Chapitre 4

Les tableaux

4.1 Tableaux Simples

EX. 69 Écrire un programme qui lit une série de valeurs lues au clavier et qui ensuite les affiche dans l'ordre inverse. La fin de la série est marquée par la lecture de la valeur -1 et on supposera qu'il n'y aura jamais plus de 100 valeurs lues.

EX. 70 Écrire la fonction `int find_max(int t[],int n)` qui renvoie l'indice du plus grand des éléments du tableau `t`, qui est de taille `n`. On supposera que le tableau n'est pas vide.

EX. 71 Écrire une fonction `symétrie_vecteur` qui opère une symétrie sur les éléments d'un vecteur V de dimension n, c'est-à-dire échange la première et la dernière composante du vecteur, la deuxième et l'avant-dernière,... $(v_1, v_2, \ldots, v_n)$ devient $(v_n, v_{n-1}, \ldots, v_1)$. Vous ne pouvez pas utiliser de vecteur de travail.

EX. 72 Écrire une fonction `produit_scalaire` qui renvoie le produit scalaire de deux vecteurs de longueur n donnés en paramètre (u, v) :

$$\sum_{i=1}^{n} u_i \cdot v_i$$

EX. 73 Écrire une fonction qui tasse un vecteur d'entiers : tous les zéros se retrouvent à la fin, l'ordre des autres nombres ne doit pas être conservé.

Exemple : (0,1,0,2,0,0,3,4) devient (4,1,3,2,0,0,0,0).

EX. 74 Écrire une fonction qui tasse un vecteur d'entiers : tous les zéros se retrouvent à la fin et l'ordre des autres nombres n'est pas modifié.
Exemple : (1,9,0,0,7,8,0,4) devient (1,9,7,8,4,0,0,0).

EX. 75 Soient deux vecteurs POS et DON de longueur n. Écrire une fonction `imprime_ind` qui affiche les éléments de DON dans l'ordre indiqué par POS (qui contient les entiers de 1 à n).

Exemple ($n = 4$)

DON	9	6	8	2

POS	2	4	3	1

donne $6\ \ 2\ \ 8\ \ 9$.

EX. 76 (QUESTION DE L'EXAMEN D'AOÛT 2006) Un nombre premier est un entier naturel strictement supérieur à 1, n'admettant que deux entiers naturels diviseurs distincts : 1 et lui-même.

On vous demande d'écrire un crible, c'est-à-dire une fonction qui trouve tous les nombres premiers inférieurs à un entier n donné.

La fonction devra avoir la signature `void crible(int t[],int n)` et remplira le tableau `t` de tous les nombres premiers inférieurs à `n`. La fin de la liste de nombres premiers sera marquée par la valeur sentinelle -1. Le tableau `t` sera supposé suffisamment grand pour stocker `n+1` entiers.

Par exemple, après l'appel `crible(t,20)`, le tableau `t` aura le contenu suivant :

2	3	5	7	11	13	17	19	-1	$\cdots$

L'évaluation de votre solution sera principalement basée sur son efficacité.

EX. 77 Le coefficient binomial C_n^p est la valeur $\frac{n!}{p!(n-p)!}$. Les coefficients binomiaux peuvent être calculé itérativement, sans effectuer ni multiplication, ni division, au moyen des formules suivantes :
$C_0^0 = 1$
$C_1^0 = 1$
$C_n^p = 0$ si $p > n$ ou si $p < 0$
$C_n^p = C_{n-1}^{p-1} + C_{n-1}^p$ sinon.
La représentation classique se fait alors sous la forme d'un triangle, appelé Triangle de Pascal.

```
ligne 0      1
ligne 1      1   1
ligne 2      1   2   1
ligne 3      1   3   3   1
ligne 4      1   4   6   4   1
ligne 5      1   5   10  10  5    1
ligne 6      1   6   15  20  15   6   1
ligne 7      1   7   21  35  35   21  7   1
```

Les coefficients C_n^k, $0 \leq k \leq n$ figurent à la n^{e} ligne. Le triangle est construit en plaçant des 1 aux extrémités de chaque ligne et en complétant la ligne en reportant la somme de deux nombres adjacents de la ligne supérieure : celui juste au dessus, et celui à sa gauche.

Écrire une fonction `void pascal(vecteur v, int n)` qui renvoie dans le vecteur `v` la n^{e} ligne du triangle de Pascal. Le tableau sera supposé suffisamment grand.

EX. 78 Écrire une fonction `gliss_1_g` effectuant un glissement circulaire gauche d'une position sur un vecteur.

Écrire une fonction `gliss_1_d` effectuant un glissement circulaire droit d'une position sur un vecteur.

: 40 min.

EX. 79 Écrire une fonction qui effectue un glissement circulaire gauche de K positions sur un vecteur. Chaque élément ne doit être déplacé qu'une seule fois et votre fonction ne doit pas utiliser de vecteur de travail.

4.2 Tableaux à Deux Dimensions : les Matrices

EX. 80 Écrire :

1. une fonction `init_mat_m_n` qui initialise une matrice $m \times n$ à composantes entières (m et n sont des constantes données).

2. une fonction `imprime_mat_m_n` qui imprime une telle matrice.

EX. 81 Écrire une fonction `trace` à valeur réelle qui calcule la trace d'une matrice A de dimension $n \times n$ donnée en paramètre :

$$\texttt{trace}(A) = \sum_{i=1}^{n} A_{ii}$$

EX. 82 Écrire une fonction qui effectue le produit de la matrice $A(l \times m)$ par la matrice $B(m \times n)$.

EX. 83 Rappel : une matrice A_{ij} (i de 1 à n, j de 1 à n) est dite symétrique si elle est égale à sa transposée, c'est-à-dire si :

$\forall i, j : 1 \leq i, j \leq n : A_{ij} = A_{ji}$

et antisymétrique si :

$\forall i, j : 1 \leq i, j \leq n : A_{ij} = -A_{ji}$.

Écrire les fonctions `symetrie` et `antisymetrie` qui testent respectivement la symétrie et l'antisymétrie d'une matrice carrée.

: 40 min.

EX. 84 Écrire une fonction rotation qui effectue une rotation, en place, de +90° (dans le sens trigonométrique) dans une matrice carrée $n \times n$.

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \text{ devient : } \begin{pmatrix} 3 & 6 & 9 \\ 2 & 5 & 8 \\ 1 & 4 & 7 \end{pmatrix}$$
