Travaux Dirigés de C n°2

Cours de langage et programmation

—Licence d'Informatique—

▸ Exercice 1. Cryptage

On veut transmettre des messages secrets. Le cryptage du message se réalise avec une "clef secrète", que possèdent l'émetteur et le destinataire. La clef secrète sert au destinataire pour décrypter le message. De telles techniques de cryptage, où l'expéditeur et le destinataire partagent le même secret, sont dites "à clef symétrique".

Le principe utilisé pour crypter un message m est en utilisant la clef secrète s est le suivant: on calcule $m \oplus s$, qu'on envoie. Le récepteur calcule $(m \oplus s) \oplus s = m \oplus (s \oplus s) = m \oplus 0 = m$, et retrouve ainsi le message d'origine.

1. Écrire une fonction prenant en argument deux tableaux de caractères contenant respectivement le message et la clef secrète, et calculant le message crypté dans le tableau de caractères contenant le message d'origine.
2. Écrire un programme cryptant un message, et le décryptant. Cette manière de crypter bien connue a effectivement été utilisée. Quels sont ses défauts ?

▸ Exercice 2. Crible d'Ératostène

On veut implémenter l'algorithme du crible d'Ératostème qui calcule tous les nombres premiers inférieurs à une borne N. Pour cela, on part de l'ensemble de tous les entiers inférieurs à N, on retire tous les multiples de 2, puis ceux de 3, de 5 et ainsi de suite.

L'ensemble des entiers inférieurs à N sera représenté par un ensemble de bits. Le premier bit de l'ensemble représente l'entier 2, le second l'entier 3, et ainsi de suite.

Au départ, dans cet ensemble, tous les bits sont à 0. On passe le bit à 1 quand on s'aperçoit que l'entier correspondant n'est pas premier. À la fin de l'algorithme, il suffit d'afficher les valeurs des entiers dont les bits correspondants sont encore à 0 : ce sont les nombre premiers inférieurs à N.

1 Les types réels

▸ Exercice 3. Ecriture binaire des types réels

Dans cet exercice on s'intéresse aux types réels que l'on suppose codés selon le format IEEE avec les tailles suivantes :

- `float` : 32 bits dont 1 bit de signe, 8 bits d'exposant, 23 bits de mantisse,
- `double` : 64 bits dont 1 bit de signe, 10 bits d'exposant, 53 bits de mantisse,
- `long double` : 96 bits dont 1 bit de signe, 32 bits d'exposant, 63 bits de mantisse.

La fonction de l'exercice précédant donne la représentation binaire d'une variable réelle quelconque, indépendemment du mode *little/big endian*, mais sans marquer les séparations entre *signe*, *exposant* et *mantisse*. On veut améliorer ça.

1. Modifier la fonction `UCharToBin` en `<void UCharToBin(uchar a, int sep)>` où `sep` marque la position d'un caractère séparateur (par exemple `'|'`).
   On l'utilisera de la sorte : `UCharToBin(a,3)`⇒`xxx|xxxxx`.

2. Adapter la fonction `ReelIEEE` de sorte que des séparateurs délimitent automatiquement, et pour chaque type, les parties *signe*, *exposant* et *mantisse* :
   - `float  x=1.;` ⇒`0|01111111|00000000000000000000000`
   - `double y=1.;` ⇒`0|0111111111|0000000000000000000000000000000000000000000000000000000000000000`

Si on ne connaissait pas *a priori* les tailles respectives des exposants et mantisses pour chacun des trois types réels, comment pourrait-on les découvrir ?

▸ Exercice 4. Approximation en calcul numérique

On sait, mathématiquement, que $\sum_{i=1}^{N} \frac{1}{i^2} \to \frac{\pi^2}{6}$ quand $N \to +\infty$.

Soit N une valeur entière, suffisamment grande (essayer avec 10^3 puis 10^6 puis 10^9).
Écrire un programme calculant cette somme de six manières:

- pour chacun des trois types réels,
- en faisant croître i, et en le faisant décroître.

Conclusion ?
note : valeur de π avec ses 30 premières décimales exactes : 3.14159265358979323846264338327 9