FUNDAMENTOS DE INFORMÁTICA
TEMA 6: DATOS ESTRUCTURADOS
EJERCICIOS

Los siguientes ejercicios deben programarse utilizando como mínimo una función , aparte del `main()`. En general, las funciones no tendrán que leer ni escribir valores, sino que analizarán los argumentos de entrada y producirán la salida correspondiente. Los programas no deberán declarar variables globales.

Nivel 1 : Ejercicios Fáciles

1. Indique si las siguientes afirmaciones son verdaderas o falsas:
   - Los elementos de un mismo array pueden ser de distintos tipos.
   - El nombre de cualquier array es un puntero al primer elemento (sin importar su tipo).
   - Una estructura nunca puede contener otra estructura.
   - Las siguientes expresiones son equivalentes: `a = ++b[0]` $\equiv$ `a = (*b)++`.
   - De la siguiente expresión `a[i].x` se puede deducir que `a` es un array de estructuras.

2. Escriba un programa que lea 20 enteros, los guarde en un array y calcule y muestre por pantalla la suma de los valores que haya en posiciones pares del array (0, 2, ...) y la suma de los valores que haya en posiciones impares del array (1, 3, ...).

   Modifique el programa para que el número de valores no sea fijo sino que se lea como entrada y sea como máximo 20.

3. Escriba un programa que sume dos vectores de tipo base `int` e imprima el resultado en pantalla. Se considerarán datos de entrada la dimensión de los vectores y las componentes de cada uno. Suponemos que los vectores tendrán un máximo de 10 componentes.

   Ejecute el programa para un caso en que cada vector tenga más de 10 componentes. ¿Qué ocurre?

4. Escriba un programa que calcule la desviación típica de un conjunto de valores enteros guardados en un array. En primer lugar el programa deberá pedir el número de valores (como máximo 20) y cada uno de ellos. Posteriormente calculará la desviación típica utilizando la fórmula:

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

   siendo N el número de elementos y $\overline{x}$ la media de todos ellos.

5. Lea un texto de teclado e imprima en pantalla un mensaje indicando cuántas veces aparecen dos letras contiguas e iguales en el mismo.

6. Implementar una función, llamada `SumaASCII` que acepte una cadena de caracteres y devuelva la suma de los códigos ASCII de sus caracteres. Implementar un programa de prueba de la función anterior que pida una cadena y muestre el número resultante. Para ello, no es necesario asignar el valor a una variable previamente, ya que se puede hacer directamente en la función de escritura. Por ejemplo:

```
printf("%u", SumaASCII(cadena))
```

7. Modificar el programa anterior para que admita un segundo argumento de tipo entero (que funcione como una variable lógica) que indique si se deben tener en cuenta o no los espacios en blanco. Es decir, si el argumento *lógico* vale 1 (*true*), los espacios serán tenidos en cuenta y en el valor resultante también se sumará el valor ASCII de los espacios que aparezcan. Por el contrario, si vale 0 (*false*) los espacios no serán tenidos en cuenta y serán obviados en la suma final.

8. Escribir la función `Reemplazar` que acepte 3 argumentos: una cadena de caracteres `CAD` y dos caracteres `A` y `B`. La función sustituirá todas las ocurrencias del carácter `A` en la cadena `CAD` por el carácter `B`. La función devolverá el número de caracteres sustituidos. Ejemplo: si `CAD` contiene la cadena "felicidad", tras la llamada `Reemplazar(CAD, 'i', 'x')`, la variable `CAD` tendrá la cadena "felxcxdad" y la función devolverá el valor 2.

9. Cuánto ocuparía en memoria la siguiente declaración de variables:

```
typedef char cadena[15];
typedef struct { cadena nombre;
                 cadena barco;
                 int siglo;
                 char categoria;
               } pirata;
pirata locos[10];
```

   Realice un pequeño fragmento de código que escriba por pantalla solamente la categoría y el siglo del pirata que dirige el barco "Temido".

10. Supongamos la siguiente declaración:

```
typedef char cadena[10];
typedef struct { cadena nombre;
                 cadena profes[2];
                 int categoria;
                 char dia; /* días válidos son 'L', 'M', 'X', */
               } clientes;           /* 'J', 'V', 'S', 'D' */
typedef clientes empresa[5];
empresa mia;
```

    (a) Realice una función a la que se le pase por valor la variable `mia` y escriba en pantalla todas las cadenas de caracteres que contiene.

    (b) ¿Cuántos caracteres pueden ser almacenados en la variable `mia` ?

(c) Realice una función con un único argumento de tipo `clientes` y que escriba, en letra (Lunes, Martes, ...), el día de la semana almacenado en el campo `dia`.

Nivel 2 : Dificultad Media

11. Escriba un programa que lea 20 números entre 10 y 100 y posteriormente imprima de los números leídos sólo aquellos que no sean el doble de otro leído anteriormente.

12. Las distancias en Kilómetros entre una serie de estaciones de tren son:

```
2.25          3.25          4.5
```

Escriba un programa que lea los datos anteriores y dé como resultado una tabla triangular con la distancia entre cualquier par de estaciones. Ejemplo:

```
     1     2     3     4
1  0.00
2  2.25  0.00
3  5.50  3.25  0.00
4  1.00  7.75  4.50  0.00
```

13. Generalizar el ejercicio anterior suponiendo que no sabemos el número N de estaciones (máximo 20). El programa pedirá primero el valor de N, luego pedirá las $N-1$ distancias entre cada dos estaciones consecutivas y, por último, mostrará los resultados.

14. Hacer un subprograma para descubrir el número de veces que aparece cada dígito entre 0 y 9 en un determinado número natural. El subprograma tendrá dos argumentos. El primero será de tipo `unsigned long int` y el segundo será de tipo `DIGITOS`, siendo éste un tipo definido por el programador, como un array de 10 `unsigned int`.

    El subprograma tomará el número del primer argumento, lo analizará y devolverá en el array del segundo argumento el número de veces que se repite cada dígito entre 0 y 9 en el número. Por ejemplo, para el número 248282, devolverá todo el array a cero, excepto la posición 2 que tendrá un 3, la posición 4 que tendrá un 1 y la posición 8 que tendrá un 2.

    El programa principal leerá un número por teclado, llamará al subprograma anterior y mostrará en pantalla el número de veces que se repite cada dígito entre 0 y 9 en el número dado, evitando mostrar los dígitos que no existen en el número original. Al final, mostrará el número de dígitos total del número introducido.

    Por ejemplo, para el ejemplo anterior del número 248282 el programa deberá mostrar las siguientes líneas:

```
Número de dígitos 2: 3
Número de dígitos 4: 1
Número de dígitos 8: 2
Número total de dígitos: 6
```


15. En un almacén, cada producto es identificado por un número al que se le añade un dígito de autoverificación. Dicho dígito de autoverificación se calculará de la siguiente forma:

    1.- Multiplicar la posición de las unidades y cada posición alternada por dos. Ejemplo: Si el número del producto es 543211 obtenemos 583412.

    2.- Sumar los dígitos no multiplicados y los resultados de los productos obtenidos en el apartado 1. En el ejemplo obtenemos 23.

    3.- Restar el número obtenido en el apartado 2 del número más próximo y superior a éste, que termine en cero. En el ejemplo sería 30–23=7.

    El resultado será el dígito de autoverificación.

    Codificar un programa que vaya aceptando números de productos y compruebe mediante el dígito de autoverificación si el número introducido es correcto o no. El proceso se repetirá hasta que se introduzca un cero como número de producto.

16. Declarar un tipo `MATRIZ` que sirva para almacenar matrices de números reales (usando un array bidimensional), de tamaño NxN, donde N será una constante. Implementar lo siguiente:

    (a) Escribir una función para hallar la matriz traspuesta de una matriz de ese tipo.

    (b) Escribir una función llamada `simetrica`, con un único argumento de tipo `MATRIZ`, que devuelva 1 si dicha matriz es simétrica y 0 si no lo es. Para resolver esta función se debe usar la función creada en el apartado anterior.

    Escriba un programa que tome de entrada todos los datos de una matriz, calcule su traspuesta, la muestre por pantalla e informe si la matriz de entrada era o no simétrica.

17. Escriba un programa que efectúe la conversión de un número entero en base 10 a cualquier base. La introducción del número se hará en formato "número/base". Ejemplo:

    `Introduzca dato: 723/4`. (indica que el número 723 hay que convertirlo a base 4).

18. Escriba un programa que convierta un número entero expresado en cualquier base natural entre 2 y 10 a una base natural entre 2 y 10. Para ello la entrada se leerá según el formato: "BaseInicio/Numero/BaseDestino". Ejemplo:

    `Introduzca dato: 4/123/8`. (Indica que debe pasar el número 123 que está expresado en base 4 a base 8).

19. Escriba un programa que simule el comportamiento de una calculadora simple para aritmética fraccional. El resultado se expresará también en forma fraccional y simplificado al máximo. Supondremos que tanto el numerador como el denominador son números sin decimales. Las fracciones se leerán en formato Numerador/Denominador. Ejemplo: `34/7`

20. Programar una función de nombre `InsertaCad`. Esta función aceptará 3 argumentos: Dos cadenas de caracteres `C1` y `C2` y un número natural `Pos`. El procedimiento insertará la cadena `C2` en la posición `Pos` de la cadena `C1`. El primer carácter de la cadena `C1` se

considerará el 1. Si `Pos` es mayor que la longitud de `C1`, entonces `C2` se insertará al final de la cadena `C1`. Si en `C1` no cabe el resultado final, se truncará.

Ejemplo: Si `C1` es un array de 50 caracteres y vale `"La felicidad"`, tras la llamada

```
InsertaCad(C1,"sabiduría genera ",4);
```

la cadena `C1` tomará el valor `"La sabiduría genera felicidad"`.

El programa principal declarará cadenas de caracteres de longitud máxima `MAXLONG`, siendo ese valor una constante. El programa leerá dos cadenas y una posición. Posteriormente insertará la segunda cadena en la primera en dicha posición e insertará la primera cadena en la segunda en la misma posición, mostrando las dos cadenas obtenidas. Las dos operaciones de inserción serán efectuadas directamente sobre las dos cadenas leídas originalmente y no se efectuará la segunda inserción sobre los resultados de la primera inserción.

21. Implementar una función que devolverá un dato de tipo `int` indicando si una cadena de caracteres, que se le pasa como único argumento, es o no un **palíndromo**. Esto lo indicará devolviendo un 1 si es palíndromo y 0 si no lo es. Una palabra o frase se dice que es un palíndromo si, ignorando los espacios, la lista de letras es idéntica hacia adelante o hacia atrás. Por ejemplo, si la frase de entrada es "Dabale arroz a la zorra el abad", el programa debe indicar que SÍ es un palíndromo.

    Sería ideal que el programa considerara como iguales las letras acentuadas con las no acentuadas, y las letras mayúsculas de las minúsculas (incluida la ñ), así como que no considerase tampoco los signos de puntuación (comas, puntos...).

    Ejemplos de otras frases palindrómicas son: "se verla al revés", "ella te dará detalle", "A la Manuela dale una mala", "a la torre, derrótala", "ligar es ser ágil", "logré ver gol", "luz azul", "ojo rojo"...

22. Implementar una función que devolverá un dato de tipo cadena que corresponderá con las siglas de la cadena que toma como argumento. Las siglas estarán formadas por las letras en mayúsculas de todas las palabras, separadas por un punto. Ejemplo: Si la frase del argumento es "Tren articulado ligero, Goicoechea Oriol", la frase que devolverá será "T.A.L.G.O.".

23. Supongamos que deseamos evaluar a un determinado número de alumnos siguiendo el criterio de que aprobará aquel que supere la nota media de la clase.

    Escriba un programa que lea un número indeterminado de alumnos (como máximo 20, y las notas de tres evaluaciones, y como resultado emita un informe indicando para cada alumno las evaluaciones que tiene aprobadas y suspensas. Ejemplo de la salida que se debe obtener.

```
Alumno        Nota-1      Nota-2      Nota-3
Juan Lopez    Aprobado    Suspenso    Aprobado
Luis Garcia   Suspenso    Aprobado    Aprobado
Pedro Ruiz    Aprobado    Aprobado    Aprobado
```


24. Escriba una función que elimine todas las veces que aparece un determinado carácter en una cadena de caracteres. La función deberá tener tres argumentos, la cadena de entrada, el carácter que se va a eliminar, y la cadena de salida. Por ejemplo, si la cadena de entrada contiene `"Estamos en el 2002"` y el carácter a eliminar es la `'s'`, la cadena resultante deberá contener `"Etamo en el 2002"`.

25. Implemente una función que simule el comportamiento de la función `strpbrk()` de C. Esta función tiene como argumentos dos cadenas de caracteres y devuelve un carácter. Este carácter es el primer carácter de la primera cadena que aparece en la segunda. Si no se repite ningún carácter devuelve `'\0'`.

    Ej: La llamada `strpbrk("buenos días tío","temporización")` devuelve `'e'`.

26. Escriba un programa que pida N (en el ejemplo 4) cadenas de M (en el ejemplo 5) letras (['a'..'z']). Implemente una función que, dada esa matriz de letras de $N \times M$ devuelva otra matriz de cardinales de las mismas dimensiones ($N \times M$), donde en cada posición se indicará cuantos caracteres iguales hay rodeando al que se encuentra en esa posición de la matriz original:

$$\left|\begin{array}{ccccc} a & b & c & c & h \\ b & b & b & c & k \\ j & b & c & k & k \\ q & w & e & r & k \end{array}\right| \Longrightarrow \left|\begin{array}{ccccc} 0 & 3 & 2 & 2 & 0 \\ 3 & 4 & 3 & 3 & 2 \\ 0 & 3 & 1 & 3 & 3 \\ 0 & 0 & 0 & 0 & 2 \end{array}\right|$$

27. Dada la siguiente declaración:

```
struct coche {
  int puertas;
  char color[15];
  char matricula[10];
};
struct coche garage[100];
```

    (a) Escriba una función que compruebe si un coche tiene una letra concreta en su matricula. Devolverá un 1 si la contiene y un 0 si no contiene esa letra. El prototipo de esa función sería el siguiente:

```
int letra_matricula(struct coche C, char letra);
```

    (b) Escriba una función que devuelva el color que más se repite y el número de coches que tienen ese color. La cabecera de la función puede ser la siguiente:

```
void Color(struct coche G[], int N, int *num_color, char color[]);
```

28. Supongamos que tenemos una compañía de discos y queremos hacer un programa que almacene en un array los datos de hasta 200 discos: código (único para cada disco), autor, título, año, número de canciones...

    - Declare una estructura de datos que nos permita almacenar la información descrita.

- Realice una función para recuperar toda la información de un disco, sabiendo su código: Se trata de recuperar esa información en una variable y no de escribirla, por lo que la función devolverá, de alguna forma, dicha información.
- Realice una función para insertar un nuevo disco en el array, sabiendo previamente sus datos: La función no leerá los datos, por lo que habrá que facilitárselos.
- Implemente una función para copiar en otro array similar todos los discos existentes de un determinado autor.

29. Dadas las siguientes declaraciones:

```
struct fecha{
    int dia, mes, anio;
};
struct persona{
    char nombre[50];
    struct fecha fecha_nac;
    double altura;
    double peso;
};
```

(a) Implemente una función `escribir()` que muestre por pantalla todos los datos de todas las personas que hay almacenadas en un array de `struct persona`. La función debe tener dos argumentos, el primero un array de `struct persona` y el segundo un entero que indique el número de datos que hay en el array. La función no devuelve nada.

(b) Escriba una función `leer()` para pedir todos los datos de una persona concreta. La función no tendrá argumentos y devolverá un `struct persona` con los datos leídos.

(c) Suponga que tenemos la siguiente declaración de variables:

```
struct persona Equipo[100];
```

Indique si las siguientes llamadas a la función implementada anteriormente son correctas o no y por qué:

```
A) Equipo=leer();        C) leer(Equipo[100]);   E) leer();
B) leer(Equipo);         D) Equipo[100]=leer();  F) Equipo[10]=leer();
```

(d) Escriba una función para buscar los datos de una persona en el array. La función buscará por el nombre de la persona y devolverá todos los datos y un valor entero (1 si ha encontrado a la persona y 0 si no lo ha encontrado).

```
int buscar(struct persona P[], int tamanno, char nombre[], struct
persona *pers);
```

(e) Escriba una función en la que se utilice la función anterior. Declare las variables que sean necesarias, pida el nombre de la persona y llame adecuadamente a la función.

Nivel 3 : Dificultad Avanzada

30. Escriba un programa que lea por teclado dos matrices ($M \times N$) y ($N \times P$) y devuelva como resultado la matriz producto ($M \times P$). Los valores M, N y P se introducirán por teclado antes de la lectura de las matrices. El tamaño máximo de M, N y P será una constante definida en el programa. Nota: Son 3 bucles anidados de M, P y N iteraciones respectivamente.

31. Se desea efectuar operaciones de suma de cantidades que se caracterizan por tener una gran cantidad de dígitos, de manera que queda excluido el uso del tipo `float` o `double`. Escriba un programa que lea dos números, los sume e imprima el resultado en pantalla. Ejemplo:

```
Número 1:     1111111111111111111111111111111111111111
Número 2:     1111111111111111111111111111111111111111
Resultado:    2222222222222222222222222222222222222222
```

Se considera que no se van a desear sumar números con más de 50 dígitos.

Los números a sumar no contienen decimales y se consideran positivos.

32. Escribir la función `EncuentraRepes` que acepte dos argumentos que serán sendas cadenas de caracteres como arrays. La función devolverá el número de veces que aparece la primera cadena en la segunda.

Ejemplo: La siguiente llamada devolverá el valor 3.

```
EncuentraRepes ("sol", "Cuando estaba solo, solamente vi el sol")
```

Escribir un programa de prueba para verificar el correcto funcionamiento de la función.

33. Los alumnos de una clase desean celebrar una cena de confraternización un día del próximo mes (que supondremos con 31 días), en el que puedan asistir la mayoría. Realizar un programa que recoja de cada alumno los días que le vendría bien para ir a la cena e imprima la fecha que prefiere la mayoría de los alumnos. Observaciones:

(a) Los datos se introducirán por teclado y cada alumno escribirá una única línea (cadena de caracteres) con los días separados por espacios en blanco: Ejemplo: "2 5 15 22 31".

(b) Comprobar que un alumno no repita el mismo día varias veces: En tal caso, dar un mensaje de error y repetir la lectura.

(c) Si el día resultante ha sido elegido por TODOS los alumnos, indicarlo con un mensaje.

(d) Si hay varios días con el máximo número de votos, mostrar todos esos días como posibles.

34. Implementar la función `SigCharPos`: Esta función acepta 3 parámetros y devuelve un `unsigned int`. El primer parámetro es una cadena de caracteres `Cad`, el segundo parámetro es un carácter `Ch` y el tercer parámetro un número natural `N`. La función devolverá la posición del siguiente carácter `Ch` en la cadena `Cad` empezando a mirar a

partir de la posición `N` (inclusive). Si el carácter no es encontrado, la función devolverá el valor más grande posible que admita el tipo que se devuelve. Ejemplos: Las llamadas siguientes devolverán los valores que se indican a la derecha:

```
SigCharPos("Aristóteles dijo que saber es acordarse", 't', 5) : 6
SigCharPos("Aristóteles dijo que saber es acordarse", 'A', 0) : 0
SigCharPos("Aristóteles dijo que saber es acordarse", 's', 0) : 3
SigCharPos("Aristóteles dijo que saber es acordarse", 's', 4) : 10
```

Nota: En la implementación de esta función sólo se pueden utilizar los subprogramas de `string.h` vistos en clase y, como en la mayoría de las funciones, no se debe ni leer ni escribir datos.

Implementar también un programa que con un menú con las siguientes opciones:

1. "Buscar siguiente carácter a partir de una posición".
2. "Buscar todas las ocurrencias de un carácter".
3. "Las dos opciones anteriores".
4. "Buscar todas las ocurrencias de un carácter a partir de una posición".
5. "Salir".

En la opción 1, pedirá los datos necesarios para la función y mostrará el resultado devuelto. En la opción 2, pedirá también los datos (excepto la posición) y mostrará las posiciones de todas las ocurrencias de un carácter en una cadena. Para esta última operación también se utilizará la función descrita. En la opción 3, pedirá también los datos y mostrará los resultados de la opción 1 y de la 2. En la opción 4, mostrará las posiciones de todas las ocurrencias de un carácter en una cadena a partir de una posición determinada. La última opción será para Salir del programa.

35. Un cuadrado mágico es una matriz cuadrada con un número impar de filas y columnas, cuyas filas y columnas (e incluso sus diagonales) suman el mismo valor. Por ejemplo la matriz siguiente es un cuadrado mágico de 3×3:

6	1	8
7	5	3
2	9	4

Fíjese que los números en cada fila, cada columna y cada diagonal suman 15.

Hacer un programa que compruebe si una matriz es un cuadrado mágico o no. El tamaño máximo de la matriz será de 20×20. En primer lugar el programa leerá el tamaño N de la matriz hasta que éste sea correcto (impar y menor que el tamaño máximo), y pedirá los datos de una matriz $N \times N$. Una vez leídos y almacenados los datos se comprobará si la matriz es un cuadrado mágico o no y se mostrará el mensaje correspondiente

Nota: Realice el ejercicio implementando distintas funciones: para leer una matriz, escribir una matriz y comprobar si una matriz es cuadrado mágico.

36. Supongamos que una cadena de televisión quiere un automatizar la gestión de su programación semanal, para lo que deberá almacenar todos los programas que se emiten durante la semana. Para cada programa se almacenarán los siguientes datos: Nombre del programa, Día de emisión, Hora de emisión, Duración, Género (Cine, Variedades, Informativo, Documental, Música, Serie), Media de audiencia semanal (millones de espectadores).

    (a) ¿Qué estructura de datos te parece la más adecuada para almacenar esta información? Escribe los tipos y variables que hayas pensado en C.

    (b) Escribe una función que lea todos los datos de un programa.

    (c) Escribe una función que saque un listado de los programas que se emiten cada día a las 10 de la noche.

37. Se dice que una matriz "pivota" sobre su diagonal si en la diagonal se encuentra el mayor valor de cada columna. Escribe un programa que lea una matriz de enteros y la modifique para que ésta pivote sobre su diagonal. En primer lugar el programa deberá pedir el número de filas y columnas (N), y los datos de la matriz $N \times N$. Posteriormente deberá intercambiar en cada columna el elemento mayor de la misma con el que haya en la diagonal. Finalmente se escribirá la matriz resultante. Ejemplo:

Matriz de entrada:			Matriz resultante:		
3	2	8	6	2	3
6	3	7	3	6	7
5	6	3	5	3	8

Nota: Realize el ejercicio usando funciones para leer, escribir y modificar la matriz.

38. Declare dos estructuras para almacenar la siguiente información en cada una de ellas:

    - **Vacuna** (3 campos): De una vacuna nos interesa el nombre de la enfermedad a la que se aplica (Hepatitis A, Tétano, Malaria...), la dosis (1, 2...) y la duración (en meses) que tiene su protección. Por ejemplo, la Hepatitis A en la primera dosis dura 6 meses pero si se inyecta una segunda dosis a los 6 meses, esa segunda dosis hace que la duración sea de 10 años.
    - **Pacientes** (3 campos): De cada paciente almacenamos su nombre, nombre de la vacuna suministrada (coincide con el nombre de la enfermedad a la que se aplica) y número de la última dosis de dicha vacuna (1, 2...).

    Supongamos que un programa principal maneja 2 arrays de estructuras correspondiendo cada uno a una de las 2 estructuras anteriores. Realice diferentes funciones independientes para las siguientes acciones:

    - Función `BuscaVacuna()`: La función imprimirá todos los datos de una vacuna en particular. El nombre de la vacuna pasará como un argumento más de la función, aparte del array de vacunas y su tamaño. Tenga en cuenta que una vacuna puede aparecer varias veces (para distintas dosis).
    - Función `PacienteVacuna()`: Dando como argumentos el nombre de un paciente y el nombre de una vacuna (además de los dos arrays que necesitamos y sus tamaños),

la función devolverá el número de meses de vigencia que tiene la vacuna para la dosis que tiene ese paciente. La función devolverá –1 si el paciente no tiene ninguna dosis de esa vacuna.

Nota: Lo mejor es primero buscar en el array de pacientes la dosis que dicho paciente tiene puesta y luego buscar la duración de esa dosis en el array vacunas. Supondremos que para cada paciente sólo se almacena la última dosis de cada vacuna.

39. Una cuenta bancaria está identificada por un código de cuenta (c/c) que es un número de 20 dígitos con el siguiente formato: EEEE OOOO DC NNNNNNNNNN

donde EEEE es el código de la Entidad (Banco o Caja de Ahorros), OOOO es el código de la Oficina a la que pertenece la cuenta dentro de esa Entidad, DC son dos Dígitos de Control de errores y los últimos 10 dígitos son el número de cuenta dentro de esa Oficina.

El primer dígito de control de errores (dígito D) se calcula a partir de los 8 dígitos del número EEEEOOOO (códigos de Entidad y Oficina). Para ello, se multiplica cada uno de esos dígitos respectivamente por los siguientes "pesos" numéricos y se suman esas multiplicaciones: 4, 8, 5, 10, 9, 7, 3 y 6. Con el número resultante se calcula el resto de dividirlo entre 11. Si ese resto es cero o uno el dígito D será ese resto y en otro caso el dígito D será el resultante de restarle ese resto al número 11. El segundo dígito de control (dígito C) se calcula a partir de los 10 dígitos del número de cuenta, exactamente de la misma forma pero variando los "pesos" de cada uno de esos dígitos, siendo respectivamente en este caso los siguientes: 1, 2, 4, 8, 5, 10, 9, 7, 3 y 6.

Ejemplo: Número de cuenta 2038 1900 13 60000 31738
· Dígito D: $8 + 0 + 15 + 80 + 9 + 63 + 0 + 0 = 175$.
$175 \% 11 = 10$. Como no es cero ni uno: D = 11 - 10 = 1.
· Dígito C: $6 + 0 + 0 + 0 + 0 + 0 + 30 + 9 + 49 + 9 + 48 = 151$.
$151 \% 11 = 8$. Como no es cero ni uno: C = 11 - 8 = 3.
Observe que el número DC es 13.

Implementar lo siguiente:

(a) Una función `ComprobarCta()` que acepte un único argumento que será una cuenta bancaria en una cadena de caracteres. La función devolverá los siguientes valores:
0 : Si los dígitos introducidos en la cuenta son correctos.
1 : Si el primer dígito es incorrecto (dígito D).
2 : Si el segundo dígito es incorrecto (dígito C).
3 : Si son incorrectos ambos dígitos.
–1 : Si la cuenta de la cadena de caracteres no es correcta.
Una cuenta no es correcta si tiene alguno de los siguientes defectos:
- No tiene un espacio entre los números de código de Entidad (EEEE), de Oficina (OOOO), Dígitos de Control (DC) y el número de cuenta.
- Falta o sobra algún dígito, en algún sitio.
- Hay caracteres en algún sitio que no son dígitos numéricos (letras...).

(b) La función principal deberá solicitar un número de cuenta y mostrar el resultado de la comprobación después de utilizar la función anterior.

Tenga en cuenta que dentro de la función del apartado (a) no se leerán ni escribirán valores. En la tabla ASCII los caracteres numéricos ('0', '1', '2'...) están ordenados.

40. Suponga que queremos trabajar con los datos de los empleados de una fábrica de chicles: Nombre completo, edad, DNI y sueldo (en euros). Supondremos que la fábrica tiene un máximo de 50 empleados.

- Defina una estructura de datos adecuada para almacenar dichos datos.
- Programe un subprograma para el que dada la estructura anterior con todos los empleados y un apellido concreto, muestre por pantalla TODOS los datos de los empleados que tienen ese apellido en su nombre completo (el apellido a buscar será una cadena de caracteres que habrá que localizar en el campo nombre de cada empleado).

Nota: Tenga en cuenta que puede haber menos de 50 empleados, por lo que deberá controlar ese caso de alguna forma y explicar esa forma. Realicelos subprogramas independientes que considere oportunos.

41. Cree un array de estructuras con los datos de 10 alumnos, almacenando la siguiente información de cada alumno: Nota de la primera convocatoria (Julio) y de la segunda convocatoria (Septiembre) y sexo (M y F). El programa deberá mostrar la nota media y la varianza tanto de la primera convocatoria como de la segunda convocatoria de los siguientes grupos de datos (3 datos por cada convocatoria):

- De todos los alumnos.
- De los alumnos masculinos.
- De los alumnos femeninos.

Tenga en cuenta que si un alumno aprueba (nota$\geq$5) en la primera convocatoria, sus datos no deben considerarse al calcular las estadísticas de la segunda convocatoria.

Supondremos que, si un alumno no se presenta a una convocatoria su nota en dicha convocatoria será negativa o superior a 10. Por supuesto, la nota media y la varianza se calcularán con respecto a los alumnos que se han presentado al examen (con nota entre 0 y 10, ambos incluidos).

42. Modifique el ejercicio anterior para mostrar, además, el porcentaje de alumnos presentados a examen con respecto al total de alumnos convocados en cada convocatoria. Debe también tener en cuenta que si un alumno aprueba en la primera convocatoria, ese alumno no es convocado en la segunda.

Los porcentajes se mostrarán también por grupos (5 datos por cada convocatoria):

1. Porcentaje global de alumnos presentados.
2. Porcentaje de hombres presentados con respecto al total y con respecto al total de hombres convocados.
3. Porcentaje de mujeres presentadas con respecto al total y con respecto al total de mujeres convocadas.

43. Modifique el ejercicio anterior para mostrar, además el porcentaje de alumnos aprobados en cada convocatoria con respecto al total de alumnos convocados en cada convocatoria y con respecto al total de alumnos presentados en cada convocatoria. Debe también

tener en cuenta que si un alumno aprueba en la primera convocatoria, ese alumno no es convocado en la segunda.

Los porcentajes se mostrarán también por grupos (10 datos por cada convocatoria):

1. Porcentaje global de alumnos aprobados con respecto al total de convocados y presentados.
2. Porcentaje de hombres aprobados con respecto al total de alumnos (hombres o mujeres) convocados y presentados, y también con respecto al total de hombres convocados y presentados.
3. Porcentaje de mujeres aprobadas con respecto al total de alumnos (hombres o mujeres) convocados y presentados, y también con respecto al total de mujeres convocadas y presentadas.