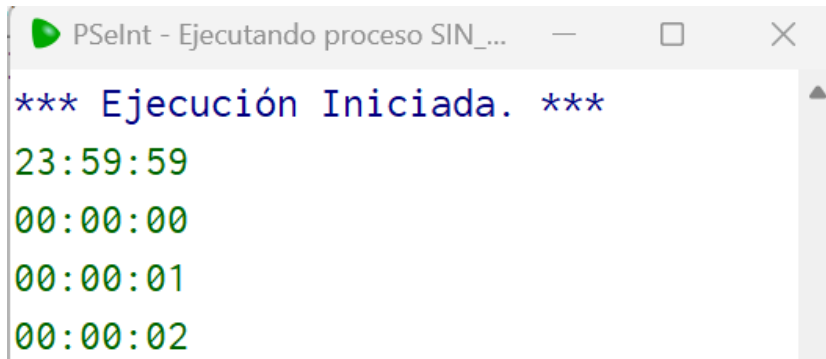


Ejercicio 1: Reloj Digital

Realiza el algoritmo para simular un reloj que nunca se pare. Debe mostrar la hora en formato 00:00:00 (hora:minuto:segundo). Por ejemplo, 05:37:09 corresponde a las 5 horas, 37 minutos y 9 segundos.

También debes hacer que espere un segundo real para darle más realismo. Para ello, usa la siguiente instrucción: *Esperar 1 segundos*

Mostrará la hora una vez por segundo, comenzando por las 23:59:59.



```
PSeInt - Ejecutando proceso SIN_...
*** Ejecución Iniciada. ***
23:59:59
00:00:00
00:00:01
00:00:02
```

Ejercicio 2: Array notas

Realiza un algoritmo que rellene un array con N notas, entre 0 y 10, generadas aleatoriamente, siendo N un número pedido al usuario (supondremos que siempre introducirá un número entero positivo).

Una vez relleno, mostrar el contenido del array y, de acuerdo a las notas contenidas, indicar cuántos estudiantes son:

Deficientes 0-3

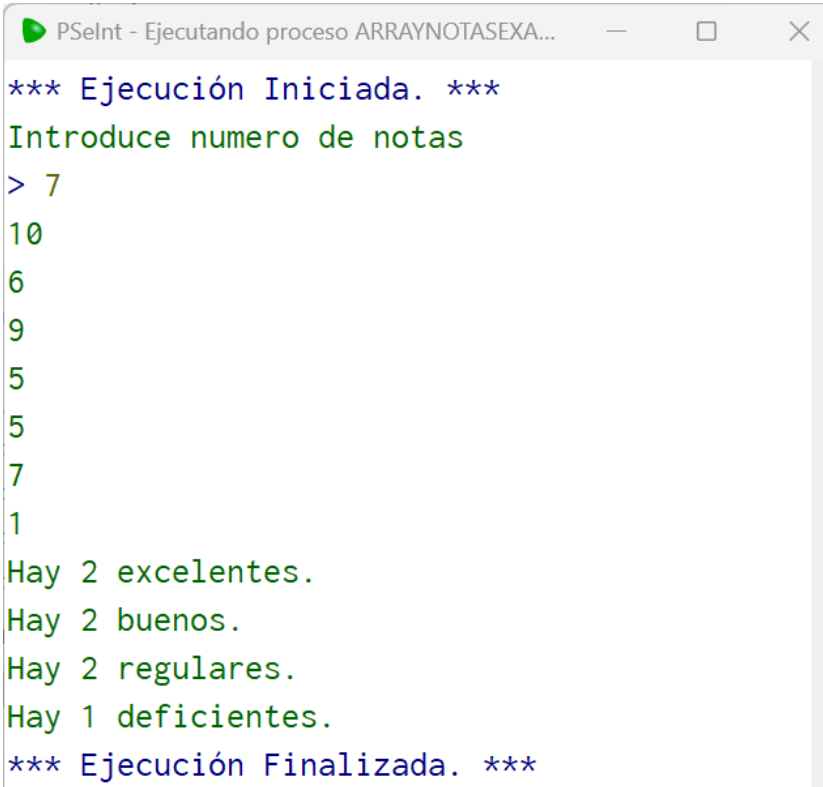
Regulares 4-5

Buenos 6-8

Excelentes 9-10

Para generar un número aleatorio entre dos números, usad la función predefinida *NumeroAleatorio<-Aleatorio(minimo,maximo)*

Ejemplo de ejecución para N=7:



```
PSeInt - Ejecutando proceso ARRAYNOTASEXA...
*** Ejecución Iniciada. ***
Introduce numero de notas
> 7
10
6
9
5
5
7
1
Hay 2 excelentes.
Hay 2 buenos.
Hay 2 regulares.
Hay 1 deficientes.
*** Ejecución Finalizada. ***
```

Ejercicio 3:

Realizar un algoritmo para simular un juego de adivinar un número, en el que el usuario introduce números y se le va indicando si es menor o mayor que el correcto hasta que acierte.

Tendremos guardado en una variable el valor que nosotros queramos como número a acertar. Este número será un entero entre 1 y 100.

Tendremos que crear y usar las siguientes funciones:

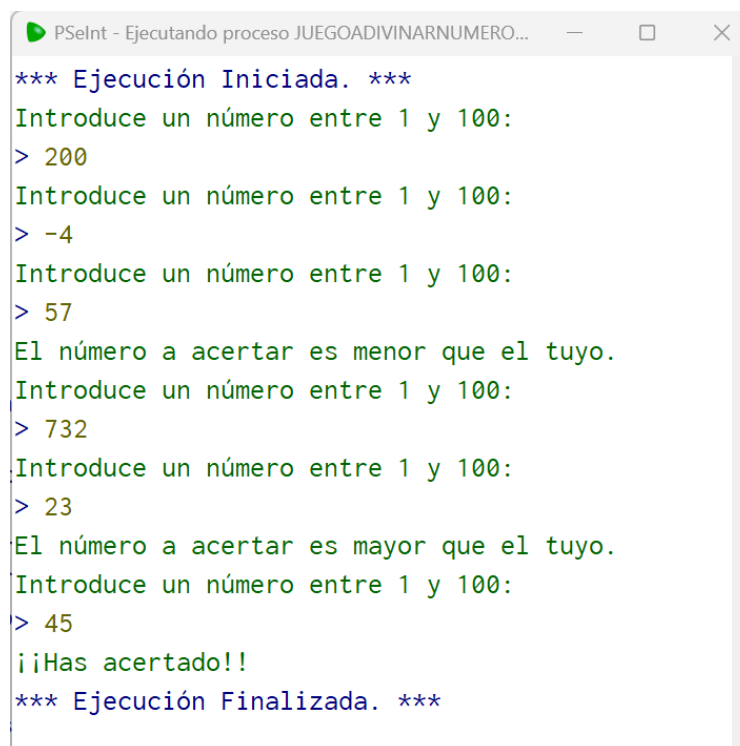
leerNumero(): Solicita un número al usuario y hasta que éste no escriba un valor entre 1 y 100, vuelve a pedir el número. La función devuelve como salida el número resultante.

comprobarValor(numeroUsuario, numeroCorrecto): comprueba si el número es el correcto y devuelve un número que puede ser: 0 si los dos números son iguales, 1 si el numeroUsuario es mayor que el numeroCorrecto, ó -1 si el numeroUsuario es menor que el numeroCorrecto

Según lo que devuelva la función comprobarValor, se mostrará un mensaje indicando si se ha acertado el número, si el número introducido es mayor que el número buscado, o si es menor.

El juego continuará hasta que el usuario acierte el número.

Ejemplo de ejecución para numeroCorrecto = 45:



```
*** Ejecución Iniciada. ***
Introduce un número entre 1 y 100:
> 200
Introduce un número entre 1 y 100:
> -4
Introduce un número entre 1 y 100:
> 57
El número a acertar es menor que el tuyo.
Introduce un número entre 1 y 100:
> 732
Introduce un número entre 1 y 100:
> 23
El número a acertar es mayor que el tuyo.
Introduce un número entre 1 y 100:
> 45
¡¡Has acertado!!
*** Ejecución Finalizada. ***
```