# Mémo . . .random

Vous voulez programmer des jeux de hasard, mais votre langage préféré ne vous offre pas de fonction aléatoire satisfaisante ? pour des raisons de mise au point vous voulez une suite « aléatoire reproductible » ? qu'à cela ne tienne, ce petit mémo permet de simuler des fonctions aléatoires de manière simple, et parfois plus efficace que les *random* built-in.

L'algorithme le plus efficace (simple, pas cher, et tout) pour obtenir une suite aléatoire entière comprise entre 0 et $m-1$ est le suivant :

$$x_{i+1} \ = \ (a.x_i \ + \ b) \ mod \ m$$

Evidemment, il est judicieux de choisir $a$, $b$ et $m$ de telle sorte que $x_i$ atteigne toutes les valeurs entre 0 et $m-1$ :

**Proposition :** Tous les nombres entre 0 et $m-1$ sont atteints si et seulement si : $b$ est premier avec $m$ , $(a-1)$ est multiple de tout diviseur premier de $m$ , et si 4 divise $m$ alors 4 divise $(a-1)$ .

Cas particulier : $x_{i+1} \ = \ (x_i \ + \ 1) \ mod \ m$ ; malheureusement, un examen appronfondi montre que cette suite n'a pas vraiment un « look »aléatoire . . . On veut généralement que les $n$ premiers éléments de la suite (avec $n << m$) soient à peu près également répartis dans l'intervalle $[0, m]$ . Voici quelques recettes de cuisine pour éviter la mésaventure précédente.

**1)** Choisir $m$ très grand (voir aussi la recette n°**7**).

**2)** Si $m$ est de la forme $2^k$ alors choisir $a \equiv 8 \ [5]$ .

**3)** Si $m$ est de la forme $10^k$ alors choisir $a \equiv 21 \ [200]$ .

**4)** Choisir de préférence $a$ tel que $\frac{m}{100} \ \leq \ a \ \leq \ m \ - \ \frac{m}{100}$

**5)** Eviter que l'écriture de $a$ en binaire ou en décimal ne présente des motifs répétitifs.

**6)** La condition « $b$ premier avec $m$ » est remplie efficacement avec $c = 1$ ou $c = a$ .

**7)** *Nota* : les chiffres de droite des $x_i$ ne sont pas vraiment aléatoires ; les chiffres de gauche sont mieux répartis. Par conséquent, si $m$ est plus grand que l'intervalle $[0, Max]$ dans lequel on souhaite engendrer des nombres aléatoirement, *ne pas quotienter par un modulo* mais utiliser la partie entière de $(\frac{x_i}{m} \times Max)$ .

Pour plus de détails, et des recettes encore plus riches, les gourmands peuvent consulter le chapitre 3 de « the art of computer programming » de D.E. Knuth (vol.2 = « seminumerical algorithms ») mais c'est, pour le moins, parfois indigeste . . .