Práctica 3: Estructuras de control (3 sesiones)

Semanas del 23 de septiembre al 7 de octubre de 2024

Parte 1: Estructuras condicionales

Ejercicio 1.1

Escribir un programa que pida dos números al usuario y muestre el mayor de ellos o, si son iguales, un mensaje indicándolo.

Ejemplos de funcionamiento

```
$ ./ejercicio1.1

Escribe un número entero: 8
Escribe otro número entero: 3
El número 8 es mayor que el número 3

$ ./ejercicio1.1

Escribe un número entero: 32
Escribe otro número entero: 32
Los números 32 y 32 son iguales
```

Ejercicio 1.2

Escribir un programa en C que pregunte al usuario una hora, sólo la hora entre 0 y 23. El programa mostrará un saludo en función de la hora: “Buenos días”, para una hora entre las 7h y las 14h; “Buenas tardes”, para una hora entre las 15h y las 20h; y “Buenas noches” para una hora entre las 21h y las 6h.

Ejemplos de funcionamiento

```
$ ./ejercicio1.2

Dime la hora del día (0-23): 5
Buenas noches

$ ./ejercicio1.2

Dime la hora del día (0-23): 13
Buenos días
```

Ejercicio 1.3

Escribir un programa en C que haga la función de calculadora para las operaciones suma (+), resta (-), multiplicación (*) y división (/). El programa preguntará al usuario el operando 1, el operando 2 y la operación, y devolverá el resultado de op1 op op2. Nota: Los operandos pueden tener decimales.

Ejemplos de funcionamiento

```
$ ./ejercicio1.3

Introduce el operando 1: 5
Introduce el operando 2: 7
Introduce la operación a realizar (+, -, *, /): +
El resultado de la operación 5.00 + 7.00 es 12.00

$ ./ejercicio1.3

Introduce el operando 1: 234.5
Introduce el operando 2: 6
Introduce la operación a realizar (+, -, *, /): /
El resultado de la operación 234.50 / 6.00 es 39.08
```

Ejercicio 1.4

Escribir un programa en C que pida al usuario tres números enteros y el orden en que mostrarlos, ascendente o descendente. Finalmente, el programa imprimirá en pantalla los números en el orden indicado por el usuario.

Ejemplos de funcionamiento

```
$ ./ejercicio1.4

Introduce tres números enteros
* Número 1: 5
* Número 2: 2
* Número 3: 8
¿En qué orden quieres que los muestre, ascendente (A) o descendente (D)? A
Números en orden ascendente: 2 5 8

$ ./ejercicio1.4

Introduce tres números enteros
* Número 1: 5
* Número 2: 2
* Número 3: 8
¿En qué orden quieres que los muestre, ascendente (A) o descendente (D)? D
Números en orden descendente: 8 5 2
```

Ejercicio 1.5

Escribir un programa en C para calcular el índice de masa corporal (IMC), que se calcula como `IMC=peso/(altura*altura)`. El programa mostrará un mensaje con el valor del IMC y el grupo en el que se encuentra según la siguiente clasificación:

- Si IMC< 18.0, Inferior al normal
- Si 18.1<=IMC<=24.9, Normal
- Si 25.0<=IMC<=29.9, Sobrepeso
- Si IMC>=30, Obesidad

Nota: El peso se expresa en kg y la altura en metros.

Ejemplos de funcionamiento

```
$ ./ejercicio1.5

Introduce el peso en kg: 74.5
Introduce la altura en metros: 1.8
Valor de IMC: 22.99 - Normal

$ ./ejercicio1.5

Introduce el peso en kg: 91.8
Introduce la altura en metros: 1.74
Valor de IMC: 30.32 - Obesidad
```

Parte 2.1: Estructuras de repetición o iterativas: bucles determinados

Ejercicio 2.1.1

Escribir un programa en C que pida al usuario un número entero e imprima por pantalla su tabla de multiplicar.

Ejemplo de funcionamiento

```
$ ./ejercicio2.1.1

Introduce un número entero: 4
4 x 0 = 0
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

Ejercicio 2.1.2

Escribir un programa en C que pida al usuario un número entero positivo e imprima los divisores que tiene. Además, también imprimirá un mensaje diciendo si el número es primo o no. Nota: Si el número introducido no es correcto se mostrará un mensaje de error al usuario y no se hará nada.

Ejemplos de funcionamiento

```
$ ./ejercicio2.1.2

Introduce un número entero: 12
Divisores del número 12: 1, 2, 3, 4, 6, 12 => NO ES PRIMO


$ ./ejercicio2.1.2

Introduce un número entero: 7
Divisores del número 7: 1, 7 => ES PRIMO

$ ./ejercicio2.1.2

Introduce un número entero: 156
Divisores del número 156: 1, 2, 3, 4, 6, 12, 13, 26, 39, 52, 78, 156 => NO ES
PRIMO
```

Ejercicio 2.1.3

Escribir un programa en C que pida al usuario un número entero positivo y muestre por pantalla la cuenta atrás desde ese número hasta cero separados por comas.

Ejemplo de funcionamiento

```
$ ./ejercicio2.1.3

Introduce un número entero positivo: 7
Cuenta atrás desde 7 hasta 0: 7, 6, 5, 4, 3, 2, 1, 0
```

Ejercicio 2.1.4

Escribir un programa que visualice en pantalla un menú que permita elegir entre tres figuras para dibujar, preguntando al usuario el tamaño, como mínimo 4 y como máximo 15, que debe tener la figura (se debe comprobar). El menú tendrá una opción adicional que permitirá salir del programa. Mientras la opción elegida no sea correcta se volverá a mostrar el menú.

Ejemplo de funcionamiento

```
$ ./ejercicio2.1.4

MENÚ
------------
1 - Figura 1
2 - Figura 2
3 - Figura 3
4 - Figura 4
```

```
5 - Figura 5
0 - Salir

Elige opción: 3
Introduce el tamaño de la figura (4 a 15): 5
```

Figura 1	Figura 2	Figura 3	Figura 4	Figura 5
*	*********	* *	*****	*
**	*******	** **	*****	***
***	*****	*** ***	*****	*****
****	***	**** ****	*****	*******
*****	*	*********	*********	*********
			*****	*******
			*****	*****
			*****	***
			*****	*

Ejercicio 2.1.5

Escribir un programa en C que pida al usuario un número par. Si el número introducido no es correcto se le debe indicar al usuario y no hacer nada. Si el número es correcto, el programa deberá mostrar en pantalla el siguiente dibujo:

```
1
3 1
5 3 1
7 5 3 1
9 7 5 3 1
```

Dibujo obtenido con un valor de 10 introducido por el usuario

Parte 2.2: Estructuras de repetición o iterativas: bucles indeterminados

Ejercicio 2.2.1

Escribir un programa en C que pida al usuario un número entero entre 50 y 100. Si el número no es correcto, se imprimirá un mensaje por pantalla indicándolo y no se hará nada. Si el número es correcto, se imprimirán en pantalla todos los números enteros que son cuadrados perfectos menores o iguales a él.

Ejemplos de funcionamiento

```
$ ./ejercicio2.2.1

Introduce un número entero entre 50 y 100: 78
Cuadrados perfectos menores que 78: 1 (1 * 1), 4 (2 x 2), 9 (3 x 3), 16 (4 x 4), 25 (5 x 5), 36 (6 x 6), 49 (7 x 7), 64 (8 x 8)
```

```
$ ./ejercicio2.2.1

Introduce un número entero entre 50 y 100: 34
ERROR: El número introducido (34) no está entre 50 y 100
```

Ejercicio 2.2.2

Escribir un programa en C que genere un número entero aleatorio entre 1 y 100. A continuación irá pidiendo números al usuario hasta que éste lo acierte. En cada intento que el usuario no acierta el programa le mostrará un mensaje diciendo si el número a adivinar es mayor o menor que el introducido. Al finalizar el programa se imprimirá el número de intentos realizados por el usuario.

Nota: Para obtener números aleatorios en C hay que incluir la librería `<stdlib.h>` y la librería `<time.h>` para inicializar la secuencia de números aleatorios cada vez que se ejecute el programa:

```c
#include<stdio.h>
#include<stdlib.h> // para generar el número aleatorio
#include<time.h> // para inicializar la semilla de números aleatorios

int main(){
  srand(time(NULL)); // inicialización de la semilla aleatoria
  int numero = 20 + rand() % 11; // número aleatorio entre 20 y 30
  // rand() % n  => genera un número aleatorio en el intervalo [0, n)
  ...
  return 0;
}
```

Ejemplo de funcionamiento

```
$ ./ejercicio2.2.2

Adivina el número oculto en el menor número de intentos posible
Introduce un número: 54
El número oculto es menor
Introduce un número: 45
El número oculto es menor
Introduce un número: 32
El número oculto es mayor
Introduce un número: 37
El número oculto es menor
Introduce un número: 35
ENHORABUENA!!! Has acertado el número oculto en 6 intentos
```

Ejercicio 2.2.3

Escribir un programa en C que genere al azar tres números sin repetir entre el 1 y el 5, y los muestre por pantalla. Al finalizar debe imprimir la cantidad de números generados en total hasta conseguir los tres distintos.

Ejemplos de funcionamiento

```
$ ./ejercicio2.2.3

Los números distintos generados son: 5, 3, 4
Números totales generados: 3
$ ./ejercicio2.2.3

Los números distintos generados son: 5, 2, 4
Números totales generados: 6
```

Ejercicio 2.2.4

Escribir un programa en C que vaya pidiendo al usuario un número real positivo hasta que se introduzca el valor 0. A continuación, el programa mostrará por pantalla el total de números introducidos y la media aritmética de todos ellos.

Ejemplo de funcionamiento

```
$ ./ejercicio2.2.4

Introduce un número positivo (0 - para terminar): 4.5
Introduce un número positivo (0 - para terminar): 6
Introduce un número positivo (0 - para terminar): 3.75
Introduce un número positivo (0 - para terminar): 0
Total de números: 3
Media aritmética calculada: 4.75
```

Ejercicio 2.2.5

Escribir un programa en C que pida al usuario un número entero y, a continuación, muestre por pantalla la siguiente información sobre el número: número total de cifras, número total de cifras pares y número total de cifras impares.

Ejemplo de funcionamiento

```
$ ./ejercicio2.2.5

Introduce un número entero: 42765
Total de cifras: 5
Total de cifras pares: 3
Total de cifras impares: 2
```