

Bases de la Programmation

TP 3 – Tableaux

V. Padovani, PPS - IRIF

Exercice 1 : Crible d’Ératosthène

Le *crible d’Ératosthène* est une méthode permettant de calculer tous les nombres premiers inférieurs à un entier *SUP* donné. Son principe est le suivant :

- On écrit la liste de tous les entiers supérieurs ou égaux à 2 et inférieurs à *SUP*.
- On effectue un parcours de cette liste. À chaque entier *i* rencontré, on supprime de la liste toutes les entiers strictement plus grands que *i* et multiples de *i* encore présents.

En fin de traitement, les nombres encore présents dans la liste sont tous les nombres premiers inférieurs à *SUP*. Voici par exemple l’état de la liste après la rencontre de 2, puis 3 – à la dernière étape, la rencontre de 5 sera suivie de l’effacement de tous les multiples de 5 encore présents (*e.g.* 25), etc.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		2	3	~~4~~	~~5~~	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~	15
		↑													
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		2	3		5		7	~~8~~	~~9~~		11		13	~~14~~	~~15~~
		↑													
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		2	3		5		7				11		13		
					↑										

etc.

Implémentez cette méthode en C. Votre programme devra isoler et tous les nombres premiers strictement inférieurs à une constante *SUP* de valeur supérieure ou égale à 2. En fin de traitement, ces nombres seront inscrits dans un tableau de taille *SUP* : chaque position *i* du tableau contiendra *i* si ce nombre est premier, et 0 sinon (la notion d’“effacement d’un nombre *i* de la liste” de la méthode précédente sera donc traduite en l’inscription d’un 0 à la position *i*). Les valeurs non nulles du tableau seront simultanément affichées pendant sa construction :

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 ...

Rappel. Un entier *j* est multiple de *i* si et seulement si *j* modulo *i* est égal à 0. L’opérateur de modulo s’écrit %.

Exercice 2 : ROT-13

ROT-13 est un algorithme simpliste de chiffrement consistant à remplacer chaque caractère alphabétique d’une chaîne par le caractère qui est 13 positions plus loin dans l’alphabet, *modulo* 26. Par exemple, la chaîne “bonjour” est codée en ROT-13 par “obawbhe”.

Ecrire un programme lisant une chaîne de caractères au clavier, puis affichant l’encodage de cette chaîne en ROT-13. On se limitera au cas où la chaîne entrée ne contient ni majuscules, ni caractères accentués, tout en pouvant contenir des espaces et des caractères de ponctuations qui seront laissés tels quels. Par exemple, l’entrée de “bonjour, le c.” affichera “obawbhe, yr c.”.

Exercice 3 : Tirage aléatoire sans répétitions

Considérons un entier $n > 0$ et un entier $k \in [0, \dots, n[$, on souhaite extraire aléatoirement k valeurs de la suite $(0, \dots, n - 1)$, mais sans jamais extraire deux fois une même valeur.

- Comment faire ?
- En application : écrire un programme déclarant un tableau à **SUP** cases, où **SUP** sera une constante définie au début du code. Le tableau sera d'abord rempli par la suite de valeurs $0, 1, 2, \dots, (\text{bSUP} - 1)$. Le programme devra ensuite afficher toutes ces valeurs dans un ordre aléatoire, et sans jamais afficher deux fois la même valeur.
- Trouver le moyen de généraliser le traitement au cas suivant. Le programme devra cette fois déclarer une matrice de taille **H** $\times$ **L**, et commencer par remplir cette matrice par des zéros.

Ensuite, pour un certain entier **k** positif et inférieur ou égal à **H** $\times$ **L** (choisi par l'utilisateur, écrit en dur dans le code, peu importe), le programme devra placer exactement **k** fois la valeur 1 dans cette matrice à des positions choisies aléatoirement – sans jamais tirer deux fois la même position.