¿QUE SON LOS NÚMEROS PSEUDOALEATORIOS?

Son unos números generados por medio de una **función** (determinista, no aleatoria) que aparentan ser aleatorios. Estos números pseudoaleatorios se generan a partir de un valor inicial aplicando iterativamente la función. La sucesión de números pseudoaleatorios es sometida a diversos tests para medir hasta qué punto se asemeja a una sucesión aleatoria.

Un PRNG (**generador de números aleatorios**) es una estructura de datos con unos algoritmos asociados que, a partir de un valor inicial, genera secuencias de números en un orden, aparentemente aleatorio o, lo que es lo mismo, al azar y es aparentemente porque no es real: si se tuviera la tecnología y el tiempo suficiente, se podría encontrar un cierto orden lógico en las secuencias y predecirlas, pero eso es otro tema; a los efectos, serán considerados como realmente aleatorios. Otro concepto es el de **semilla**. La semilla, no es más que ese valor inicial del que se hablaba antes. Un valor, a ser posible, bastante aleatorio en sí mismo.

Por ejemplo, un método utilizado en criptografía es el generador RSA dado por

*x(i+1) = x(i)^d + mod (p*q), con:*

- p, q primos
- d un número que verifica que mcd(d, (p-1)*(q-1)) = 1.

Los dos principales inconvenientes de las sucesiones de números pseudoaleatorios son los siguientes:

- A partir de un mismo valor inicial se genera la misma sucesión.
- En general, la sucesión es periódica.

Estos inconvenientes se solucionan escogiendo generadores con periodos largos. De todas maneras, en muchas aplicaciones son útiles estos defectos, ya que nos irá bien que la sucesión parezca aleatoria, pero a la vez conozcamos que los números se repiten periódicamente y cuál es la longitud del periodo. También cuando se quiere repetir un experimento en las mismas condiciones, nos interesará partir del mismo generador y con el mismo valor inicial.

FUNCIONES PARA GENERAR NÚMEROS PSEUDO ALEATORIOS EN C

- **int rand(void).** The function computes a pseudo-random number x based on a seed value stored in an internal static-duration object, alters the stored seed value, and returns x. x is in the interval [0, RAND_MAX]. (<stdlib.h>)
- **RAND_MAX**.The macro yields the maximum value returned by rand. (<stdlib.h>)
- **void srand(unsigned int seed)**. The function stores the seed value seed in a static-duration object that rand uses to compute a pseudo-random number. From a given seed value, that function always generates the same sequence of return values. The program behaves as if the target environment calls srand(1) at program startup. (<stdlib.h>)
- **time_t time(time_t *tod)**. time gives the current time, in seconds, elapsed since 00:00:00 GMT, January 1, 1970, and stores that value in the location pointed to by timer, provided that timer is not a NULL pointer. (#include <time.h>)

EJEMPLOS DE GENERACIÓN DE NÚMEROS PSEUDO ALEATORIOS EN C

```
#include <stdio.h>
#include <stdlib.h>     // Para manejar nums. aleatorios
#include <time.h>       // Para fijar la semilla del generador de nums. aleat.

int main (void)
{  time_t t;         // tipo definido en time.h
   srand ((int) time(&t)); // Fija la semilla del generador:
                           // Inicializa el generador de numeros
                           // aleatorios con el reloj del sistema
                           // Si no se hace la llamada a srand, siempre
                           // se obtiene la misma secuencia
   int numValores = 100;
   int valorEntero, inf, sup;
   float valorFloat, min, max;
   int i;

   //GENENERA 100 VALORES ENTEROS EN EL INTERVALO 0 <= valor <= sup
   inf=0, sup=10;
   printf("\n\nGenerando %d valores enteros en el intervalo [%d, %d]: \n", numValores, inf, sup);
   for (i=0; i<numValores; i++)
   {
      if(i%10==0 && i!=0) printf("\n");
      valorEntero = rand () % (sup+1);
      printf("%d\t", valorEntero);
   }
   printf("\nPulse una tecla para continuar ...");
   getchar();


    //GENENERA 100 VALORES ENTEROS EN EL INTERVALO inf <= valor <= sup
   inf=-5, sup=10;
   printf("\n\nGenerando %d valores enteros en el intervalo [%d, %d]: \n", numValores, inf, sup);
   for (i=0; i<numValores; i++)
   {
      if(i%10==0 && i!=0) printf("\n");
      valorEntero = inf + (rand () % (sup-inf+1));
      printf("%6d\t", valorEntero);
   }
   printf("\nPulse una tecla para continuar ...");
   getchar();


   //GENENERA 100 VALORES FLOAT EN EL INTERVALO 0 <= valor <= 1
   inf=0, sup=1;
   printf("\n\nGenerando %d valores float en el intervalo [%d, %d]: \n", numValores, inf, sup);
   for (i=0; i<numValores; i++)
   {
       if(i%5==0 && i!=0) printf("\n");
       valorFloat = ((float)rand ())/ RAND_MAX;
       printf("%8.5f\t",valorFloat);
   }
   printf("\nPulse una tecla para continuar ...");
   getchar();


   //GENENERA 100 VALORES FLOAT EN EL INTERVALO min <= valor <= max
   min=-10, max=0.5;
   printf("\n\nGenerando %d valores float en el intervalo [%f, %f]: \n", numValores, min, max);
   for (i=0; i<numValores; i++)
   {
      if(i%5==0 && i!=0) printf("\n");
      valorFloat = (float) rand();
      valorFloat = min + ((valorFloat * (max - min))/RAND_MAX);
      printf("%8.5f\t", valorFloat);
   }
   return (0);
}
```