

### Exercice 1

Ecrire un algorithme qui prend de l'utilisateur deux entiers puis affiche le signe du produit sans faire la multiplication et le signe de la somme sans faire l'addition.

### Exercice 2

Ecrire un algorithme qui prend une suite d'entiers inférieurs à 10 jusqu'à ce que l'utilisateur introduit la valeur 0. L'algorithme doit alors afficher la somme, le produit et la moyenne des ces nombres.

### Exercice 3

Faire l'algorithme qui prend un entier  $n > 0$  de l'utilisateur, et donne comme résultat la somme:

$$S = \begin{cases} 1 + 3 + 5 + \dots + n & \text{si } n \text{ est un nombre impair} \\ 2 + 4 + 6 + \dots + n & \text{si } n \text{ est un nombre pair} \end{cases}$$

Il ne faut pas accepter une valeur négative de  $n$  et le programme doit donner un seul résultat selon la valeur de  $n$  et non pas les deux sommes.

### Exercice 4

Une classe contient 200 places réparties en 10 rangées et 20 colonnes. Chaque élève possède un numéro entre 1 et 200. Lors d'un examen, on place les élèves sur les places selon leur numéro en commençant par la première rangée puis la deuxième et ainsi de suite comme le montre le tableau suivant:

Rangée 1	1	2	...	20
Rangée 2	21	22	...	40
...	...	...	...	...
Rangée 10	181	182	...	200
	Colonne 1	Colonne 2	...	Colonne 20

On demande de faire un algorithme qui prend de l'utilisateur un entier entre 1 et 200 puis affiche dans quelle rangée et dans quelle colonne l'élève doit se placer. Par exemple, si  $n = 35$ , l'élève doit se placer sur la 2ème rangée et la 15ème colonne.

### Exercice 5

Ecrire un algorithme qui remplit un tableau T de 100 entiers et calcul la somme S des éléments durant la phase de remplissage. Une fois le remplissage terminé, le programme doit calculer la moyenne M des éléments puis il doit remplir un tableau F de 200 éléments de sorte que:

$$F[0] = T[0] - M, F[1] = T[0] + M$$

$$F[2] = T[1] - M, F[3] = T[0] + M$$

....

$$F[198] = T[99] - M, F[199] = T[99] + M$$

### Exercice 6

Dans cet algorithme on suppose qu'on a un tableau T de 100 éléments entiers remplis par l'utilisateur. On demande de rechercher l'élément le plus petit du tableau puis de retrancher sa valeur de chaque élément du tableau.

### Exercice 7

Faire l'algorithme qui prend un entier impair  $n > 0$  de l'utilisateur, et donne comme résultat la somme:

$$S = 2 + \frac{2^3}{3!} + \frac{2^5}{5!} + \frac{2^7}{7!} + \dots + \frac{2^n}{n!}$$

Il ne faut pas accepter une valeur paire ou négative de  $n$ .

### Exercice 8

- Que donne l'instruction:  $1243 \bmod 10$ .
- Que donne l'instruction:  $1243 \text{ div } 10$ .
- Ecrire un algorithme qui prend de l'utilisateur un entier  $n > 0$  puis affiche la somme de ses chiffres. Par exemple, si  $n = 1243$ , l'ordinateur doit afficher 10 qui représente la somme  $3 + 4 + 2 + 1$ .

### Exercice 9

Ecrire un algorithme qui prend 50 notes quelconques de l'utilisateur et affiche la moyenne des notes qui sont comprises entre 10 et 20 (bornes incluses).

### Exercice 10

Le but de cet exercice est de faire un algorithme qui permet de compter le nombre de voyelles dans une chaîne. Le travail consiste à:

- Ecrire une fonction **Voyelle** qui retourne vrai si le caractère C donné en paramètre est une voyelle (On rappelle que les voyelles sont a, e, i, o, u et y).
- Sachant que toute chaîne de caractères se termine par le caractère '\0', écrire une fonction **Longueur** qui retourne le nombre de caractères dans une chaîne CH.
- Ecrire une fonction **nbVoyelle** qui retourne le nombre de voyelles dans une chaîne de caractères CH donnée en paramètre. On utilisera la fonction **longueur** pour savoir le nombre de caractères dans la chaîne.
- Compléter l'algorithme qui va prendre une chaîne de l'utilisateur, puis affiche le nombre de voyelles ainsi que le nombre de caractères qui ne sont pas des voyelles dans cette chaîne.

### Exercice 11

Le calcul de la racine carrée X d'un nombre réel positif A est fait par approximations successives en utilisant la relation de récurrence suivante:

$$X_j = \begin{cases} A & \text{si } j = 1 \\ \frac{1}{2} \left( X_{j-1} + \frac{A}{X_{j-1}} \right) & \text{si } j > 1 \end{cases}$$

Lorsque  $j$  augmente, la valeur de  $X_j$  sera plus proche de  $\sqrt{A}$ .

On demande de faire une fonction récursive **racine** qui ayant la valeur de A et de  $j$  donne l'approximation d'ordre  $j$  de la racine de A.

### Exercice 12

On rappelle que le produit scalaire de deux vecteurs  $V_1(x_1, x_2, x_3, \dots, x_n)$  et  $V_2(y_1, y_2, y_3, \dots, y_n)$  est donné par:

$$x_1y_1 + x_2y_2 + x_3y_3 + \dots + x_ny_n$$

Si le résultat est nul, alors les vecteurs seront perpendiculaires.

- a. Ecrire une fonction *Produit* qui calcule le produit scalaire de deux vecteurs d'entiers U et V de même dimension  $n$ .
- b. Ecrire une fonction *Perpendiculaire* qui teste si deux vecteurs U et V de même dimension  $n$  sont perpendiculaires.

### Exercice 13

- a. Ecrire une procédure *LireTab* qui permet de remplir un tableau T de  $n$  éléments entiers donnés par l'utilisateur de sorte que les éléments soient donnés par ordre croissant. La procédure ne doit pas accepter des valeurs inférieures aux valeurs déjà introduites. On suppose en plus que  $n > 1$ .
- b. Ecrire une procédure *EcrireTab* qui permet d'afficher les éléments d'un tableau de  $n$  éléments.
- c. Ecrire une procédure *Fusion* qui permet de prendre les éléments de deux tableaux triés et de les mettre dans un troisième tableau de sorte que ce dernier soit encore trié. (Il ne faut pas remplir le tableau puis le trier après).
- d. Utiliser les procédures précédentes pour écrire un algorithme permettant de prendre de l'utilisateur deux tableaux de 100 entiers triés puis afficher l'ensemble formé par les éléments des deux tableaux d'une manière triée.

### Exercice 14

Ecrire un algorithme qui remplit un tableau successivement des nombres:  
 $1!, 2!, 3!, \dots, 100!$

### Exercice 15

Ecrire un algorithme qui vérifie si les cases d'indice impair dans un tableau de  $n$  éléments sont classées dans l'ordre croissant. (On suppose que le tableau est rempli et il n'est pas nécessaire de le remplir).

### Exercice 16

Ecrire un algorithme qui remplit un tableau successivement des nombres:  
 $5, 5^2, 5^3, \dots, 5^{100}$

### Exercice 17

Ecrire un algorithme qui affiche les cases divisibles par leur indice dans un tableau de  $n$  éléments, c'est-à-dire de voir si T[1] est divisible par 1, T[2] par 2, T[3] par 3 et ainsi de suite jusqu'à T[n] par  $n$ . (On suppose que le tableau est rempli et il n'est pas nécessaire de le remplir).

### Exercice 18

Ecrire un algorithme qui remplit un tableau contenant 100 éléments, de sorte que les 5 premières cases contiennent le nombre 1, les 5 cases suivantes contiennent le nombre 2, les 5 cases qui viennent après contiennent le nombre 3 et ainsi de suite. En d'autres termes, le tableau doit contenir les nombres:  
 $1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, \dots$

### Exercice 19

Ecrire un algorithme qui remplit un tableau T de 100 entier. Ensuite, l'algorithme doit calculer la somme:  
 $S = T[1] - T[2] + T[3] - T[4] + T[5] - T[6] + \dots - T[100]$

### Exercice 20

Faire l'algorithme qui prend une valeur entière  $n > 0$  de l'utilisateur, et donne comme résultat la somme:

$$S = \frac{n}{2 \times 3} + \frac{n^2}{3 \times 4} + \frac{n^3}{4 \times 5} + \dots + \frac{n^n}{(n+1)(n+2)}$$

Il ne faut pas accepter une valeur négative de  $n$ .

### Exercice 21

- Que donne l'instruction:  $(43 \bmod 10) \times 10 + 43 \text{ div } 10$ .
- En s'inspirant de l'équation précédente, écrire un algorithme qui prend de l'utilisateur un nombre entier  $n$  entre 1000 et 9999, puis affecte à une variable  $p$  le nombre inversé et affiche ensuite la valeur de  $p$ . Par exemple, si  $n = 1243$  le programme doit affecter à  $p$  la valeur 3421 puis il doit afficher le nombre 3421. (On suppose que l'utilisateur donne des valeurs acceptables et par suite il n'est pas demandé de faire la vérification sur la validité de ces valeurs. La note complète sera donnée à ceux qui utilisent une répétition et non pas une seule instruction).

### Exercice 22

Ecrivez un algorithme qui lit deux valeurs entières (A et B) du clavier et qui indique si le produit de A et B est négatif ou non sans faire la multiplication.

### Exercice 23

Faire l'algorithme qui remplit un tableau de 100 éléments par les factoriels des nombres allant de 1 à 100, c'est-à-dire:

$$T[1] = 1!, T[2] = 2!, T[3] = 3!, \dots, T[100] = 100!$$

Il faut signaler que chaque case du tableau se déduit de la case précédente.

### Exercice 24

On donne un tableau de 100 éléments et on suppose que les éléments du tableau sont introduits, donc il n'est pas demandé de faire le remplissage du tableau. Ecrire un algorithme qui compte le nombre des éléments du tableau qui sont divisibles par 4.

### Exercice 25

Le but de cet exercice est de construire et d'afficher le triangle de Pascal en calculant les coefficients binomiaux (sans utiliser un tableau). Le triangle de Pascal a la forme ci-dessous:

$$\begin{array}{ccccccc} & & & & & & 1 \\ & & & & & & 1 & 1 \\ & & & & & 1 & 2 & 1 \\ & & & & 1 & 3 & 3 & 1 \\ & & 1 & 4 & 6 & 4 & 1 \end{array}$$

L'élément qui se trouve à la ligne  $i$  et la colonne  $j$  représente le coefficient binomial  $C_i^j = \frac{i!}{j!(i-j)!}$ . Les indices des lignes et des colonnes commencent par 0.

Donc le triangle précédent est équivalent à:

$$\begin{array}{cccccc}
C_0^0 & & & & & \\
C_1^0 & C_1^1 & & & & \\
C_2^0 & C_2^1 & C_2^2 & & & \\
C_3^0 & C_3^1 & C_3^2 & C_3^3 & & \\
C_4^0 & C_4^1 & C_4^2 & C_4^3 & C_4^4 & 
\end{array}$$

- a. Ecrire une fonction réursive **Fact** qui calcule la factorielle d'un entier  $n$ , sachant que:

$$n! = \begin{cases} 1 & \text{si } n = 0 \\ n \times (n-1)! & \text{si } n > 0 \end{cases}$$

On suppose que le nombre  $n$  n'est pas négatif.

- b. Ecrire une fonction **Coef** qui calcule le coefficient  $C_i^j = \frac{i!}{j!(i-j)!}$  en utilisant la fonction de la partie a. Cette fonction doit avoir  $i$  et  $j$  comme paramètres.
- c. Ecrire une procédure **Ligne(k:entier)** qui affiche tous les coefficients de la ligne  $k$ , c'est-à-dire les éléments:

$$C_k^0 \quad C_k^1 \quad \dots \quad C_k^k$$

(On suppose dans ce cas que l'exécution successive de l'instruction écrire affiche les résultats sur la même ligne)

- d. Ecrire une procédure **Triangle(p:entier)** qui affiche  $p$  lignes du triangle de Pascal.
- e. Ecrire un algorithme qui prend le nombre de lignes  $n$  de l'utilisateur et qui affiche le triangle de Pascal correspondant.

### Exercice 26

- a. Ecrire une fonction **Nchiffres** qui fournit le nombre de chiffres existants dans un entier  $a$  (positif ou négatif), donné en paramètre.
- b. Utiliser la fonction précédente pour écrire une procédure **Tri** qui permet de trier un tableau  $T$  de  $N$  éléments par ordre croissant suivant le nombre de chiffres de chacun de ses éléments.  
Par exemple : 1, 32, 31, 235, -1223, 12236, -54566666 sont triés par ordre croissant suivant le nombre de chiffres.

### Exercice 27

Un polynôme  $P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$  peut être écrit en utilisant la méthode de Horner sous la forme:

$$P(x) = (((\dots(a_n x + a_{n-1})x + a_{n-2})x + \dots + a_1)x + a_0$$

Ecrire un algorithme qui prend les valeurs de  $x$  et  $n$  puis les valeurs de  $a_n, a_{n-1}, \dots, a_0$  pour calculer la valeur de  $P(x)$  en utilisant la méthode de Horner. On commence par prendre la valeur de  $a_n$ , on calcule  $a_n x$ , puis on prend  $a_{n-1}$ , on l'ajoute à la somme et on multiplie le résultat par  $x$  pour avoir  $(a_n x + a_{n-1})x$  et ainsi de suite.

### Exercice 28

- a. On demande de faire une procédure appelée **minmax** qui donne les rangs des éléments minimal ( $rmin$ ) et maximal ( $rmax$ ) d'un tableau  $T$  entre deux indices  $n_1$  et  $n_2$ . La procédure a la forme:

- procedure minmax(T[Nmax], n1, n2, &rmin, &rmax: entier)*
- b. Ecrire une procédure qui permet d'échanger deux éléments d'un tableau  $T$  qui se trouvent respectivement aux indices  $i$  et  $j$ . La forme de cette procédure doit être:
- procedure echange(&T[Nmax], i, j: entier)*
- c. Ayant un tableau de taille  $n$ , on désire faire une procédure qui permet de le trier dans l'ordre croissant, en suivant la méthode détaillée ci-dessous:  
 On recherche dans le tableau entre les éléments 1 et  $n$ , en un seul passage, les rangs des éléments maximal et minimal. On permute ensuite l'élément minimal avec le premier élément du tableau et l'élément maximal avec le dernier élément du tableau. On répète le même travail entre les éléments 2 et  $(n - 1)$ , puis entre les éléments 3 et  $(n - 2)$ , ... jusqu'à terminer le tableau.  
 Utiliser les procédures précédentes pour créer une procédure *tri* qui applique la méthode expliquée pour trier le tableau  $T$  de taille  $n$ . La forme de cette procédure doit être:
- procedure tri(&T[Nmax], n: entier)*

### Exercice 29

Ecrire une fonction récursive permettant de calculer la suite définie par:

$$U_1 = 1, U_2 = 2 \text{ et } U_n = 2U_{n-1} - U_{n-2}$$

### Exercice 30

Ecrire un algorithme qui permet de remplir un tableau de  $n$  éléments par les valeurs 1, 5, 9, 13, ...,  $4n + 1$ .

### Exercice 31

Faire l'algorithme qui prend une valeur entière  $n$  de l'utilisateur, et donne comme résultat la somme:

$$S = \frac{1}{n} + \frac{2}{n+1} + \frac{3}{n+2} + \dots + \frac{n}{(2n-1)}$$

### Exercice 32

Faire l'algorithme qui donne la première valeur de  $n$  pour laquelle la somme précédente est supérieure à une valeur  $p$  donnée par l'utilisateur.

### Exercice 32

Faire l'algorithme qui remplit un tableau  $T$  de 100 entiers tels que:

$$T[1] = 1 \text{ et } T[i] = 2 \times T[i-1] + 5.$$

### Exercice 33

Faire l'algorithme qui compte le nombre d'éléments pairs dans un tableau. Le remplissage du tableau ainsi que sa dimension réelle (qui est inférieure à 100), devront être faites par l'utilisateur.

### Exercice 34

Faire l'algorithme qui demande de l'utilisateur un entier  $n$  et n'accepte que des nombres entre 5 et 20. Ensuite le programme doit afficher tous les entiers entre 1 et  $n$ .

**Exercice 35**

Faire un algorithme qui change les valeurs des éléments d'un tableau de sorte que les éléments aux positions impaires soient multipliés par 2 et les éléments aux positions paires soient divisés par 2.

**Exercice 36**

- Faire un algorithme qui demande de l'utilisateur deux nombres entiers  $n$  et  $p$  tout en exigeant que  $p$  soit positif ou nul. Ensuite, l'algorithme doit donner la valeur de  $n^p$  sans utiliser l'opérateur puissance.
- Que faut-il changer dans l'algorithme précédent pour considérer le cas où  $p$  est un nombre négatif.

**Exercice 37**

Le calcul du montant de la facture du téléphone mobile est basé sur le barème suivant:

Abonnement mensuel (AM) = 25\$.

Abonnement Clip (AC) = 6\$.

Connexion réseau (CR) = 0.07\$ / minute.

Taxe gouvernementale (TG) = 0.06\$ / minute.

TVA = 10% sur le total.

On demande de faire l'algorithme qui prend de l'utilisateur le nombre de minutes (NM) et s'il est abonné au service Clip (SC) puis affiche une facture détaillée avec le total à payer. (On suppose que l'utilisateur donne des valeurs acceptables et par suite il n'est pas demandé de faire le test sur la validité de ces valeurs).

**Exercice 38**

- Que donne l'instruction:  $1244534 \text{ div } 1000$ .
- Que donne l'instruction:  $1244 \text{ mod } 10$ .
- Ecrire un algorithme qui prend de l'utilisateur deux nombres entiers  $n$  et  $p$ , puis affiche le chiffre qui se trouve dans  $n$  à la  $p^{\text{ième}}$  position. Par exemple, si  $n = 11243$  et  $p = 2$ , on aura 2. Alors que si  $p = 0$  on aura la valeur 3 et si  $p > 4$  on aura la valeur 0.  
(On suppose que l'utilisateur donne des valeurs acceptables et par suite il n'est pas demandé de faire le test sur la validité de ces valeurs).

**Exercice 39**

Faire l'algorithme qui lit les valeurs de deux entiers  $n$  et  $x$  puis donne la valeur de la somme:

$$S = \frac{x \sin x}{1 \times 2} - \frac{x^2 \sin^2 x}{2 \times 3} + \frac{x^3 \sin^3 x}{3 \times 4} - \frac{x^4 \sin^4 x}{4 \times 5} + \dots + (-1)^{n+1} \frac{x^n \sin^n x}{n \times (n+1)}$$

On suppose que la fonction  $\sin(x)$  existe.

**Exercice 40**

On suppose qu'un tableau T de 100 entiers est rempli. Faire l'algorithme qui prend de l'utilisateur deux indices  $a$  et  $b$  (on suppose que  $b > a$ ) et permet de faire une rotation d'un pas à droite des éléments du tableau entre les indices  $a$  et  $b$ .

### Exercice 41

Ecrire une fonction qui donne l'indice de la première occurrence d'une chaîne  $t$  dans une chaîne  $s$  à partir d'un indice  $k$ . La forme de cette fonction doit être:

***Indice:entier(t:chaîne, s:chaîne, k:entier)***

On suppose que la fonction ***longueur(s)*** donnant le nombre de caractères de la chaîne  $s$  existe et peut être utilisée. Si la chaîne  $t$  ne se trouve pas dans la chaîne  $s$  à partir de l'indice  $k$ , la fonction doit donner -1 comme résultat.

### Exercice 42

Ecrire une fonction récursive permettant de calculer le terme  $U_n$  de la suite de Fibonacci définie par:

$$U_0 = 0, U_1 = 1 \text{ et } U_n = U_{n-1} + U_{n-2} \text{ pour } n \geq 2$$

### Exercice 43

- Ecrire une fonction ***Abs*** qui donne la valeur absolue d'un réel  $x$ .
- Ecrire une fonction ***TermeN(x:réel, a:réel)*** qui donne la valeur de l'expression:

$$\frac{2x + \frac{a}{x^2}}{3}$$

- Pour calculer la racine cubique d'un nombre réel  $a \geq 0$  on peut utiliser la méthode itérative suivante:

On prend une valeur initiale  $x_1 = \frac{a}{3}$  puis on cherche les termes successifs  $x_2, x_3, \dots$

$$\text{de la suite définie par } x_{n+1} = \frac{2x_n + \frac{a}{x_n^2}}{3}.$$

Après chaque itération, on compare  $x_{n+1}$  à  $x_n$  et on arrête lorsque  $|x_{n+1} - x_n| < 10^{-6}$ . Dans ce cas,  $x_n$  représente la racine cubique de  $a$ .

Ecrire alors la fonction qui donne la racine cubique d'un nombre  $a$  en utilisant la méthode décrite ci-dessus.

### Exercice 44

Ecrire un algorithme qui prend une chaîne  $s$  de l'utilisateur puis affiche une chaîne qui ressemble à la précédente mais en répétant les lettres dont l'indice est multiple de 3. Par exemple, si  $s = \text{"Le petit prince"}$ , la sortie doit être "LLe ppetiit pprinnc". On rappelle que les chaînes commencent par l'indice 0.

On suppose que la fonction ***longueur(s)*** qui donne le nombre de caractères de la chaîne  $s$  existe et on peut l'utiliser dans l'algorithme.

### Exercice 45

Ecrire une fonction récursive permettant de calculer la valeur de  $C_n^p$  sachant que:

$$C_n^p = \frac{n}{p} C_{n-1}^{p-1} \text{ et } C_n^0 = 1$$

### Exercice 46

- a. Ecrire une fonction qui recherche dans un tableau T de N éléments entiers le premier élément nul à partir d'un indice  $i$  et nous donne son indice. La forme de l'entête de cette fonction est:

**Fonction ZeroSuivant:entier(T[N]:entier, N:entier, i:entier)**

Par exemple, si le tableau est:

1	4	0	2	9	0	0	3	9	4	0	5
---	---	---	---	---	---	---	---	---	---	---	---

L'appel: ZeroSuivant(T,12,1) donne 3 et l'appel: ZeroSuivant(T,12,4) donne 6 et l'appel: ZeroSuivant(T,12,7) donne 7.

Si la fonction ne trouve pas un élément nul, elle doit donner -1 comme résultat.

- b. Ecrire une procédure qui affiche les éléments d'un tableau T de N éléments entiers entre deux indices  $i$  et  $j$ . L'entête de cette procédure doit être:

**Procédure Affiche(T[N]:entier, N:entier, i:entier, j:entier)**

- c. Ecrire un procédure qui utilise la fonction et la procédure précédentes pour afficher les éléments qui se trouvent entre deux zéros tout en indiquant le numéro de la série affichée. Par exemple, si on considère le tableau précédent, on doit avoir:

**Série 1:** 2, 9 **Série 2:** 3, 9, 4.

### Exercice 47

Les tarifs d'affranchissement d'une lettre sont les suivants:

En-dessous de 20 g : 2000 L.L.

A partir de 20 g mais en-dessous de 50 g : 3500 L.L.

A partir de 50 g : 5000 L.L.

Ecrire un algorithme qui lit le poids d'une lettre et affiche le tarif correspondant.

### Exercice 48

Ecrire un algorithme qui prend de l'utilisateur les coordonnées  $x$  et  $y$  d'un point et écrit sur l'écran une phrase indiquant dans quel quadrant se trouve le point. On suppose que l'utilisateur ne donne pas des valeurs nulles.

### Exercice 49

Faire l'algorithme qui prend une valeur entière  $n$  de l'utilisateur, et lui donne comme résultat la somme des nombres impaires inférieurs ou égaux à  $n$ .

$$S = 1 + 3 + 5 + \dots + p \text{ tel que } p \leq n$$

Si  $n \leq 0$  l'ordinateur doit afficher la valeur 0.

### Exercice 50

1. Faire l'algorithme qui prend une valeur entière  $n > 0$  de l'utilisateur, et lui donne comme résultat la somme:

$$S = \sum_{i=1}^n \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

Le programme ne doit pas accepter une valeur  $n \leq 0$ .

2. Qu'est-ce qu'il faut changer dans l'algorithme, si on veut calculer la somme:

$$S' = \sum_{i=1}^n (-1)^{i+1} \frac{1}{i} = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + (-1)^{n+1} \frac{1}{n}$$

### Exercice 51

On considère un tableau T de taille  $N_{\max} = 1000$ . Ecrire un algorithme qui prend de l'utilisateur la taille réelle utilisée du tableau et le remplit par des entiers. Ensuite le programme doit échanger les éléments 1 et 2, puis les éléments 3 et 4 et ainsi de suite tant que c'est possible jusqu'à la fin du tableau. Une fois l'échange est terminée, le programme doit afficher le tableau obtenu.

### Exercice 52

On suppose qu'un tableau T de taille  $N_{\max} = 1000$  est rempli de N éléments entiers. Faire l'algorithme qui met dans  $T[i]$  la valeur  $T[i] + T[i + 1]$  si i est impair et la valeur  $T[i] - T[i - 1]$  si i est pair (faite attention à ne pas utiliser dans la soustraction la nouvelle valeur de l'élément  $T[i - 1]$  au lieu de la valeur originale).

### Exercice 53

Faire un algorithme qui réserve un tableau de dimension  $N_{\max} = 1000$ , puis le remplit d'entiers tout en prenant de l'utilisateur le nombre d'éléments à remplir et en vérifiant que cette valeur est inférieure à  $N_{\max}$ . Une fois le remplissage est terminé, le programme doit chercher, parmi les éléments introduits, la position de la dernière occurrence d'une valeur entière X donnée par l'utilisateur. Si l'élément X ne figure pas dans le tableau on doit le signaler à l'utilisateur à travers un message.

### Exercice 54

Dans cet exercice on va utiliser deux procédures et une fonction. On suppose qu'on a une classe de 50 élèves.

- Ecrire une procédure qui permet de remplir dans un tableau  $N[50]$  les notes des 50 élèves. L'entête de la procédure étant:  
Procédure Lecture(&N[50]:réel)
- Ecrire une procédure, appelée "Affichage", qui permet d'afficher les notes du tableau  $N[50]$  avec la mention correspondante, en utilisant le barème ci-après:

Une note < 10	Echoué
$10 \leq$ la note $\leq$ 19	Réussi
Une note = 20	Excellent
- Ecrire une fonction appelée Max permettant de donner la note maximale du tableau  $N[50]$ .
- Ecrire un programme utilisant les procédures et la fonction des parties précédentes et permettant de prendre les notes des 50 élèves, d'afficher ces notes avec les mentions correspondantes ainsi que la plus haute note. Il n'est pas nécessaire de recopier le corps des procédures et de la fonction (c'est-à-dire les instructions entre les { }).

### Exercice 55

Ayant un tableau de taille  $n$ , on désire faire une procédure qui permet de le trier dans l'ordre croissant, en suivant la méthode détaillée ci-dessous:  
On recherche dans le tableau entre les éléments 1 et  $n$ , en un seul passage, les rangs des éléments maximal et minimal. On permute ensuite l'élément minimal avec le premier élément du tableau et l'élément maximal avec le dernier élément du tableau. On répète le même travail entre les éléments 2 et  $(n - 1)$ , puis entre les éléments 3 et  $(n - 2)$ , ... jusqu'à terminer le tableau.

- a. On demande de faire une procédure appelée **minmax** qui donne les rangs des éléments minimal (rmin) et maximal (rmax) d'un tableau **T** entre deux indices **n<sub>1</sub>** et **n<sub>2</sub>**. La procédure a la forme:  
*procédure minmax(T[n]:entier, n<sub>1</sub>:entier, n<sub>2</sub>:entier, &rmin:entier, &rmax:entier)*
- b. Utiliser la procédure précédente pour créer une procédure **tri** qui applique la méthode expliquée pour trier le tableau **T** de taille **n**. Cette procédure a la forme:  
*procédure tri(&T[n]:entier, n:entier)*

### Exercice 56

Ecrire une fonction entière qui retourne la place de la **dernière occurrence** de la valeur **x** dans le tableau **V** d'entiers **triés** à **N** éléments et retourne **-1** si **x** n'appartient pas à **V**. (**x**, **V** et **N** seront les paramètres de la fonction)

### Exercice 57

- a. Ecrire une procédure qui permet de remplir dans un tableau **N[50]** les notes de 50 élèves. L'entête de la procédure étant:

Procédure Lecture(&N[50]:réel)

- b. Ecrire une procédure, appelée "Affichage", qui permet d'afficher les notes du tableau **N[50]** avec la mention correspondante, en utilisant le barème ci-après:

Une note < 10	Echoué
10 ≤ la note ≤ 19	Réussi
Une note = 20	Excellent

On note qu'il ne faut pas tester directement la note aux valeurs 10, 19 et 20, mais il faut transformer la note en 0, 1 ou 2 et utiliser le tableau **Me** contenant 3 éléments, tels que **Me[0]**= "Echoué", **Me[1]**= "Réussi" et **Me[2]**= "Excellent".

(Si l'élève fait une comparaison directe avec les valeurs 10, 19 et 20 il aura la moitié de la note).

### Exercice 58

On veut "nettoyer" un texte (tableau de caractères) en éliminant les espaces en surplus: chaque fois que l'on trouve une suite d'espaces, il faudra les remplacer par un seul espace sauf au début de la phrase où il faudra les éliminer totalement. Ainsi la phrase:

" il fait très beau aujourd'hui "

deviendra:

"il fait très beau aujourd'hui ".

Faire la procédure correspondante.

### Exercice 59

Le but de cet exercice est de résoudre un système de trois équations à trois inconnues.

- a. Ecrire une fonction **Det2** qui calcule le déterminant d'ordre 2. On rappelle qu'un tel déterminant est donné par:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = a \times d - b \times c$$

Une telle fonction doit avoir la forme:

*fonction Det2(a:réel, b:réel, c:réel, d:réel):réel*

- b. Sachant qu'un déterminant d'ordre trois est donné par:

$$\begin{vmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{vmatrix} = a_{00} \times \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} - a_{01} \times \begin{vmatrix} a_{10} & a_{12} \\ a_{20} & a_{22} \end{vmatrix} + a_{02} \times \begin{vmatrix} a_{10} & a_{11} \\ a_{20} & a_{21} \end{vmatrix}$$

écrire une fonction **Det3** qui utilise la fonction précédente pour calculer le déterminant d'ordre trois. Les éléments de ce déterminant seront placés dans un tableau d'ordre trois et la forme de cette fonction doit être:

*fonction Det3(T[3][3]:réel):réel*

- c. Ecrire une procédure **lecture** qui permet de remplir un tableau T[3][4] de trois lignes et quatre colonnes représentant les coefficients du système:

$$\begin{cases} a_{00}x + a_{01}y + a_{02}z = a_{03} \\ a_{10}x + a_{11}y + a_{12}z = a_{13} \\ a_{20}x + a_{21}y + a_{22}z = a_{23} \end{cases} \Rightarrow T = \begin{array}{|c|c|c|c|} \hline a_{00} & a_{01} & a_{02} & a_{03} \\ \hline a_{10} & a_{11} & a_{12} & a_{13} \\ \hline a_{20} & a_{21} & a_{22} & a_{23} \\ \hline \end{array}$$

- d. Ecrire une procédure **Select** qui ayant l'indice d'une colonne du tableau T, recopie de ce tableau les trois premières colonnes dans un autre tableau F[3][3] tout en remplaçant la colonne indiquée en paramètre par la dernière colonne. Si l'indice donné est celui de la dernière colonne, alors la procédure doit recopier les trois premières colonnes sans remplacer aucune d'elles. La forme de cette procédure doit être:

*procédure Select(T[3][4]:réel, c:entier, &F[3][3])*

Par exemple, l'appel de la fonction select(T, 0, F) doit mettre dans F les éléments:

$$F = \begin{array}{|c|c|c|} \hline a_{03} & a_{01} & a_{02} \\ \hline a_{13} & a_{11} & a_{12} \\ \hline a_{23} & a_{21} & a_{22} \\ \hline \end{array}$$

- e. On pose:

$$\Delta = \begin{vmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{vmatrix}, \Delta_x = \begin{vmatrix} a_{03} & a_{01} & a_{02} \\ a_{13} & a_{11} & a_{12} \\ a_{23} & a_{21} & a_{22} \end{vmatrix},$$

$$\Delta_y = \begin{vmatrix} a_{00} & a_{03} & a_{02} \\ a_{10} & a_{13} & a_{12} \\ a_{20} & a_{23} & a_{22} \end{vmatrix} \text{ et } \Delta_z = \begin{vmatrix} a_{00} & a_{01} & a_{03} \\ a_{10} & a_{11} & a_{13} \\ a_{20} & a_{21} & a_{23} \end{vmatrix}$$

Si  $\Delta = 0$  alors le système n'admet pas de racines. Par contre, si  $\Delta \neq 0$  alors le système admet les solutions:

$$x = \frac{\Delta_x}{\Delta}, y = \frac{\Delta_y}{\Delta} \text{ et } z = \frac{\Delta_z}{\Delta}$$

Ecrire alors une fonction **Système** qui donne la solution d'un système de trois équations à trois inconnues et retourne la valeur 1 si la solution est possible. Si la solution n'est pas possible, cette fonction doit retourner -1. Une telle procédure doit avoir la forme:

*fonction Système(T[3][4]:réel, &x:réel, &y:réel, &z:réel):entier*

On peut remarquer que le tableau F représentant les éléments de  $\Delta$  peut être obtenu en faisant  $\text{Select}(T, 3, F)$  et  $\Delta = \text{Det3}(F)$ . De même le tableau F représentant les éléments de  $\Delta_x$  est obtenu en faisant  $\text{Select}(T, 0, F)$ , ...

### Exercice 60

- a. Ecrire une fonction booléenne qui permet de vérifier si un entier, donné en paramètre, est **premier** ou non. Une telle fonction aura la forme suivante:

*fonction premier(N:entier):booléen*

- b. Ecrire une fonction booléenne qui permet de vérifier si un entier, donné en paramètre, est **pair** ou non. Une telle fonction aura la forme suivante:

*fonction pair(N:entier):booléen*

- c. Ecrire une procédure **Copier** qui a comme paramètres deux tableaux A[N] et B[N] de type entier. Cette procédure permet de copier dans le tableau B les éléments premiers ou pairs du tableau A. Une telle procédure aura la forme:

*procédure Copier(A[N]: entier, N: entier, &B[N]: entier)*

### Exercice 61

- a. Ecrire une fonction **Max** qui permet de vérifier si un élément A[k][l] est le plus grand élément de la ligne k d'un tableau réel A[N][M]. Une telle fonction aura la forme suivante:

*fonction Max(A[N][M]:réel, N:entier, M:entier, k:entier, l:entier):booléen*

- b. Ecrire une fonction **Min** qui permet de vérifier si un élément A[k][l] est le plus petit élément de la colonne l d'un tableau réel A[N][M]. Une telle fonction aura la forme suivante:

*fonction Min(A[N][M]:réel, N:entier, M:entier, k:entier, l:entier):booléen*

- c. Ecrire une procédure **PointCol** qui permet d'afficher tous les éléments d'un tableau réel A[N][M] qui sont en même temps les plus grands dans leurs lignes et les plus petits dans leurs colonnes. Une telle procédure aura la forme suivante:

*procédure PointCol(A[N][M]: réel, N:entier, M:entier)*