Annexe B : Tirage de nombres aléatoires

Les ordinateurs étant des machines déterministes, leur demander de produire des nombres réellement aléatoires n'est pas du tout trivial. Il existe plusieurs solutions de différentes qualités et donc de coûts différents. Pour les domaines dans lesquels le caractère réellement aléatoire est crucial, comme la cryptographie (les clefs de chiffrement doivent être parfaitement aléatoires pour garantir une sécurité maximale), on peut connecter l'ordinateur à un appareil spécifique qui génère des nombres aléatoires. Pour d'autres domaines, il est possible de se contenter de nombres dits « pseudo-aléatoires » et d'utiliser une méthode algorithmique, sans appareil autre que l'ordinateur lui-même. On utilise alors une fonction appelée générateur de nombres pseudo-aléatoires.

Un générateur de nombres pseudo-aléatoires peut être vu comme une suite avec un u_0 et une relation de récurrence $u_{n+1} = f(u_n, u_{n-1}, u_{n-2} \ldots)$ suffisamment compliquée pour que les valeurs u_1, u_2, u_3 … *semblent* sans rapport entre elles, et *semblent* donc aléatoires. En réalité, la fonction f est bien déterminée et les nombres ne sont donc pas réellement aléatoires. On dit qu'ils sont pseudo-aléatoires.

La bibliothèque **stdlib** du langage C fournit un générateur « basique » de nombres pseudo-aléatoires. Ce générateur est très loin d'être parfait, mais il suffira pour nos besoins. Les fonctions à appeler sont **srand** (appelée une seule fois au début du programme pour initialiser le générateur) et **rand** (appelée à chaque fois que l'on a besoin d'un nombre aléatoire).

- La fonction `srand(unsigned int graine)` prend en paramètre un entier appelé « graine », qui va être utilisé comme u_0 pour initialiser la suite de nombres. On pourrait demander à l'utilisateur de saisir au clavier un entier de son choix et l'utiliser comme graine. Mais souvent, on préfère fabriquer automatiquement un entier à partir de l'heure précise à laquelle l'exécution est lancée, et utiliser cet entier comme graine. De cette façon, on obtiendra automatiquement des tirages différents à chaque exécution, ce qui est en général souhaitable (sauf lorsqu'on débogue…).
- La fonction `rand()` simule une loi uniforme et renvoie un entier pseudo-aléatoire compris entre 0 et RAND_MAX. RAND_MAX est une constante définie dans stdlib.h, sa valeur dépend des implémentations mais elle est souvent égale à 0x7FFF (en hexadécimal), ie. le plus grand entier relatif codable sur 4 octets. Pour obtenir un entier dans une plage donnée, il faudra donc convertir la valeur donnée par rand().

Exemple :

```
#include <stdlib.h>   // pour srand() et rand()
#include <time.h>     // pour time()

int main() {
  int aleatoire;
  int min = 1, max = 31;
  int nombreDeValeurs = max - min + 1 ;

  /* Initialisation du générateur, à ne faire qu'une fois dans le programme */
  srand((unsigned int)time(NULL));

  /* Tirage de 100 entiers aléatoires compris entre min et max inclus */
  for (int i = 0; i < 100; i++) aleatoire = (rand() % nombreDeValeurs) + min;

  return 0;
}
```