pasarán como parámetros las direcciones de las variables.
   Para probar la función escribir un programa que pida los datos por pantalla y muestre los contenidos después de llamar a la función.

4. Crear un programa que lea un número determinado (<100) de números reales introducidos por teclado los almacene en un vector para mostrarlos luego en orden inverso.

   Nota: Para recorrer el array se deberá usar aritmética de punteros en lugar de usar los índices del array.

5. Escribir una función que tras pedir un día de la semana (de 1 a 7) devuelva un puntero a cadena con el nombre del día. La función contendrá un array de apuntadores a cadena.
   Para probar la función se realizará un programa que pida un día de la semana en número y escriba el día de la semana en letra.

6. Escribir un programa que inicialice una cadena con una palabra cualquiera. El programa deberá obtener la dirección de la primera letra de la cadena. Una vez sabida esta dirección la mostrará por pantalla y realizará un bucle dando 3 oportunidades para que el usuario introduzca la dirección de la tercera letra de la cadena. En caso de no introducirla bien después de los 3 intentos, deberá sacar un mensaje indicando cuál es la dirección correcta.

Cadenas

1. Escribir un programa que pida una cadena de caracteres (de longitud máxima 80 caracteres) y devuelva la cadena escrita al revés.

   Ayuda: Para saber la longitud de la cadena se puede usar la función *strlen()* de la librería *string.h*.

2. Realizar un programa que lea una cadena de caracteres de una longitud menor de 80 y visualice los caracteres de la siguiente forma:

   primero, último, segundo, penúltimo, tercero, antepenúltimo, ...

3. Escribir una **función** que cambie las letras mayúsculas de una cadena a minúsculas y viceversa. El programa principal pedirá una cadena por pantalla y se la pasará a dicha función esperando el resultado correcto que se mostrará por pantalla.

4. Escribir un programa que pida primero un carácter por teclado y que luego pida una cadena. El programa calculará cuántos caracteres tiene la cadena hasta que lea el carácter introducido primero. Se deberá mostrar un mensaje en pantalla con el número de caracteres introducidos hasta llegar al carácter primero.

5. Escribir un programa que cuente el número de letras, dígitos y signos comunes de puntuación de una cadena introducida por teclado.

   Ayuda: Para saber si un carácter es numérico comparar que su valor es mayor que '0' y menor que '9', para saber si es alfabético comprobar que está entre 'a' y 'z' y considerar signos de puntuación el resto de los caracteres.

   Nota: No considerar ni la ñ ni las letras acentuadas, ya que tienen códigos ASCII fuera del rango a-z

6. Realizar un programa que lea una cadena de caracteres con espacios en blanco excesivos: elimine los espacios en blanco iniciales y finales y sólo deje uno entre cada dos palabras.

7. Crear un programa que pida una cadena de caracteres y devuelva otra sin signos de puntuación ni números. La cadena devuelta debe tener todos los caracteres en mayúsculas.

   Ayuda:
   Se pueden usar las siguientes funciones estándar de C: ispunct(), islower(), gets()
   En TurboC también existe la función: strupr()

   Código ASCII de 'A' 65
   'a' 97

8. Crear un programa que pida por pantalla una cadena de 80 caracteres de longitud máxima y que calcule el número de veces que aparece otra cadena determinada, también pedida por teclado.

   Como salida el programa debe escribir un mensaje con el número de veces que aparece la palabra dada.

   Ayuda: Se pueden usar las funciones siguientes: *strstr(), gets()*

9. Escribir un programa que compruebe si una cadena pedida por teclado es un palíndromo o no. El programa no tendrá en cuenta si la palabra está escrita con mayúsculas o minúsculas.

   Nota: Un palíndromo es una palabra que se lee igual al derecho que al revés.

   Ej: `radar, 11011011, Ana, Otto`

10. Escribir una función que compare 2 cadenas de caracteres devolviendo –1 si son iguales y 0 si son distintas.

11. Escribir un programa que pida dos cadenas (de longitud máxima 10 caracteres) y muestre por pantalla el resultado de las siguientes operaciones:

    - Obtener la longitud de ambas cadenas
    - Comparar alfabéticamente ambas cadenas indicando si son iguales o bien cuál es la mayor y cuál la menor.
    - Concatenar la segunda cadena al final de la primera, dejando un espacio blanco entre ambas.
    - Copiar el contenido de la segunda cadena en la primera.

12. Escribir un programa con un menú que permita las siguientes opciones:

    - Introducir una cadena de 40 caracteres de longitud máxima

    - Pasar a mayúsculas una cadena leída desde teclado. Para ello, escribir un procedimiento que transforme caracteres de letras minúsculas a mayúsculas dejando como están las letras mayúsculas.
      Nota: Para llevar a cabo esta operación tener en cuenta la representación de los caracteres en ASCII. (‘A’ – 65, ‘a’ – 97)

    - Pasar a minúsculas una cadena desde teclado. Para ello, escribir un procedimiento que transforme caracteres de letras minúsculas a mayúsculas. Dejando como están las letras minúsculas.

    - Dada la cadena de caracteres introducida en el punto 1, obtener otra, de forma que la cadena resultante tenga sus caracteres a una distancia **d** (en el código ASCII) de los caracteres de la cadena original. Se considerará el alfabeto circular, es decir que tras la letra ‘z’ viene la letra ‘A’. La distancia **d** se introducirá desde teclado.

Nota: Antes de traducir la cadena se convertirá en una cadena de letras minúsculas, tal y como se hace en el punto 2.

- Salir del programa

Nota común: En todos los casos si se intenta efectuar alguna operación antes de introducir la cadena se deberá mostrar un mensaje de error comunicando al usuario que la cadena está vacía.

13. Julio Cesar enviaba mensajes a sus legiones encriptando los mensajes mediante el siguiente algoritmo:

    Se escogía un número *n* como clave y se sumaba a cada letra en el alfabeto *n* posiciones. Así, si la clave escogida fuese 5, la 'a' pasaría a ser la 'f', mientras que la 'f' pasaría a ser la 'k'. Para las últimas letras del abecedario se seguiría desde el principio. Así, con la clave de 5 la 'y' pasaría a ser la 'd'.

    Se pide crear un programa que encripte una frase mediante este algoritmo.

14. Construir un programa que implemente una calculadora para números enteros:

    El programa pedirá primero la operación y luego los operandos.

    Las operaciones válidas serán: *sumar*, *restar*, *multiplicar*, *dividir*, *salir*.

    Si la operación es distinta de salir se pedirán los operandos y luego se mostrará el resultado.

    Si la orden es distinta de las anteriores se mostrará un error diciendo que se trata de una orden desconocida.

    Los operandos se recogerán como cadenas de caracteres y se convertirán en números enteros con la función *atoi()*

    Otras funciones que se pueden usar serán: *gets()* y *strcmp()*.

15. Realizar un programa que permita calcular el NIF., conocido el DNI. de una persona.

    El programa deberá leer, sobre una cadena de caracteres, el número del DNI. del interesado. Seguidamente deberá averiguar si es un valor válido (todos los caracteres deben ser numéricos y representar un valor entre 100.000 y 99.999.999). Si la entrada es válida se deberá calcular el NIF. y representar el número completo con los puntos de millares y millones en las posiciones correspondientes, así como la letra del NIF al final de la cadena de caracteres separada por un espacio en blanco.

Cálculo de la letra del NIF:

Se obtiene el resto de la división del número del DNI, entre 23, y en función del resultado se asigna un carácter según la siguiente tabla:

0 = 'T'	7 = 'F'	14 = 'Z'	21 = 'K'
1 = 'R'	8 = 'P'	15 = 'S'	23 = 'E'
2 = 'W'	9 = 'D'	16 = 'Q'	
3 = 'A'	10 = 'X'	17 = 'V'	
4 = 'G'	11 = 'B'	18 = 'H'	
5 = 'M	12 = 'N'	19 = 'L'	
6 = 'Y	13 = 'J'	20 = 'C'	

16. Tal vez el esquema de codificación más famoso de todos es el código Morse, desarrollado por Samuel Morse en 1832 para el sistema telegráfico. El código Morse asigna una serie de puntos y rayas a cada letra del abecedario, a cada dígito y a algunos caracteres especiales (punto, coma, dos puntos y punto y coma). Ver tabla adjunta.

Escribir un programa que lea una frase y la codifique en código Morse. También escriba un programa que lea una frase en código Morse y la convierta en su equivalente en castellano. La separación entre letras se indicará mediante un espacio, mientras que la separación entre palabras se indicará mediante 3 espacios.

Carácter	Código
A	. -
B	- . .
C	- . - .
D	- . .
E	.
F	. . - .
G	- - .
H	….
I	. .
J	. - - -
K	- . -
L	. - . .
M	- -
N	- .
O	- - -
P	. - - .
Q	- - . -
R	. - .
S	…

Carácter	Código
T	-
U	. . -
V	. . . -
W	. - -
X	- . . -
Y	- . - -
Z	- - . .
Dígitos	
1	. - - - -
2	. . - - -
3	. . . - -
4	. . . . -
5	. . . . .
6	- . . . .
7	- - . . .
8	- - - . .
9	- - - - .
0	- - - - -

17. Escribir un programa que permita al usuario realizar las siguientes operaciones:

    1. Mostrar la fecha y hora por pantalla
    2. Sacar por pantalla el contenido de un fichero ASCII.
    3. Sacar por pantalla el contenido de un directorio
    4. Limpiar la pantalla
    5. Salir del programa

    Para ello se hará uso de los comandos del sistema operativo:
    En Linux: date, cat, ls, clear
    En DOS: date, time, type, dir, cls

Estructuras

1. Realizar un programa que permita realizar las operaciones básicas (sumar, restar, multiplicar y dividir) números complejos.

   El programa deberá utilizar una variable que represente el número complejo en su forma polar, con sus dos componentes *módulo* y *argumento*.

2. Crear un programa que lea las siguientes variables proporcionadas desde teclado con el siguiente formato:

Posición de los caracteres	***Campo***
1-8	Matricula
9-13	Cilindrada
14-16	Potencia
17-27	Modelo
28-38	Marca

   Ej: `BI6755CC1400 75FIESTA    FORD`

   Y las introduzca en la correspondiente estructura. El programa deberá mostrar la estructura obtenida para comprobar que la conversión ha sido correcta.

   Nota: Construir una función que muestre por pantalla la estructura recibida. La estructura deberá pasarse por referencia para no malgastar espacio en la pila.

3. Disponemos de la información correspondiente a una jaula de un Zoo en una variable de tipo registro con los siguientes campos:

Numero de jaula	Entero	
Especie del animal	Cadena de caracteres	
Nombre del animal	Cadena de caracteres	
Edad	Entero	
Peso	Real	
Kilogramos de comida diaria	Real	
Frecuencia de limpieza de jaula	Entero	(veces al día)
Estado de la salud del animal	Carácter (B, R, M -> Buena, Regular o Mala)	
Descendencia	Sí o No	
Peligroso	Sí o No	

   Por motivos de transferencia de información a otros organismos necesitamos descomponer la información contenida en esa variable en dos variables diferentes:

   Una va a contener los datos de mantenimiento de la jaula del animal:

Numero de jaula	Entero	
Kilogramos de comida diaria	Real	
Frecuencia de limpieza de jaula	Entero	(veces al día)
Peligroso	Sí o No	

Otra, los datos del animal:

Especie del animal	Cadena de caracteres
Nombre del animal	Cadena de caracteres
Edad	Entero
Peso	Real
Estado de la salud del animal	Carácter (B, R, M → Buena, Regular o Mala)
Descendencia	Sí o No

Escribir un programa que lea una variable de información global y la descompongan en dos variables, una de información de mantenimiento y otra de información del animal y visualice ambas variables.

Nota: Dentro de los subprogramas que considere necesarios deberá haber uno, llamado *descomponer*, que tome un registro y lo descomponga en dos.

4. Crear un programa que permita introducir cierta información relativa a los vuelos diarios que parten de un aeropuerto en un array formado por registros. Cada registro contendrá la siguiente información sobre el vuelo correspondiente:

   a) **Número de vuelo** (*No tiene por qué coincidir con el índice del array*)

   b) **Hora de partida** (En dos campos): 1. **Hora:** 0..23
      2. **Minutos:** 0..59

   c) **Origen del vuelo:** Cadena de caracteres

   d) **Destino del vuelo:** Cadena de caracteres

   e) **Número de pasajeros:** Entero

   Una vez introducidos los datos de todos los vuelos se preguntará si se desea obtener información de algún vuelo. En caso de que el usuario responda afirmativamente se pedirá el número de vuelo. El programa buscará el vuelo en al array y accederá a la información que contiene a partir de su número de vuelo, mostrando por pantalla todos sus datos.

   El programa se ejecutará repetitivamente hasta que el usuario indique que no desea obtener más información de ningún vuelo.

   Nota: Se deben diseñar las funciones que visualicen un vuelo, busquen un vuelo en el array, introduzcan la información de un vuelo en el array, etc...

5. Una compañía utiliza aviones para fumigar las cosechas contra una gran variedad de plagas. Lo que la compañía cobra a los granjeros depende de contra qué es lo que desean fumigar, y de cuantos m^2 de tierra quieren que se fumiguen de acuerdo con la siguiente tabla:

Tipo1:	Fumigación contra malas hierbas	18 € / m^2
Tipo2:	Fumigación contra langostas	36 € / m^2
Tipo3:	Fumigación contra gusanos	54 € / m^2
Tipo4:	Fumigación contra todo lo anterior	90 € / m^2

   Además, si el área a fumigar es mayor que 10.000 m^2, el granjero goza de un descuento del 7%.

   Se trata de escribir un programa que lea los datos de un conjunto de granjeros y al final calcule, para cada uno de ellos, la factura correspondiente. De cada granjero se tendrá la siguiente información:

   Nombre

   Tipo de fumigación (código entre 1 y 4)

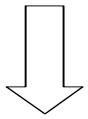
   Número de m^2 que se van a fumigar.

   Fecha de factura (a su vez, con los componentes: minutos, hora, día, mes y año)

   Nota: Los componentes de la fecha de la factura deben ser leídos en el sistema (utilizando funciones de la librería `time.h`)

6. Escriba un programa que genere un array de 50 números enteros entre 1 y 1000 de forma aleatoria. A continuación, partiendo de ese array genere otro array de 50 estructuras tal que cada estructuras tenga un campo para el número y otro (cadena de caracteres) para indicar si dicho número es primo o no. Por ejemplo:

203	672	23	319	426	1	.....	862	373	273	203



203	672	23	319	426	1	.....	862	373	273	203
NO	NO	SI	NO	NO	SI	.....	NO	SI	NO	NO

   Finalmente muestre en pantalla la suma de todos los números del array que sean primos.

7. Escribir un programa que cree una base de datos temporal sobre el personal de agentes de policía. La base de datos almacenará cuatro datos acerca de cada persona:

   - Nombre (Array de caracteres)
   - Apellido (Array de caracteres)
   - Código (Entero)
   - Categoría (Entero: 0: detective, 1: ayudante, 2: director)

   El programa preguntará acerca de cuántos datos se añadirán a la base de datos y luego mostrará los datos de todos los agentes introducidos.

8. Programa que implemente una agenda.

   Se guardarán los siguientes datos por persona:

   - *Nombre:* Máximo 15 caracteres.
   - *Apellidos:* Máximo 35 caracteres.
   - *Sobrenombre:* Máximo 10 caracteres.
   - *Teléfono:* Máximo 10 caracteres.
   - *Fecha de nacimiento:* 8 caracteres (con el formato dd/mm/aa, pudiendo ser espacios los dígitos del año

   Se creará un menú con las siguientes opciones:

   1. *Alta* de una nueva persona a la agenda con los correspondientes datos.
   2. *Eliminar* a una persona de la agenda.
   3. *Búsqueda* de un nombre en particular (La búsqueda se hará por sobrenombre)
   4. *Listado* de todas las personas empezando por la primera introducida (Para pasar a la siguiente se deberá pulsar intro).
   5. *Guardar* agenda en disco.
   0. *Salir*

9. Se deberá escribir un programa que permita introducir y consultar la tabla periódica de los elementos químicos. Para ello, se construirá un menú con las siguientes opciones:

   ```
   1. Introducir elementos de la tabla periódica
   2. Listar todos los elementos de tabla periódica
   3. Mostrar elemento de la tabla periódica por número
      atómico
   4. Mostrar elemento de la tabla periódica por símbolo
   5. Salir
   ```

La **opción 1** preguntará cuántos elementos de la tabla periódica se quieren introducir e irá preguntando sucesivamente por ellos.

Evidentemente, no es necesario rellenar el array con todos los elementos de la tabla periódica.

La **opción 2** listará la información acerca de todos los elementos químicos introducidos ordenados por número atómico.

La **opción 3** pedirá el número atómico del elemento que se quiere consultar y mostrará por pantalla la información correspondiente a dicho elemento.

La **opción 4** pedirá el símbolo del elemento que se quiere consultar y mostrará por pantalla la información correspondiente a dicho elemento.

Se guardará la siguiente información sobre cada elemento.

```
1. Símbolo del elemento
2. Nombre completo
3. Peso Atómico
```

El número atómico vendrá indicado por la posición en el array de elementos que representa la tabla periódica, y por tanto no será necesario almacenar esta información en el registro correspondiente.

Nota: Se puede mejorar el programa añadiendo una opción que permita guardar los datos en un fichero de disco.

Ficheros

1. Crear un programa que abra un fichero y escriba números enteros y otro programa calcule el valor máximo (si hay varios basta con uno de ello) y la media de todos los números contenidos en el fichero anterior:

   Ayuda: Utilícense las funciones *fprintf* y *fscanf*.

2. Leer completamente un fichero de texto, carácter a carácter (o en cantidades mayores, para que sea más rápido). El programa debe contar las vocales, los caracteres alfabéticos y los dígitos que hay en el fichero.

3. Crear un programa que abra un fichero y escriba en él dos cadenas, cada una acabada con el carácter de nueva línea y otro programa que lea la segunda cadena escrita en el mismo fichero.

   Comprobar que el fichero existe y visualizar su contenido con un editor de textos (p.e. el *Block de Notas* de Windows)

   Nota: Escribir una cadena con espacios intercalados.

   Ayuda: Utilícense las funciones *fputs* y *fgets*. Consultar la ayuda.

4. Escribir un programa que tome caracteres de teclado y, de uno en uno, los escriba en un fichero cuyo nombre es previamente pedido por pantalla.

5. Escribir un programa que saque por pantalla el contenido de un fichero cuyo nombre es pedido por pantalla.

   Nota: El nombre del fichero debe proporcionarse mediante la línea de argumentos al llamar al fichero. Es decir:

   ```
   C:> MostrFic fichero.txt
   ```

6. Escribir un programa que comprueba que un fichero de código contiene el mismo número de "{" que de "}" en su código. En caso de que no sea así el programa mostrará un mensaje indicando que el número de "{" es distinto que el número de "}".

7. Escribir un programa que use dos ficheros: uno de lectura y otro de escritura. El programa leerá los caracteres de un fichero, y tras una operación de cambio de mayúsculas a minúsculas y viceversa, escogida por el usuario, los escribirá en un segundo fichero.

8. Escribir el código necesario en el programa de la agenda de la lección de estructuras (Ejercicio 8) para que se añadan cuatro opciones más. De esta forma, el menú quedará de la siguiente manera:

1. *Alta* de una nueva persona a la agenda con los correspondientes datos.
2. *Eliminar* a una persona de la agenda.
3. *Búsqueda* de un nombre en particular (La búsqueda se hará por sobrenombre)
4. *Listado* de todas las personas empezando por la primera introducida (Para pasar a la siguiente se deberá pulsar intro).
5. *Leer* la agenda desde el disco (formato binario). El nombre del fichero será *agenda.bin*.
6. *Guardar* la agenda en el disco (formato binario). El nombre del fichero será *agenda.bin*.
7. *Guardar* la agenda en el disco para imprimir (en modo texto). El nombre del fichero será *agenda.txt*.
8. *Guardar* en un fichero de texto información de las personas que cumplen años en un determinado mes (cuyo número se leerá desde teclado). El nombre del fichero será el número del mes seguido de la extensión txt.

   Se guardará la fecha de nacimiento, el sobrenombre y el teléfono, ocupando la información de cada persona una línea. La información deberá estar encolumnada debajo de una cabecera, tal como se muestra en el ejemplo:

```
FECHA DE NACIMIENTO     SOBRENOMBRE          TELEFONO
13/12/85                Koldo                946478383
09/12/90                Ana                  653765432
```

0. *Salir*

9. Escribir un programa que permita visualizar el contenido de un fichero pasado desde la línea de comando en formatos hexadecimal y carácter, siguiendo el esquema del ejemplo:

```
C:> Ver Fichero.txt
48 6F 6C 61 2C 20 73 6F-79 20 75 6E 61 20 63 61   Hola, soy una ca
64 65 6E 61 0D 0A 59 20-79 6F 20 6F 74 72 61 0D   dena..Y yo otra.
0A EB 11 80 3E AF D2 00-74 03 E8 B2 E8 E8 FC E6   ....>...t.......
```

   Ayuda: Para averiguar si un carácter se puede imprimir existe la función *isprint()*.

10. Escribir un programa que pida el nombre de un fichero y lo borre de disco.

    Ayuda: Usar la función *remove()*.

11. Crear una base de datos almacenada en un fichero para personal universitario. Cada elemento de la base de datos constará de 3 campos: Nombre, apellido y edad.

    Adicionalmente, será necesario crear otro programa que lea los registros de la base de datos y los muestre secuencialmente por pantalla.

12. Escribir un programa que haga una conversión de un tipo de formato de fichero a otro. El formato de partida será el siguiente:

Posición de los caracteres	***Campo***
1-9	Matricula
9-14	Cilindrada
14-17	Potencia
17-28	Modelo
28-38	Marca

Ej: `BI6755CC1400 75FIESTA    FORD`

y el formato de destino serán las correspondientes estructuras, es decir:

Matrícula	8 caracteres
Cilindrada	entero
Potencia	entero
Modelo	10 caracteres
Marca	10 caracteres

Nota: Los datos de partida pueden introducirse con el programa 2.

13. Una entidad bancaria posee en un fichero la siguiente información para un conjunto de personas:

    - Nombre.
    - Número de cuenta.
    - Crédito solicitado.

    El programa funcionará basándose en un menú con las siguientes operaciones:

    1.- Añadir un crédito al archivo
    2.- Visualizar en pantalla la información correspondiente al crédito mayor
    3.- Copiará en otro fichero el conjunto de personas que hayan solicitado un crédito superior a cierta cantidad establecida por el banco (la cual deberá ser introducida por teclado).
    4.- Visualizar todo el contenido de un archivo

14. Se necesita construir un programa que a partir de la fórmula de un compuesto químico (supuestamente puro) y su peso en gramos obtenga la cantidad que dicho compuesto contiene de cada uno de los elementos químicos que lo forman.

    Para ello se escribirán dos programas:

    El **primero** de ellos escribirá en disco la tabla periódica. Por cada elemento de la tabla periódica se guardará la siguiente información:

    - Símbolo del elemento
    - Nombre completo
    - Peso Atómico

El número atómico de cada elemento vendrá representado por su posición en la tabla periódica.

El **segundo** de ellos leerá el fichero con la tabla periódica introducido con el programa anterior y presentará un menú con las siguientes opciones.

```
1. Listar tabla periódica
2. Mostrar elemento de la tabla periódica
3. Obtener la composición de un compuesto químico
0. Salir
```

La **opción 1** mostrará un listado con todos los elementos de la tabla periódica, parando la ejecución cuando se llene una pantalla para permitir verlos todos.

La **opción 2** pedirá el símbolo de un elemento químico, lo buscará en la tabla periódica y mostrará toda la información relativa. Es decir, su número atómico, el símbolo del mismo, su nombre completo y su peso atómico.

La **opción 3** permitirá introducir la fórmula del compuesto químico. Para ello irá preguntará primero por el número de átomos constitutivos y luego por el nombre y número de átomos del elemento en el compuesto. Posteriormente, se pedirá el peso en gramos del compuesto a analizar.
Una vez calculada con la tabla periódica, a partir de las proporciones obtenidas de la fórmula, la cantidad en gramos de cada uno de los elementos se mostrará el informe con la composición en gramos del compuesto.

Finalmente, la **opción 4** permite salir del programa.