FUNDAMENTOS DE INFORMÁTICA
TEMA 5: Funciones
Ejercicios

Como ejercicios de este tema puede programar los ejercicios de temas anteriores pero utilizando funciones. Al realizar las funciones intente que sean lo más generales posible, de tal modo que sirvan en cualquier sitio desde donde se las llame. Esto se puede conseguir teniendo en cuenta lo siguiente:

- Dentro de la función no se deben, en general, leer ni escribir valores. Los valores que se necesitan pueden venir por distintas vías (no sólo la lectura) y los valores de salida pueden ser usados de distintas formas (no sólo para escribirlos por pantalla).
- Defina bien los parámetros de entrada y salida dejando bien claro lo que necesita y devuelve la función.
- No use variables globales.

Nivel 1 : Ejercicios Fáciles

1. Considera la siguiente función:

```
void escr(char ch,int long) {
   while (long > 0) {
      putchar(ch);
      long - -;
   }
}
```

   (a) Si `ch` tiene el valor `'X'` y `numero` el valor 5, ¿cuál sería el efecto de ejecutar cada una de las siguientes llamadas a la función:

```
escr(ch,4*numero-12)      escr(ch,6)      escr(5,numero)
escr('/',numero)          escr('.',6)     escr('p',-10)
```

   (b) Escribe llamadas a la función `escr` para que cuando se ejecuten produzcan las siguientes salidas: 1) 35 guiones sucesivos; 2) 6 veces tantos espacios en blanco como el valor de `numero`; 3) el valor actual de `ch` 14 veces.

2. Considera el siguiente programa:

```
#include <stdio.h>
int i; /* Precaución: variable global */
/***********************/
void F1() {
  i=i+2;
}
```



```
/***********************/
void F2() {
  int i;
  i=0;
  F1();
  printf("%d\n",i);
}
/***********************/
void F3() {
  int i;
  i=4;
}
/***********************/
void F4() {
  i=i+3;
  printf("%d\n",i);
  F3();
  printf("%d\n",i);
}
/* FUNCIÓN PRINCIPAL */
void main() {
  i=12;
  F2();
  printf("%d ",i);
  F4();
  printf("%d",i);
}
```

   (a) Determina el ámbito de cada una de las tres variables `i` declaradas en el programa.

   (b) Muestra lo que ocurre cuando se ejecuta el programa.

   (c) ¿Qué ocurre si se declara la variable `i` del módulo principal justo antes de la función `main()`?

   Nota: La variable `i` que hay declarada después del `#include` es una variable global y se puede utilizar en todas las funciones del programa. Hay que evitar el uso de estas variables globales y a cambio usar siempre variables locales, es decir, usar en cada función las variables definidas dentro de la misma.

3. Seguir la ejecución del siguiente programa y averiguar lo que saldría por la pantalla. Después, compila el programa y ejecútalo en un ordenador para comprobar los resultados que has obtenido:

```
/* Programa de Prueba de los pasos de parámetros */
#include <stdio.h>

/* Subprograma con pasos de parámetros por valor
   y por referencia */
```

```
void dos(int x,int y,int *z) {
   *z = x + y + *z;
   printf("%d %d %d\n",x,y,*z);
}
void main() {
  int a,b,c;
  a = 5;
  b = 8;
  c = 3;
  dos(a,b,&c);
  dos(7,a+b+c,&a);
  dos(a*b,a / b,&c);
  printf("%d", c);
}
```

4. ¿ Qué salida produce la ejecución del siguiente programa?

```
#include <stdio.h>
int a,b,c,x,y; /* Precaución: variables globales */
/*************************************/
void Primero() {
  a = 3*a;
  c = c+4;
  printf("Primero: %d %d %d\n",a,b,c);
}
/*************************************/
void Segundo() {
  int b;
  b = 8;
  c = a+c+b / 3;
  printf("Segundo: %d %d %d\n", a,b,c);
}
/*************************************/
void Tercero(int *x, int y) {
  *x = *x+4;
  y = y+1;
  printf("Tercero: %d %d %d %d %d\n",a,b,c,*x,y);
}
/********* FUNCIÓN PRINCIPAL **********/
void main() {
  a = 3;  b = 2;  c = 1;
  x = 11; y = 22;
  Primero();
  Segundo();
  Tercero(&a,b);
  printf("Anidado: %d %d %d %d %d\n",a,b,c,x,y);
  }
```


5. Dadas las siguientes declaraciones en un determinado programa:

```
unsigned int a,b,c;
int si;
int uno(unsigned int x,unsigned int y)
......
void dos (unsigned int *x, unsigned int y)
......
unsigned int tres(unsigned int x)
```

¿ Cuáles de las siguientes llamadas son válidas?

```
a) if (uno(a,b))  ...        b) dos(&a,b+3);
c) si = uno(c,5);            d) si=dos(&c,5);
e) dos(&a,tres(a));          f) dos(tres(b),c);
g) if (tres(a)) ...          h) b=tres(dos(&a,5));
i) dos(4,c);                 j) si=uno(dos(a,5));
```

6. Escribe una función que tome 3 parámetros: dos de tipo natural (entero positivo) y uno de tipo carácter. La función deberá sumar, restar, multiplicar o dividir los valores de los dos primeros parámetros dependiendo del código indicado en el tercer parámetro, y devolver el resultado. El programa principal deberá pedir de forma repetida dos números y un operador y mostrar el resultado por pantalla.

7. Escribe un programa que lea 2 números enteros positivos y un carácter. En función de este carácter leído efectuará las siguientes operaciones:

   - 'p' : calcula los números perfectos en el rango dado por los 2 números.
   - 'm': calcula el mínimo común múltiplo de ambos números.
   - 'd' : calcula el máximo común divisor de los mismos.
   - '+', '*', '–', '/' : realiza la operación correspondiente con los números.

   Nota: Un número perfecto es aquel para el cual se cumple que la suma de sus divisores (excepto él mismo) es igual al número. Ej: El 6 es un número perfecto porque 6=1+2+3 (1, 2 y 3 son los divisores de 6).

8. Implementar lo siguiente:

   - Una función que, dados como argumentos los valores de 2 resistencias (en ohmios), devuelva la resistencia global que ofrecen ambas si se conectan en paralelo.
   - Un programa que calcule la resistencia global de n resistencias conectadas en paralelo. El programa pedirá sucesivamente una tras otra las n resistencias. El programa entenderá que no hay más resistencias cuando lea una resistencia menor o igual a cero. El programa se implementará de forma que utilice la función anterior.

9. Escribir una función que, dados (como argumentos por Valor) los valores de 3 resistencias (en ohmios), devuelva la resistencia global que ofrecen si se conectan en paralelo. Implementar esta función usando y sin usar la función del ejercicio anterior.

Nivel 2 : Dificultad Media

10. Codificar un programa que resuelva las ecuaciones simultáneas siguientes:

$$ax + by + c = 0 \qquad \text{y} \qquad px + qy + r = 0$$

Realizar un procedimiento encargado de la solución de las ecuaciones que devuelva el valor de x e y en función del resto de parámetros de las ecuaciones. El procedimiento debe también devolver una variable entera que sea 1 (*true*) en el caso de que las ecuaciones tengan solución y 0 (*false*) en caso contrario. El programa debe leer los valores, resolver las ecuaciones usando el procedimiento diseñado e imprimir en pantalla los valores en caso de existir o un mensaje de error si no es posible resolverlas.

Resolver con dicho módulo las ecuaciones siguientes:

```
3x + 2y - 7 = 0     3x + 2y -  7 = 0     3x + 2y -  7 = 0
9x - 5y + 1 = 0     9x + 6y - 21 = 0     9x + 6y - 20 = 0
```

11. Defínase una función llamada `SumaDeDigitos` que reciba como parámetro un número `unsigned int` y devuelva la suma de los dígitos que lo componen. Componga con ella un programa donde se lea el número, se calcule esta suma y se presente en pantalla el resultado.

Unos ejemplos de ejecución son: para la entrada 102 el resultado es 3 y para 1800 el resultado es 9.

12. Dattatreya Kaprekar (1905-1986) fue un matemático indio que nunca escribió nada de alto nivel y sus trabajos no han dado origen a ninguna investigación especial ni tienen ninguna utilidad importante, pero son sorprendentes y originales. Entre sus trabajos está el siguiente:

- Elegir un número de cuatro cifras A (no todas iguales y en el que las primeras pueden ser ceros).
- Ordenar las cifras del número A en orden creciente B y luego en orden decreciente C.
- Restar ambos números: D = C – B.
- Volver a empezar pero considerando ahora el número D (en vez del número A).

Lo curioso es que independientemente del número A elegido, al final siempre se llega al mismo número. Hacer un programa que averigue ese número final (6174).

Fases del programa: Primero hacer una función que dado el número A como único argumento, nos devuelva el número D. Esa función sólo efectuará un ciclo. El programa principal pedirá un número y aplicará la función sucesivamente hasta que el resultado de la función sea igual a su argumento, es decir, hasta que ya no tenga sentido seguir aplicando la función porque siempre obtenemos el mismo número. En ese momento el programa mostrará ese número y terminará.

13. Otra idea de Kaprekar es coger un número A de 5 cifras que dividimos en dos partes, una B con las 3 primeras cifras y otra C con las 2 últimas. Se suma 2 a ambos números B y C, despreciando el acarreo si se supera el número de cifras inicial de B y C. Invertid los dos números poniendo delante el de dos cifras, CB, obteniendo un nuevo número de 5 cifras. Ahora volver a empezar, repitiendo el proceso, hasta que obtengamos, de nuevo, el número inicial, contando el número de pasos que vamos dando.

Por ejemplo: A partir del número 45113, Kaprekar descubrió que se obtiene el número inicial en 24568 etapas: 45113, 15453, 55156, 58553...

14. Siguiendo con Kaprekar, descubrió que algunos números al elevarlos al cuadrado y partir el resultado en dos partes de 3 dígitos, si se suman esas dos partes se obtiene el número inicial. Por ejemplo, 297, cuyo cuadrado es 88209, si ahora dividimos en dos partes y las sumamos obtenemos que: 88+209=297. Si a esos números los llamamos números de Kaprekar, este ejercicio consiste en hacer un procedimiento que calcule y muestre todos los números de Kaprekar menores a N, donde N es un valor natural que se introduce como único argumento de ese procedimiento.

15. El matemático y geómetra indú **Bhaskhara**, el Sabio, vivió en el siglo XII y en su famosa obra *Lilavati*, plantea el problema siguiente:

> Un barco que volvía de Serendib con una carga de especias sufrió un fuerte temporal y, gracias a la pericia de 3 valerosos marineros lograron salvar la vida, el barco y su cargamento. El capitán del navío, agradecido, les dijo que al llegar a puerto repartiría un cofre de monedas de oro entre los 3 valerosos marineros.
>
> Por la noche, uno de los valerosos marineros, desconfiado, abrió el cofre y repartió las monedas en 3 montones y viendo que sobraba una, pensó que eso traería problemas y la tiró al mar. Hecho esto, se guardó su parte en un bolsillo, dejando el resto en el cofre. Más tarde, un segundo marinero, ignorando lo que hizo el primero, procedió del mismo modo: Dividió las monedas que había en el cofre en 3 montones y viendo que sobraba una, pensó que traería problemas y la tiró al mar. Hecho esto, se guardó su parte en un bolsillo, dejando el resto en el cofre.
>
> La casualidad quiso que el tercer marinero procediera de la misma forma: Dividió las monedas del cofre en 3 montones y viendo que sobraba una, pensó que traería problemas, la tiró al mar y se guardó su parte en un bolsillo, dejando el resto en el cofre.
>
> A la mañana siguiente, el capitán se extrañó de que faltaran monedas, pues él sabía que inicialmente había entre 200 y 300 monedas, pero para no crear problemas, cumplió su palabra: Dividió las monedas en 3 montones y, viendo que sobraba una, se la quedó para él, y entregó un montón a cada uno de los 3 avariciosos marineros.

La pregunta es: **¿Cuántas monedas había inicialmente en el cofre?**

Programar una función sin argumentos de nombre `Marineros3` que calcule y devuelva el número de monedas que había inicialmente en el cofre. El programa principal debe

mostrar ese número y, usándolo debe calcular y mostrar las cantidades que se llevaron cada uno de los 3 marineros.

Nota: La solución es: 241 monedas, repartidas de la siguiente forma 103, 76, 58, 1 y 3 monedas, respectivamente para los 3 marineros, el capitán y el mar.

16. En el año del señor de 1582 el Papa Gregorio XIII implantó el llamado CALENDARIO GREGORIANO, una modificación del anterior calendario, el Juliano, que se adapta mejor a la duración real de un año (una vuelta al sol). Este calendario es el que usamos en la actualidad y, según él, un año es BISIESTO:

    1. Si es divisible por 4 y no por 100
    2. Caso de ser divisible por 100 que lo sea también por 400.

    Ejemplos: El 1900 no fue bisiesto y el 2000 sí.

    Codifique lo siguiente:

    (a) Función `Bisiesto`: Devuelve un valor lógico verdad si el año que toma como único argumento es bisiesto y falso en caso contrario.

    (b) Función `Dia1Enero`: Acepta un año como único argumento y calcula, usando la función anterior, el día de la semana del 1 de Enero de ese año. Para ello, tendremos en cuenta que el 1 de Enero de 1996 fue Lunes. El día de la semana resultante será devuelto por la función usando un número (1: Lunes, ..., 7: Domingo).

    (c) Programa Principal: Lee un año y, usando las funciones anteriores, indicará si dicho año es o no bisiesto, así como el día de la semana del 1 de Enero de ese año. El día de la semana se escribirá por pantalla con el nombre (Lunes, Martes, ...). Se debe controlar que el año sea superior a 1582.

Nivel 3 : Dificultad Avanzada

17. Dos números a y b se dice que son amigos si la suma de los divisores de a (salvo él mismo) coincide con b y viceversa. Diseña un programa que tenga como entrada dos números naturales n y m y que muestre en la pantalla todas las parejas de números amigos que existan en el intervalo determinado por n y m. Hágalo programando distintas funciones.

18. Implementar una función, `Digit(N,num)` que devuelva el dígito N-ésimo de un número `num` de tipo `long unsigned int`, teniendo en cuenta que el dígito 0 es el dígito más a la derecha (el menos significativo). La función devolverá -1 si el número no tiene suficientes dígitos. Ejemplos:

`Digit (0,3456)`	Devuelve 6
`Digit (1,3456)`	Devuelve 5
`Digit (4,3456)`	Devuelve -1

    Escriba un programa que tome como entrada desde teclado el número y la posición y escriba el dígito resultante. Si la posición es mayor que el número de dígitos se escribirá en pantalla el mensaje "`No existe ningún dígito en esa posición`".

    NOTA: Considere la posibilidad de que el número `N` sea negativo.

19. Escribe un programa que acepte como entrada desde teclado un número entero positivo y dé como salida el resultado de sumar dos a dos los dígitos que aparecen en posiciones simétricas respecto al dígito central dentro del número dado como entrada. Utilice para ello la función realizada en el ejercicio anterior. Por ejemplo :

    Para el número : 2354869
    la salida es: `2+9 = 11, 3 + 6 = 9, 5 + 8 = 13, 4`

    Para el número : 6582
    la salida es : `6 + 2 = 8, 5 + 8 = 13`

20. Las resistencias electrónicas suelen ir identificadas por un código de colores que permite marcar cada resistencia con su valor (en Ohmios, Ω) y su Tolerancia (en %). Este código de colores viene representado en la siguiente tabla:

Dígito	*Color*	*Multiplicador*	*Tolerancia*
	Ninguno		20%
	Plata	0.01	10%
	Oro	0.1	5%
0	Negro	1	
1	Marrón	10	
2	Rojo	10^2	2%
3	Naranja	10^3	
4	Amarillo	10^4	
5	Verde	10^5	
6	Azul	10^6	
7	Violeta	10^7	
8	Gris		
9	Blanco		

    El código que suele emplearse en las resistencias es un código de 4 colores, es decir, cada resistencia está marcada con 4 bandas y cada una de ellas puede ser de diferente color. Cada banda tiene un significado, que depende de cada color:

    - Las primeras 2 bandas indican un número de 2 dígitos: Esos dos dígitos vienen dados por el color de esas bandas, según la columna "Dígito" de la tabla.
    - La tercera banda es un valor por el que se multiplicará el número obtenido por las bandas anteriores. Una vez multiplicados ambos valores, obtenemos el valor de la resistencia en Ohmios (Ω).
    - La cuarta banda indica la tolerancia de la resistencia y, como puede verse en la tabla, no puede ser de cualquier color.

    Ejemplo: Unas resistencias con los siguientes colores, tienen los siguientes valores de resistencia y tolerancia:

Verde-Azul-Amarillo-Oro	560kΩ, 5%
Rojo-Negro-Rojo-Rojo	2kΩ, 2%
Rojo-Rojo-Marrón-Plata	220Ω, 10%

Según todo lo anterior, implemente un subprograma que permita calcular la resistencia y la tolerancia de una resistencia, sabiendo los códigos de colores. El subprograma tendrá, como mínimo, 4 argumentos, que serán números naturales, y que indicarán el color de las bandas según la columna "Dígito". Los colores Oro, Plata y Ninguno tomarán los valores 10, 11 y 12 respectivamente.

Implementar otro subprograma que muestre por pantalla el dígito que le corresponde a cada color (incluyendo los dígitos 10, 11 y 12). Codificar también un programa principal que pida los colores de las 4 bandas y muestre los valores devueltos por el anterior subprograma. El programa mostrará el dígito que le corresponde a cada color usando el procedimiento ya creado y leerá de teclado 4 números que corresponderán a los colores de las 4 bandas. Tras esta lectura mostrará los datos de la resistencia con esos colores en las bandas. El programa se repetirá indefinidamente hasta que lea un valor negativo como color de una banda.

21. Implementar una función que devuelva un dato de tipo `float`, que coincida con la conversión a EUROS de la cantidad dada como primer argumento, también de tipo `float`. El segundo y último argumento de esta función, será un dígito (`unsigned int`) que codifique el país de origen de la moneda que se está empleando en la cantidad indicada en el primer argumento. Para ello, se tendrán en cuenta las cantidades indicadas en la siguiente tabla:

Dígito	*País*	*1 EURO*	*Moneda*
1	España	166.386	Peseta
2	Italia	1936.27	Lira
3	Alemania	1.95583	Marco
4	Francia	6.55957	Franco
5	Portugal	200.482	Escudo
6	Austria	13.7603	Chelín
7	Bélgica	40.3399	Franco
8	Finlandia	5.94573	Marco
9	Irlanda	0.787564	Libra
10	Luxemburgo	40.3399	Franco
11	Holanda	2.20371	Florín

Codifique también un programa principal que, haciendo uso de la función programada, nos permita visualizar esos datos. Observe que la función no debe ni leer, ni escribir nada, simplemente se limitará a calcular el valor y devolverlo (con `return`). Será el programa principal el que se encargará de leer y escribir los datos pertinentes.

22. Implemente ahora otra función para que nos permita efectuar conversiones entre todas las monedas involucradas, incluidos los EUROS. Es decir, el ejercicio también debe permitir efectuar la conversión entre pesetas y liras, o entre pesetas y francos... El primer argumento de la función será la cantidad de la que se desea efectuar la conversión, el segundo argumento será un `unsigned int` indicando la moneda de origen esa cantidad y, por último, el tercer argumento será otro `unsigned int` que indicará la moneda destino, a la que se desea efectuar la conversión. Ejemplo:

1 Peseta (PTE) = 1/166.386 Euros (EUR) = 1936.27/166.386 Liras (LIT) = 11.637 Liras

23. El genio de las matemáticas indio Srinavasa Ramanujan (1887-1920) trabajó en Cambridge con su colega inglés Godfrey Hardy (1877-1947). Cuando Ramanujan enfermó, Hardy solía visitarlo en el hospital. Un día, Hardy comentó a su amigo: "El taxi que me ha traído tenía un número bastante soso, era el ????". Ramanujan respondió: "No Hardy, ese es un número muy interesante. Es el más pequeño de los números que se pueden expresar como la suma de 2 cubos, de dos maneras distintas". O sea, ???? = $\mathtt{a}^3 + \mathtt{b}^3 = \mathtt{c}^3 + \mathtt{d}^3$.

    Escriba una función que calcule el número del taxi de Hardy, sabiendo que el número no tenía más de 4 cifras.

    NOTA: Una forma de resolverlo es escribiendo 4 bucles anidados para mover las variables `a`, `b`, `c` y `d`. La solución es: 1729.

24. Programe una función con dos argumentos. El primero será un carácter y el segundo un puntero a carácter (un paso de arg. por referencia). La función servirá para descubrir el tipo de carácter que es el primer argumento y, según eso devolverá (con `return`) los siguientes valores:

    - **2**: El carácter es una letra del idioma español en mayúculas (se incluyen las vocales acentuadas, la U con diéresis y la Ñ, por supuesto).
    - **1**: El carácter es una letra del idioma español en minúsculas.
    - **0**: El carácter es un dígito numérico (de '0' a '9').
    - **−1**: El carácter es de cualquier otro tipo.

    Además, si el carácter es una letra se devolverá en el segundo argumento la misma letra cambiando su estado (de mayúsculas a minúsculas y viceversa). En la función no debe usarse ningún número que referencie al código ASCII (en su lugar se usarán los caracteres entre comillas simples).

25. Programe una función que lea sucesivamente caracteres uno a uno hasta que lea un punto (carácter '.'). La función escribirá cada carácter leído intercambiando mayúsculas por minúsculas y viceversa. Para ello usará la función anterior. Además, la función tendrá 3 argumentos numéricos en los que devolverá cuántos caracteres hay de cada tipo (letras, números y el resto). Observe que los 3 argumentos deben pasar por referencia.