Actividad resuelta 4.1

Diseñar la función `eco()` a la que se le pasa como parámetro un número *n*, y muestra por pantalla *n* veces el mensaje «Eco...».

Actividad resuelta 4.2

Escribir una función a la que se le pasen dos enteros y muestre todos los números comprendidos entre ellos.

Actividad resuelta 4.3

Realizar una función que calcule y muestre el área o el volumen de un cilindro, según se especifique. Para distinguir un caso de otro se le pasará como opción un número: 1 (para el área) o 2 (para el volumen). Además, hay que pasarle a la función el radio de la base y la altura.

$$\text{área} = 2\pi \cdot \text{radio} \cdot (\text{altura} + \text{radio})$$

$$\text{volumen} = \pi \cdot \text{radio}^2 \cdot \text{altura}$$

Actividad resuelta 4.4

Diseñar una función que recibe como parámetros dos números enteros y devuelve el máximo de ambos.

Actividad resuelta 4.5

Crear una función que, mediante un booleano, indique si el carácter que se pasa como parámetro de entrada corresponde con una vocal.

Actividad resuelta 4.6

Diseñar una función con el siguiente prototipo:

```
boolean esPrimo(int n)
```

que devolverá `true` si `n` es primo y `false` en caso contrario.

Actividad resuelta 4.7

Escribir una función a la que se le pase un número entero y devuelva el número de divisores primos que tiene.

Solución

Actividad resuelta 4.8

Diseñar la función `calculadora()`, a la que se le pasan dos números reales (operandos) y qué operación se desea realizar con ellos. Las operaciones disponibles son: sumar, restar, multiplicar o dividir. Estas se especifican mediante un número: 1 para la suma, 2 para la resta, 3 para la multiplicación y 4 para la división. La función devolverá el resultado de la operación mediante un número real.

Actividad resuelta 4.9

Repetir la Actividad resuelta 4.4 con una versión que calcule el máximo de tres números.

Actividad resuelta 4.10

Diseñar una función que calcule a^n, donde a es real y n es entero no negativo. Realizar una versión iterativa y otra recursiva.

Actividad resuelta 4.11

Escribir una función que calcule de forma recursiva el máximo común divisor de dos números. Para ello sabemos:

$$mcd(a, b) = \begin{cases} mcd(a - b, b) & \text{si } a \geq b \\ mcd(a, b - a) & \text{si } b > a \\ a & \text{si } b = 0 \\ b & \text{si } a = 0 \end{cases}$$

4.1. Los parámetros en la llamada a una función en Java pueden ser opcionales si:

a) Todos los parámetros son del mismo tipo.
b) Todos los parámetros son de distinto tipo.
c) Nunca pueden ser opcionales.
d) Siempre que el tipo devuelto no sea `void`.

4.2. Una variable local (declarada dentro de una función) puede usarse:

a) En cualquier lugar del código.
b) Solo dentro de `main()`.
c) Solo en la función donde se ha declarado.
d) Ninguna de las opciones anteriores es correcta.

4.3. El tipo devuelto de todas las funciones definidas en nuestro programa tiene que ser siempre:

a) `int`.
b) `double`.
c) `void`.
d) Ninguna de las opciones anteriores es correcta.

4.4. ¿Qué instrucción permite a una función devolver un valor?

a) `value`.
b) `return`.
c) `static`.
d) `function`.

4.5. La forma de distinguir entre dos o más funciones sobrecargadas es:

a) Mediante su nombre.
b) Mediante el tipo devuelto.
c) Mediante el nombre de sus parámetros.
d) Mediante su lista de parámetros: número o tipos.

4.6. ¿Cuál es la definición de una función recursiva?

a) Es aquella que se invoca desde dentro de su propio bloque de instrucciones.

b) Es aquella cuyo nombre permite la sobrecarga y además realiza alguna comprobación mediante `if`.

c) Es aquella cuyo bloque de instrucciones utiliza alguna sentencia `if` (lo que llamamos caso base).

d) Es aquella que genera un bucle infinito.

4.7. El paso de parámetros a una función en Java es siempre:

a) Un paso de parámetros por copia.

b) Un paso de parámetros por desplazamiento.

c) Un paso de parámetros recursivo.

d) Un paso de parámetros funcional.

4.8. En el caso de que una función devuelva un valor, ¿cuál es la recomendación con respecto a la instrucción `return`?

a) Utilizar tantos como hagan falta.

b) Emplear tantos como hagan falta, pero siempre que se encuentren en bloques de instrucciones distintas.

c) Usar solo uno.

d) Utilizar solo uno, que será siempre la primera instrucción de la función.

4.9. ¿Cuáles de las siguientes operaciones se pueden implementar fácilmente mediante funciones recursivas?

a) $a^n = a \times a^{n-1}$.

b) esPar(n) = esImpar(n – 1) y esImpar(n) = esPar(n – 1).

c) suma(a, b) = suma(a + 1, b – 1).

d) Todas las respuestas anteriores son correctas.

4.10. En los identificadores de las funciones, al igual que en los de las variables, se recomienda utilizar la siguiente nomenclatura:

a) `suma_notas_alumnos()`.

b) `sumanotasalumnos()`.

c) `SumaNotasAlumnos()`.

d) `sumaNotasAlumnos()`.

4.11. Diseña una función que calcule y muestre la superficie y el volumen de una esfera.

$$Superficie = 4\pi \cdot radio^2$$

$$Volumen = \frac{4\pi}{3} \cdot radio^3$$

4.12. Implementa la función

```
static double distancia(double x1, double y1, double x2, double y2)
```

que calcula y devuelve la distancia euclídea que separa los puntos (x1, y1) y (x2, y2). La fórmula para calcular esta distancia es:

$$distancia = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

4.13. Crea la función `muestraPares(int n)` que muestre por consola los primeros `n` números pares.

4.14. Escribe una función a la que se pase como parámetros de entrada una cantidad de días, horas y minutos. La función calculará y devolverá el número de segundos que existen en los datos de entrada.

4.15. Diseña una función a la que se le pasan las horas y minutos de dos instantes de tiempo, con el siguiente prototipo:

```
static int diferenciaMin(int hora1, int minuto1, int hora2, int minuto2)
```

La función devolverá la cantidad de minutos que existen de diferencia entre los dos instantes utilizados.

4.16. Implementa la función `divisoresPrimos()` que muestra, por consola, todos los divisores primos del número que se le pasa como parámetro.

4.17. Escribe una función que decida si dos números enteros positivos son amigos. Dos números *a* y *b* son amigos si la suma de los divisores propios (distintos de él mismo) de *a* es igual a *b*. Y viceversa.

Para probar se pueden usar los números 220 y 284, que son amigos.

4.18. Crea una función que muestre por consola una serie de números aleatorios enteros. Los parámetros de la función serán: la cantidad de números aleatorios que se mostrarán y los valores mínimos y máximos que estos pueden tomar.

4.19. Sobrecarga la función realizada en la Actividad de aplicación 4.18 para que el único parámetro sea la cantidad de números aleatorios que se muestra por consola. Los números aleatorios serán reales y estarán comprendidos entre 0 y 1.