Sesión 3: Sentencias de control de flujo de ejecución (4h)

En esta sesión de laboratorio se realizarán programas sencillos que contendrán sentencias de control de flujo condicionales e iterativas y anidaciones entre ellas. Se plantearán diferentes algoritmos para el tratamiento de secuencias y se trabajarán problemas relacionados con las conversiones de tipo.

El objetivo de esta sesión es que el alumno domine el uso de la diferentes sentencias de control de flujo y sea capaz de identificar cuales de ellas son necesarias para resolver un problema determinado.

Ejercicio 1:

Determine la salida por pantalla que genera cada uno de los siguientes fragmentos de código:

(a)

```
int i, k;

k = 5;
for (i = 0; i < 3; i++)
{
      k = k + 10;
      printf("%d %d\n", i, k);
}
printf("%d %d\n", i, k);
```

(b)

```
int i, k;

i = 0;
k = 5;
while (i < 3)
{
      k = k + 10;
      printf("%d %d\n", i, k);
      i = i + 1;
}
printf("%d %d\n", i, k);
```

(c)

```
int i, k;

i = 0;
k = 5;
do {
      k = k + 10;
      printf("%d %d\n", i, k);
      i = i + 1;
} while (i < 3);
printf("%d %d\n", i, k);
```

Ahora compruebe si los resultados son correctos escribiendo un programa (fichero llamado *sesion3_ej1.c*) que contenga los fragmentos de código anteriores.

Ejercicio 2:

Escriba un programa (en el fichero *sesion3_ej2.c*) que primero pida al usuario que introduzca por teclado un número natural n (n>=0) y a continuación muestre por pantalla la secuencia de números naturales comprendidos entre 0 y n, estos incluidos.
Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca un numero natural: 8
Secuencia de numeros: 0, 1, 2, 3, 4, 5, 6, 7, 8
```

```
Introduzca un numero natural: 0
Secuencia de numeros: 0
```

```
Introduzca un numero natural: 11
Secuencia de numeros: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
```

Ejercicio 3:

Escriba un programa que primero pida al usuario que introduzca por teclado una secuencia de números naturales y a continuación muestre por pantalla la suma de todos los números de la secuencia introducida.
a) Escriba una primera versión (fichero *sesion3_ej3a.c)* asumiendo que la secuencia contiene exactamente 10 números. Use la sentencia **`for`** en la implementación.
Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca una secuencia de 10 numeros naturales: 1 3 4 2 5 6 8 7 9
10
Suma: 55
```

b) Escriba una segunda versión (fichero *sesion3_ej3b.c)* asumiendo que la secuencia de números finaliza con el valor entero -1 (este valor no se debe sumar al resultado final) y que la secuencia puede estar vacía. Utilice la sentencia **`while`** en la implementación.
Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca una secuencia de numeros naturales y finalice con -1: -1
Suma: 0
```

```
Introduzca una secuencia de numeros naturales y finalice con -1: 3 4
6 5 -1
Suma: 18
```

c) Escriba una tercera versión (fichero *sesion3_ej3c.c)* asumiendo que la secuencia de números finaliza con el valor entero -1 (este valor no se debe sumar al resultado final) y que la secuencia contiene al menos un número natural. Impleméntelo usando la sentencia **`do-while`**.
Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca una secuencia de numeros naturales y finalice con -1: 2 3
-1
Suma: 5
```

Ejercicio 4:

Escriba un programa (en el fichero *sesion3_ej4.c*) que primero pida al usuario que introduzca por teclado un número natural n (n>0) y a continuación muestre por pantalla todos sus divisores.

Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca un numero natural: 12
Los divisores de 12 son: 1, 2, 3, 4, 6, 12
```

```
Introduzca un numero natural: 19
Los divisores de 19 son: 1, 19
```

Ejercicio 5:

a) Escriba un programa (en el fichero *sesion3_ej5a.c*) que primero pida al usuario que introduzca por teclado un número natural m $(m \geq 0)$ y a continuación muestre por pantalla $m!$ (el factorial de m). **Declare m del tipo `unsigned long`**. Recuerde que:

$$m! = m \times (m-1) \times (m-2) \times \cdots \times 2 \times 1$$
$$0! = 1$$

Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca el valor de m: 4
4! = 24
```

```
Introduzca el valor de m: 14
14! = 87178291200
```

b) Escriba un programa (en el fichero *sesion3_ej5b.c*) que primero pida al usuario que introduzca por teclado un número natural n $(n >= 0)$ y a continuación muestre por pantalla el valor aproximado del número e. Recuerde que el número e se puede calcular de forma aproximada usando la siguiente serie numérica:

$$e = \sum_{i=0}^{n} \frac{1}{i!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!}$$

El valor de n indica la cantidad de términos a considerar de la serie. En la medida en que el valor de n aumente, el valor obtenido del número e será más preciso.

Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca el valor de n: 0

Numero e = 1.000000
```

```
Introduzca el valor de n: 5

Numero e = 2.716667
```

```
Introduzca el valor de n: 15

Numero_e = 2.718282
```

Ejercicio 6:

a) Escriba un programa (en el fichero *sesion3_ej6a.c*) que primero pida al usuario que introduzca por teclado el valor de un número real x y de un número natural n ($n >= 0$) y, a continuación, muestre por pantalla el valor de x^n (x elevado a n). Para realizar la implementación de este ejercicio, solamente se pueden utilizar las operaciones aritméticas elementales (+, -, *, /, %). **No se pueden utilizar las funciones de la librería matemática `math.h`**. Declare las variables reales de tipo `double` y las naturales de tipo `unsigned int`.

Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca el valor de x (real): 3.1
Introduzca el valor de n (natural): 4
El resultado de 3.10 elevado a 4 es: 92.35
```

b) Escriba un programa (en el fichero *sesion3_ej6b.c*) que primero pida al usuario que introduzca por teclado un número real x y un número natural n ($n > 0$) y a continuación muestre por pantalla el valor numérico de $cos(x)$, utilizando la siguiente aproximación por serie de Taylor:

$$cos\,(x) = \sum_{i=0}^{n} \left(\frac{x^{2i}}{(2i)!}\,(-1)^i \right)$$

Para obtener un resultado más preciso, declare las variables reales de tipo `double`.

Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca un valor real (x): 0
Introduzca un valor natural (n): 5
cos(0.00) = 1.00
```

```
Introduzca un valor real (x): 3.14
Introduzca un valor natural (n): 6
cos(3.14) = -1.00
```

```
Introduzca un valor real (x): 1.57
Introduzca un valor natural (n): 4
cos(1.57) = 0.00
```

```
Introduzca un valor real (x): 0.79
Introduzca un valor natural (n): 9
cos(0.79) = 0.70
```

Ejercicio 7:

Escriba un programa (en el fichero *sesion3_ej7.c*) que muestre continuamente por pantalla el siguiente menú de opciones (en verde) hasta que el usuario seleccione la opción 4 para salir:

```
1. Mostrar divisores (ejercicio 4)
2. Calcular numero e (ejercicio 5)
3. Calcular coseno (ejercicio 6)
4. Salir

Escoja una opción:
```

Cuando el usuario elije una opción entre 1 y 3, se debe ejecutar el código correspondiente para realizar la acción indicada. Tenga en cuenta que las acciones correspondientes a las opciones 1, 2, y 3 del menú coinciden con los ejercicios 4, 5 y 6 anteriores. Una vez finalizada la acción, se debe mostrar nuevamente el menú de opciones por pantalla. Cuando el usuario elije la opción 4, el programa finaliza. Si el usuario introduce cualquier otro valor distinto de 1, 2, 3 ó 4, por pantalla saldrá el mensaje "**Opción incorrecta**" (en rojo) y se mostrará nuevamente el menú de opciones.
Recuerde incluir la librería `colores.h` en el fichero de código para poder mostrar un mensaje en color. Para compilar deberá ejecutar el siguiente comando:

gcc –o sesion3_ej8 sesion3_ej8.c colores.c

Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
1. Mostrar divisores (ejercicio 4)
2. Calcular numero e (ejercicio 5)
3. Calcular coseno (ejercicio 6)
4. Salir

Escoja una opcion: 1
Introduzca un numero natural: 23

Los divisores de 23 son: 1, 23

1. Mostrar divisores (ejercicio 4)
2. Calcular numero e (ejercicio 5)
3. Calcular coseno (ejercicio 6)
4. Salir

Escoja una opcion: 3
Introduzca un valor real (x): 2.09
Introduzca un valor natural (n): 7

cos(2.09) = -0.50

1. Mostrar divisores (ejercicio 4)
2. Calcular numero e (ejercicio 5)
3. Calcular coseno (ejercicio 6)
4. Salir

Escoja una opcion: 2
Introduzca el valor de n: 3

Numero_e = 2.666667

1. Mostrar divisores (ejercicio 4)
2. Calcular numero e (ejercicio 5)
3. Calcular coseno (ejercicio 6)
4. Salir

Escoja una opcion: 7
Opcion Incorrecta
```

```
1. Mostrar divisores (ejercicio 4)
2. Calcular numero e (ejercicio 5)
3. Calcular coseno (ejercicio 7)
4. Salir

Escoja una opcion: 4
```

Ejercicio 8:

Escriba un programa (fichero *sesion3_ej8.c*) que primero pida al usuario que introduzca por teclado una frase acabada en '.' y a continuación muestre por pantalla el número de palabras que contiene la frase introducida. Tenga en cuenta las siguientes consideraciones:

1. Las palabras de la frase están separadas por uno o más espacios en blanco.
2. No hay espacios en blanco al inicio y al final de la frase (entre la última palabra y el punto).
3. La frase contiene al menos una palabra.

Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca una frase acabada en punto: Estoy.
Numero de palabras: 1
```

```
Introduzca una frase acabada en punto: Estoy   haciendo la practica.
Numero de palabras: 4
```

Ejercicio 9:

a) Escriba un programa en C (en el fichero *sesion3_ej9a.c*) que primero pida al usuario un número natural n $(n > 0)$ y a continuación determine si es un número primo. En el caso de que el número sea primo, el programa mostrará por pantalla un mensaje en color verde. En caso contrario el mensaje será en color rojo e indicará que no es un número primo.

Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca un numero natural: 1

El numero 1 es primo
```

```
Introduzca un numero natural: 4

El numero 4 no es primo
```

```
Introduzca un numero natural: 17

El numero 17 es primo
```

b) Escriba un programa en C (en el fichero *sesion3_ej9b.c*) que primero pida al usuario que introduzca una secuencia de números naturales y, a continuación, muestre por

pantalla todos los números de la secuencia que son primos. La secuencia de números finaliza con el valor entero -1 y la secuencia puede estar vacía.
Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca una secuencia de numeros naturales y finalice con -1:
2 3 17 6  -1
El numero 2 es primo
El numero 3 es primo
El numero 17 es primo
```

```
Introduzca una secuencia de numeros naturales y finalice con -1:
4 6 -1

No hay numeros primos en la secuencia
```

Ejercicio 10:

a) Escriba un programa en C (en el fichero *sesion3_ej10a.c*) que pida al usuario un número natural n ($n > 0$) y dibuje por pantalla un triángulo rectángulo formado por n líneas de '*' (asteriscos).
Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca un numero natural: 6

*
**
***
****
*****
******
```

```
Introduzca un numero natural: 4

*
**
***
****
```

b) Modifique el programa anterior (y guárdelo en el fichero *sesion3_ej10b.c*) para que muestre las filas de asteriscos en ROJO y AZUL de forma alternada, comenzando siempre por el color ROJO.
Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca un numero natural: 4

*
**
***
****
```

c) Modifique el programa del apartado (a) (y guárdelo en el fichero *sesion3_ej10c.c*) para que muestre las columnas de asteriscos en VERDE y AMARILLO de forma alternada, siendo la primera columna siempre de color VERDE.
Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca un numero natural: 4

*
**
***
****
```

d) Escriba una última versión del programa (y guárdelo en el fichero *sesion3_ej10d.c*) para que muestre las columnas de asteriscos en VERDE y AMARILLO de forma alternada, siendo la primera columna siempre de color VERDE, y el triángulo salga invertido.
Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca un numero natural: 4

****
***
**
*
```

Ejercicio 11:

Escriba un programa en C (en el fichero *sesion3_ej11.c*) que pida al usuario que introduzca por teclado una secuencia de números reales que corresponden con las notas finales de una clase. La secuencia finaliza con un valor negativo cualquiera. El programa debe mostrar por pantalla la nota media de la clase, la nota más baja, la nota más alta, el número de alumnos aprobados y el número de alumnos suspendidos.
Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca una secuencia de notas finalizada con un valor
negativo: 3.2, 5.8, 10, 9.8, 7.2, 4.5, 6.7, 1.2, 0.5, -3

Nota media: 5.43
Nota mas alta: 10.00
Nota mas baja: 0.50
Numero de aprobados: 5
Numero de suspendidos: 4
```

Ejercicio 12:

Finalizar el programa del fichero *sesion3_ej12.c* para que muestre por pantalla la evolución de un juego llamado "**Ratón que te pilla el gato**". Este juego dispone de una pista circular con 60 casillas, enumeradas de la 0 a la 59. El ratón comienza en la casilla 29 y el gato comienza en la casilla 0. Ratón y gato tiran alternativamente un dado. Empieza tirando el ratón. El ratón avanza tantas casillas como indique el dado. El gato avanza el número de casillas que indique el dado multiplicado por 2. El juego se acaba cuando el gato pilla al ratón (en este caso gana el gato) o cuando el ratón pilla al gato (en este caso gana el ratón). *Pillar* significa que ambos están en la misma casilla. El programa debe mostrar por pantalla la evolución del juego tal como se muestra en el ejemplo. Al final, el programa indica quién es el ganador.
Ejemplo de ejecución:

```
Raton: He sacado un 6. Voy a la casilla 35
Gato: He sacado un 6. Voy a la casilla 12
Raton: He sacado un 6. Voy a la casilla 41
Gato: He sacado un 6. Voy a la casilla 24
Raton: He sacado un 1. Voy a la casilla 42
Gato: He sacado un 6. Voy a la casilla 36
Raton: He sacado un 5. Voy a la casilla 47
Gato: He sacado un 2. Voy a la casilla 40
Raton: He sacado un 1. Voy a la casilla 48
Gato: He sacado un 2. Voy a la casilla 44
Raton: He sacado un 4. Voy a la casilla 52
Gato: He sacado un 4. Voy a la casilla 52
Gato: He ganado!!!!
```

Ayuda: Para la implementación se debe usar las funciones `inicializar_azar` y `numero_al_azar` de la librería `azar.h` que os proporcionamos. Con estas funciones podemos generar números aleatorios diferentes en cada ejecución.
Para ello, al inicio del programa se debe llamar a la función `inicializar_azar()` que inicializa la semilla del generador de números aleatorios.
A partir de aquí, cada vez que se quiera generar un número aleatorio para simular un lanzamiento del dado, se deberá llamar a la función `numero_al_azar(MAX)` que devolverá un número entero aleatorio entre `0` y `MAX-1`. Por ejemplo, dada una variable `num` de tipo int, después de ejecutar:

```
num = numero_al_azar(100);
```

la variable `num` almacenará un valor entero entre 0 y 99.