UNIVERSIDAD DE MÁLAGA
DPTO. DE LENGUAJES Y CC. DE LA COMPUTACIÓN
E.T.S. DE INGENIERÍA INFORMÁTICA

Práctica Nº 6
Registros y arrays

Ejercicios de clase.

1. Desarrolla una función de nombre `mayor` que tome como parámetro de entrada un array totalmente relleno de valores reales y devuelva como resultado el valor mayor contenido en dicho array. Como tamaño del array puede tomarse el valor 10 (define una constante para ello). Crea también una función `main` para comprobar el correcto funcionamiento de la función diseñada.

2. Diseña una función `media` para calcular la media de las estaturas de una clase. La función recibe como parámetro de entrada una estructura de tipo `TEstaturas`, con las estaturas (en centímetros) de los alumnos de una determinada clase. Define dicho tipo de forma que pueda almacenar hasta un máximo de `MAX = 50` estaturas. El tipo debe definirse de forma que se pueda controlar en todo momento la cantidad de estaturas almacenadas. Crea después otras dos funciones para determinar cuántos alumnos son más altos y cuántos más bajos que la media. El nombre de las funciones queda a criterio del alumno.

   Crea también una función `main` para comprobar que las tres funciones se han codificado correctamente.

3. Diseña un algoritmo que lea de teclado una secuencia de números entre 0 y 9 y cuente el número de veces que se repite cada dígito. La secuencia de números de entrada se da por finalizada al leer un número negativo. Ej.:

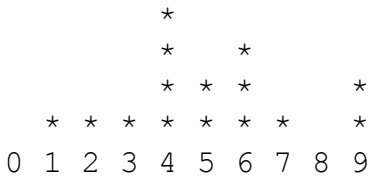
   Entrada: 1 8 7 3 4 8 5 9 5 0 0 4 8 4 5 3 2 8 -1
   Salida: 0: 2; 1:1; 2: 1; 3: 2; 4: 3; 5: 3; 6: 0; 7: 1; 8: 4; 9: 1

Ejercicios de refuerzo.

4. Un histograma es una gráfica que muestra la frecuencia con que aparecen en una lista dada los distintos valores que la pudieran formar. Por ejemplo, si los valores de una lista pueden estar comprendidos entre 0 y 9, y la lista es:

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

   entonces su histograma es:



Esto indica que los dígitos 0 y 8 no aparecen ninguna vez, que 1, 2, 3 y 7 aparecen una vez, 5 y 9 dos veces, etc. Escribe un programa que lea una lista de números comprendidos entre 0 y 9 separados por espacios (la lista acabará cuando se lea un número negativo, y a priori no se puede determinar cuántos números contiene) e imprima por pantalla un histograma como el anterior. Para ello deben usarse funciones y arrays.

5. La denominada *Criba de Eratóstenes* es un método para determinar los números primos entre 1 y N, siguiendo los siguientes pasos:
   - Se escriben los números naturales entre 1 y N.
   - Se tacha el 1.
   - Se deja el 2 y se tachan todos los demás números pares.
   - Se deja el 3 y se tachan todos sus múltiplos.
   - Como el 4 ya está tachado, pasamos al 5, que se deja y se tachan todos sus múltiplos (los del 5).
   - Etc.

   Así, cuando pasemos del 13, estarán tachados 14, 15 y 16, con lo que seguimos el proceso en el 17. El proceso acaba cuando llegamos a la raíz cuadrada de N. Los números que queden sin tachar, serán primos. La siguiente figura muestra este método aplicado del 1 al 100. Aparecen en blanco los números no marcados y que, por tanto, son primos.

   ~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ ~~25~~ ~~26~~ ~~27~~ ~~28~~
   29 ~~30~~ 31 ~~32~~ ~~33~~ ~~34~~ ~~35~~ ~~36~~ 37 ~~38~~ ~~39~~ ~~40~~ 41 ~~42~~ 43 ~~44~~ ~~45~~ ~~46~~ 47 ~~48~~ ~~49~~ ~~50~~ ~~51~~ ~~52~~
   53 ~~54~~ ~~55~~ ~~56~~ ~~57~~ ~~58~~ 59 ~~60~~ 61 ~~62~~ ~~63~~ ~~64~~ ~~65~~ ~~66~~ 67 ~~68~~ ~~69~~ ~~70~~ 71 ~~72~~ 73 ~~74~~ ~~75~~ ~~76~~
   ~~77~~ ~~78~~ 79 ~~80~~ ~~81~~ ~~82~~ 83 ~~84~~ ~~85~~ ~~86~~ ~~87~~ ~~88~~ 89 ~~90~~ ~~91~~ ~~92~~ ~~93~~ ~~94~~ ~~95~~ ~~96~~ 97 ~~98~~ ~~99~~
   ~~100~~

   Se pide crear un procedimiento denominado `eratostenes` que, mediante la criba descrita y haciendo uso de arrays, debe tomar como parámetro un natural N (menor o igual que una constante dada MAXIMO) e imprimir por pantalla todos los números primos del 1 al N.

   Crea también una función `main` para comprobar que el procedimiento se ha codificado correctamente

6. Para realizar operaciones con números complejos, podemos definir el siguiente tipo:

   ```
   struct TComplejo {
       double p_real, p_imaginaria;
   }
   ```

   Escribe procedimientos que realicen las operaciones de suma, resta, multiplicación y división de números complejos definidos con el tipo anterior:

   ```
   void sumar(TComplejo &resultado, const TComplejo &a, const TComplejo &b)
   void restar(TComplejo &resultado, const TComplejo &a, const TComplejo &b)
   void multiplicar(TComplejo &resultado, const TComplejo &a, const TComplejo &b)
   void dividir(TComplejo &resultado, const TComplejo &a, const TComplejo &b)
   ```

   Crea asimismo una función `main` que permita comprobar el correcto funcionamiento de los mismos.

7. Realizar los ejercicios 11 y 12 de la Tercera relación de problemas.