Travaux Dirigés d'informatique n°8

Cours d'Informatique de Première Année

—DEUG Sciences S2-SM/STPI—

Nombres aléatoires

Nous allons découvrir des fonctions permettant de générer des nombres aléatoires. Après s'être familiarisés avec ces fonctions, nous les utiliserons pour effectuer une approximation du nombre π par la méthode de Monte-Carlo (cf. cours). Enfin, nous programmerons un jeu contre l'ordinateur.

Plusieurs familles de fonctions aléatoires sont disponibles dans la bibliothèque *stdlib.h*. Nous allons utiliser la famille de fonctions **rand48** et plus particulièrement les fonctions suivantes :

- **srand48** sert à initialiser le processus de génération aléatoire. Cette fonction attend un paramètre de type *long int*. **Attention** : la suite de nombres générés après l'initialisation par `srand48` est entièrement dépendante du nombre qu'on lui a passé en paramètre. Cela signifie que pour deux initialisations identiques, on obtiendra les mêmes suites de nombres aléatoires ! On pourra utiliser la fonction `time` de la librairie *time.h* pour initialiser le tirage aléatoire, par exemple :

  ```
  srand48( time(NULL) );
  ```

- **lrand48** renvoie un nombre entier positif aléatoire de type *long int* appartenant à l'intervalle $[0, 2^{31}[$;
- **mrand48** renvoie un nombre entier aléatoire de type *long int* appartenant à l'intervalle $[-2^{31}, 2^{31}[$;
- **drand48** renvoie un nombre décimal aléatoire de type *double* appartenant à l'intervalle $[0.0, 1.0[$.

Dans les exercices suivants, on prendra soin d'initialiser le tirage aléatoire au début du `main` à l'aide de `srand48` avant d'appeler les fonctions demandées.

▸ Exercice 1.

1. *Écrire une fonction* `DixAlea` *qui initialise le tirage aléatoire avec un nombre fixé quelconque puis affiche dix nombres entiers tirés aléatoirement à l'aide de la fonction* `lrand48`*. Que constate-t'on en appelant plusieurs fois cette fonction ?*
2. *Corriger l'initialisation afin que le tirage aléatoire soit "correct".*

▸ **Exercice 2.** *La fonction* `lrand48` *renvoie un nombre entier compris dans un intervalle fixé (voir plus haut). Si cet intervalle ne nous convient pas, il peut être utile de disposer d'une fonction permettant de choisir cet intervalle.*
Pour cela, écrire la fonction `AleaEntier` *(utilisant* `lrand48`*) qui prend en argument deux entiers* min *et* max *tels que* $min < max$ *et renvoie un nombre entier aléatoire compris dans l'intervalle* $[min, max]$.

▸ **Exercice 3.** *Nous allons écrire une fonction* `ApproxPi` *qui prend en argument un nombre entier positif* n *et qui renvoie une approximation du nombre* π *évaluée après* n *tirages aléatoires successifs (méthode de Monte-Carlo).*
Nous procéderons de la façon suivante :

1. *Tirons aléatoirement un point du plan d'abscisse et d'ordonnée comprises dans l'intervalle* $[0.0, 1.0[$.
2. *Déterminons s'il appartient au cercle trigonométrique. Dans ce cas on itère un compteur* j.
3. *Répétons le processus à partir de l'étape 1, et cela* n *fois.*
4. *Une évaluation de* π *est donnée par* $\frac{j}{n} \times 4$.

Plus le nombre d'itérations est grand, plus l'approximation est correcte.

▸ **Exercice 4.** *(pour les plus rapides !)*
Réaliser un programme permettant de jouer contre l'ordinateur au ***jeu du nombre caché****. Le principe du jeu est le suivant : l'ordinateur choisit aléatoirement un nombre entier positif dans une fourchette déterminée. Ensuite, l'ordinateur et le joueur essaient successivement de deviner ce nombre, les seules indications dont ils disposent étant "Plus grand" ou "Plus petit". Le premier qui trouve le nombre a gagné.*
On pourra utiliser la fonction `AleaEntier` *écrite dans l'exercice 2 pour faire jouer l'ordinateur.*