



REPUBLIQUE TUNISIENNE  
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE  
LA RECHERCHE SCIENTIFIQUES ET TECHNOLOGIQUES



UNIVERSITE DE JENDOUBA  
FACULTE DES SCIENCES JURIDIQUES, ECONOMIQUES ET DE GESTION DE JENDOUBA

# **Fascicule de Travaux Dirigés** **Algorithmique et structures de données I**

**Adressé aux étudiants de 1<sup>ère</sup> année Licence Fondamentale en  
Informatique Appliquée à la Gestion**

**Equipe pédagogique :**

**Riadh IMED FEREH**  
**Maître de conférences en Informatique**

**Riadh BOUSLIMI**  
**Technologue en Informatique**

**Année Universitaire : 2006-2007**

# PRÉFACE

---

Ce fascicule des travaux dirigés d'algorithmique et structures de données est à l'intention des étudiants de la première année en Licence en Informatique Appliquée à la Gestion de la Faculté des Sciences Juridiques, Économique et de Gestion de Jendouba.

Le fascicule comporte 6 TD avec leurs corrections qui sont réparties comme suit :

TD1 : Les actions élémentaires simples

TD2 : Les structures conditionnelles

TD3 : Les structures itératives

TD4 : Les chaînes de caractères

TD5 : Les sous-programmes

TD6 : Les tableaux

L'objectif principal est de faire apprendre aux étudiants à résoudre un problème. Nous avons introduit dans le TD1 toutes les structures de données qui vont être les objets de manipulation de toutes les définitions de contrôle qui suivront.

Quant aux structures de contrôle, nous les avons développées séparément dans deux travaux dirigés (TD2 et TD3). Nous commencerons par des exercices sur les structures simples, puis les structures conditionnelles et enfin les structures itératives.

Ces trois travaux dirigés nous permettront de manipuler les chaînes de caractères qui seront notre TD4. Nous traiterons dans ce TD les programmes généralement posés afin que l'étudiant sera capable de résoudre des problèmes qui leurs ressemblent.

Ces outils nous permettront par la suite d'aborder les fonctions et les procédures. En effet, les sous-programmes sont la base de la programmation pour simplifier l'écriture d'un programme et lui rendre plus lisible.

Nous terminons à la fin par le TD5 qui sera consacré pour les tableaux et les traitements avancés et on verra dans ce dernier les algorithmes de recherche et les algorithmes de tri.

Avant d'assister à la séance de TD, chaque étudiant doit préparer sérieusement le TD se rapportant à la manipulation qu'il va effectuer et ce à l'aide du cours.

Enfin, nous espérons que le présent ouvrage aura le mérite d'être un bon support pédagogique pour l'enseignant et un document permettant une concrétisation expérimentale pour l'étudiant.

**Les auteurs**

**Riadh IMED Fareh**

**Riadh BOUSLIMI**

## **Table des matières**

<b>TD n° 1(Les actions élémentaires simples).....</b>	<b>4</b>
<b>Correction du TD 1 .....</b>	<b>6</b>
<b>TD n° 2(Les structures conditionnelles) .....</b>	<b>8</b>
<b>Correction du TD 2 .....</b>	<b>9</b>
<b>TD n° 3(Les structures itératives).....</b>	<b>11</b>
<b>Correction du TD 3 .....</b>	<b>13</b>
<b>TD n° 4(Les chaînes de caractères).....</b>	<b>18</b>
<b>Correction du TD 4 .....</b>	<b>19</b>
<b>TD n° 5(Procédures et fonctions).....</b>	<b>22</b>
<b>Correction du TD 5 .....</b>	<b>23</b>
<b>TD n° 6(Les Tableaux).....</b>	<b>26</b>
<b>Correction du TD 6 .....</b>	<b>27</b>
<b>Bibliographie.....</b>	<b>35</b>



Faculté des Sciences Juridiques,  
Economiques et de Gestion de Jendouba

Année Universitaire : 2006/2007 – Semestre 1

Module : Algorithmique et structures de données I

Classe : 1<sup>ère</sup> LFIAG

Enseignants: Riadh IMED FERREH & Riadh BOUSLIMI

# TD n° 1

(Les actions élémentaires simples)

## Objectifs

- ✓ Connaître le vocabulaire de base de programmation (constante, variable, expression,...)
- ✓ Comprendre la démarche de programmation
- ✓ Comprendre les actions algorithmiques simples
- ✓ Connaître la structure générale d'un algorithme

## Exercice 1

1. Quel est l'ordre de priorité des différents opérateurs de l'expression suivante :

$$((3 * a) - x^2) - (((c - d) / (a / b)) / d)$$

2. Evaluer l'expression suivante :

$$5 + 2 * 6 - 4 + (8 + 2^3) / (2 - 4 + 5 * 2)$$

3. Ecrire la formule suivante sous forme d'une expression arithmétique :

$$\frac{(3 - xy)^2 - 4ac}{2x - z}$$

## Exercice 2

Sachant que  $a = 4$ ,  $b = 5$ ,  $c = -1$  et  $d = 0$ , évaluer les expressions logiques suivantes :

1.  $(a < b)$  ET  $(c \geq d)$
2. NON  $(a < b)$  OU  $(c \neq b)$
3. NON  $(a \neq b^2)$  OU  $(a * c < d)$

## Exercice 3

Donner toutes les raisons pour lesquelles l'algorithme suivant est incorrect :

**Algorithme Incorrect**

$x, y$  : Entier

$z$  : Réel

**Début**

$z \leftarrow x + 2$

$y \leftarrow z$

$x * 2 \leftarrow 3 + z$

$y \leftarrow 5y + 3$

**Fin.**

#### **Exercice 4**

Ecrire un algorithme qui lit deux entiers au clavier et qui affiche ensuite leur somme et leur produit.

#### **Exercice 5**

Ecrire un algorithme qui calcule et affiche la résistance d'un composant électronique en utilisant la loi d'Ohm :

$$U = R \times I \quad \text{avec} \quad \left\{ \begin{array}{l} U : \text{Tension en V} \\ R : \text{Résistance en } \Omega \\ I : \text{Intensité en A} \end{array} \right.$$

## Correction du TD 1

### Exercice 1

$$1. \quad \underset{1}{((3} \quad \underset{3}{*} \quad \underset{2}{a)} \quad \underset{8}{-} \quad \underset{4}{x} \quad \underset{6}{^2)} \quad \underset{5}{-} \quad \underset{7}{(((c-d)} \quad \underset{6}{/} \quad \underset{5}{(a/b))} \quad \underset{7}{/} \quad \underset{7}{d)}$$

$$2. \quad 5 + 2 * 6 - 4 + (8 + 2^3) / (2 - 4 + 5 * 2) = 15$$

$$3. \quad ((3 - x * y)^2 - 4 * a * c) / (2 * x - z)$$

### Exercice 2

1. Faux
2. Vrai
3. Faux

**NB :** le résultat d'une expression logique est toujours Vrai ou Faux.

### Exercice 3

1	<b>Algorithme</b> Incorrect
2	x,y : Entier
3	z : Réel
4	<b>Début</b>
5	z ← x + 2
6	y ← z
7	x * 2 ← 3 + z
8	y ← 5y + 3
9	<b>Fin.</b>
10	

Cet algorithme est incorrect pour plusieurs raisons:

- Ligne 1 : le mot Algorithme s'écrit avec un "h" au milieu.
- Ligne 2 : la déclaration des variables commence par le mot "Var".
- Ligne 5 : la valeur de x est indéterminée.
- Ligne 6 : incompatibilité de type (un réel affecté à une variable de type entier).
- Ligne 7 : le membre gauche d'une affectation doit être une variable.
- Ligne 8 : il faut écrire 5 \*y et non 5y.

**Exercice 4** : calcul de la somme et du produit de deux entiers

**Algorithme** Som\_Prod

**Var**

a , b , s , p : Entier

**Début**

Ecrire("Entrer la valeur de a="), Lire(a)

Ecrire("Entrer la valeur de b="), Lire(b)

$s \leftarrow a + b$

$p \leftarrow a * b$

Ecrire("Somme=",s)

Ecrire("Produit=",p)

**Fin.**

**Exercice 5** : calcul de la résistance d'un composant électrique

**Algorithme** Résistance

**Var**

U, I , R : Réel

**Début**

Ecrire("Entrer la tension="), Lire(U)

Ecrire("Entrer l'intensité="), Lire(I)

$R \leftarrow U / I$  (\* on suppose toujours  $I \neq 0$  \*)

Ecrire("Résistance =",R," Ohms")

**Fin.**



Faculté des Sciences Juridiques,  
Economiques et de Gestion de Jendouba

Année Universitaire : 2006/2007 – Semestre 1

Module : Algorithmique et structures de données I

Classe : 1<sup>ère</sup> LFIAG

Enseignants: Riadh IMED FERREH & Riadh BOUSLIMI

## TD n° 2

(Les structures conditionnelles)

### Objectif

- ✓ Construire des algorithmes comportant des traitements conditionnels.

### Exercice 1

Ecrire un algorithme qui calcule et affiche la valeur absolue d'un entier quelconque lu au clavier.

### Exercice 2

Ecrire un algorithme qui calcule et affiche si la valeur d'un entier quelconque lu au clavier est paire ou impaire.

### Exercice 3

Ecrire un algorithme permettant de résoudre dans R une équation du second degré de la forme  $ax^2+bx+c=0$ .

### Exercice 4

Ecrire un algorithme permettant de simuler une calculatrice à 4 opérations(+, -, \*, et /). Utiliser la structures "selon" pour le choix de l'opération à affecter.

### Exercice 5

Ecrire un algorithme qui lit un caractère au clavier puis affiche s'il s'agit d'une lettre minuscule, d'une lettre majuscule, d'un chiffre ou d'un caractère spécial.

### Exercice 6

Une année bissextile (contient 366 jours) si elle est multiple de 4, sauf les années de début de siècle (qui se terminent par 00) qui ne sont bissextiles que si elles sont divisibles par 400.

Exemples

- 1980 et 1996 sont bissextiles car elles sont divisibles par 4
- 2000 est une année bissextile car elle est divisible par 400
- 2100 et 3000 ne sont pas bissextiles car elles ne sont pas divisibles par 400.

Ecrire un algorithme qui permet de déterminer si un entier positif donné correspond à une année bissextile ou non.



## Correction du TD 2

### Exercice 1

**Algorithme** Val\_Abs

**Var**

x, va : Entier

**Début**

Ecrire("Entrer un entier="), Lire(x)

**Si** ( x >= 0 ) **Alors**

va  $\leftarrow$  x

**Sinon**

va  $\leftarrow$  -x

**FinSi**

Ecrire("|",x,"|=",va)

**Fin.**

### Exercice 2

**Algorithme** pair\_impair

**Var**

x : Entier

**Début**

Ecrire("Entrer un entier="), Lire(x)

**Si** ( x Mod 2 = 0 ) **Alors**

Ecrire("c'est un entier pair")

**Sinon**

Ecrire("c'est un entier impair")

**FinSi**

**Fin.**

### Exercice 3

**Algorithme** equa2d

**Var**

a,b,c,delta : Réel

**Début**

Ecrire("Entrer la valeur de a(non nulle)="), Lire(a)

Ecrire("Entrer la valeur de b="), Lire(b)

Ecrire("Entrer la valeur de c="), Lire(c)

delta  $\leftarrow$   $b^2 - 4*a*c$

**Si** ( delta < 0 ) **Alors**

Ecrire("pas de solution dans R")

**Sinon Si** ( delta = 0 ) **Alors**

Ecrire("x1 = x2 = ", -b/(2\*a))

**Sinon**

Ecrire("x1 = ", (-b-racine(delta))/(2\*a))

Ecrire("x2 = ", (-b+racine(delta))/(2\*a))

**FinSi**

**Fin.**

**Remarque :** Dans cet algorithme, on suppose que l'utilisateur va toujours entrer une valeur de a non nulle. Sinon ce n'est pas une équation du second degré.

#### Exercice 4

**Algorithme** calculatrice

**Var**

val1, val2 : Réel  
opération: caractère

**Début**

Ecrire("Première opérande="), Lire(val1)  
Ecrire("Opération="), Lire(opération)  
Ecrire("Deuxième opérande="), Lire(val2)

**Selon** opération **Faire**

"+" : Ecrire("Résultat =", val1 + val2)  
"- " : Ecrire("Résultat =", val1 - val2)  
"\*" : Ecrire("Résultat =", val1 \* val2)  
"/" : **Si** ( val2 # 0) **Alors**  
    Ecrire("Résultat =", val1 / val2)  
    **Sinon**  
        Ecrire("Division par zéro!")  
    **FinSi**

**Sinon**

Ecrire("opérateur erroné...")

**FinSi**

**Fin.**

#### Exercice 5

**Algorithme** Nature\_Caractère

**Var**

c: caractère

**Début**

Ecrire("Entrer un caractère="), Lire(c)

**Selon** c **Faire**

"a".. "z" : Ecrire("C'est une lettre miniscule")  
"A".. "Z" : Ecrire("C'est une lettre majuscule")  
"0".. "9" : Ecrire("C'est un chiffre")

**Sinon**

Ecrire("c'est un caractère spécial")

**FinSi**

**Fin.**

#### Exercice 6

**Algorithme** Bissextile

**Var**

n : Entier

**Début**

Ecrire("Entrer l'année="), Lire(n)

**Si** ( n Mod 400 = 0 ) OU (( n Mod 100 # 0) ET (n Mod 4) = 0))<0 ) **Alors**  
    Ecrire("Année bissextile")

**Sinon**

Ecrire("Année non bissextile")

**FinSi**

**Fin.**



Faculté des Sciences Juridiques,  
Economiques et de Gestion de Jendouba

Année Universitaire : 2006/2007 – Semestre 1

Module : Algorithmique et structures de données I

Classe : 1<sup>ère</sup> LFIAG

Enseignants: Riadh IMED FERREH & Riadh BOUSLIMI

## TD n° 3

(Les structures itératives)

### Objectif

- ✓ Construire des algorithmes comportant des traitements itératifs.

### Exercice 1

Ecrire un algorithme qui lit un entier positif n puis affiche tous ses diviseurs.

### Exercice 2

Ecrire un algorithme avec trois versions qui lit un entier positif n puis calcule et affiche son factoriel selon la formule  $n! = 1 \times 2 \times \dots \times n$ .

- Pour...Faire
- Tant que ... Faire
- Répéter ... Jusqu'à...

### Exercice 3

Ecrire un algorithme permettant de :

- Lire un nombre fini de notes comprises entre 0 et 20.
- Afficher la meilleure note, la mauvaise note et la moyenne de toutes les notes.

### Exercice 4

Calculer  $a^b$  avec a réel et b entier par multiplication successives.

### Exercice 5

Ecrire un algorithme qui lit un entier positif et vérifie si ce nombre est premier ou non.

**Remarque :** un nombre premier n'est divisible que par 1 ou par lui-même.

### Exercice 6

Ecrire un algorithme qui lit deux entiers positifs A et B puis calcule et affiche leur PGCD en utilisant la méthode suivante:

- Si  $A = B$ ;  $\text{PGCD}(A,B) = A$
- Si  $A > B$ ;  $\text{PGCD}(A,B) = \text{PGCD}(A-B,B)$
- Si  $A < B$ ;  $\text{PGCD}(A,B) = \text{PGCD}(A,B-A)$

**Exemple:**  $\text{PGCD}(18,45) = \text{PGCD}(18,27) = (\text{PGCD}(18,9) = \text{PGCD}(9,9) = 9$

### **Exercice 7**

Ecrire un algorithme qui calcule le PPCM (Plus Petit Commun Multiple) de 2 entiers positifs A et B en utilisant la méthode suivante :

- Permuter, si nécessaire, les données de façon à ranger dans A le plus grand des 2 entiers;
- Chercher le plus petit multiple de A qui est aussi multiple de B.

**Exemple:**  $\text{PPCM}(6,8) = \text{PPCM}(8,6) = 24$ .

### **Exercice 8**

Ecrire un algorithme qui calcule et affiche les 10 premiers termes de la suite de Fibonacci.

La suite de Fibonacci est définie par :

- $F_0 = 1$
- $F_1 = 1$
- $F_n = F_{n-2} + F_{n-1}$  pour  $n > 1$ .

### **Exercice 9**

Ecrire un algorithme qui calcule la somme harmonique  $s = \sum_{i=1}^n \frac{1}{i}$  ; n est un entier positif lu à partir du clavier

**Exemple:** Pour  $n = 3$ ,  $s = 1 + 1/2 + 1/3 = 1.83$

### **Exercice 10**

Parmi tous les entiers supérieurs à 1, seuls 4 peuvent être représentés par la somme des cubes de leurs chiffres.

A titre d'exemple,  $153 = 1^3 + 5^3 + 3^3$  est un nombre cubique.

Ecrire un algorithme permettant de déterminer les 3 autres.

**Note :** les 4 nombres sont compris entre 150 et 410.

### **Exercice 11**

Un nombre parfait est un nombre présentant la particularité d'être égal à la somme de tous ses diviseurs, excepté lui-même.

Le premier nombre parfait est  $6 = 3 + 2 + 1$ .

Ecrire un algorithme qui affiche tous les nombres parfaits inférieurs à 1000.

## Correction du TD 3

### Exercice 1

**Algorithme** Diviseurs

**Var**

n,i : Entier

**Début**

Ecrire("Entrer un entier positif="), Lire(n)

**Pour** i **de** 1 **à** n **Faire**

**Si** ( n Mod i = 0 ) **Alors** (*\* Si le reste de la valeur de n est égale à 0 \**)

Ecrire(i)

**Fin Si**

**Fin Pour**

**Fin.**

### Exercice 2

#### ▪ Version Pour... Faire

**Algorithme** Facto

**Var**

n,i,f : Entier

**Début**

Ecrire("Entrer un entier positif="), Lire(n)

f  $\leftarrow$  1 (*\* initialisation de la factorielle à 1 puisque 1!=1 \**)

**Pour** i **de** 2 **à** n **Faire**

f  $\leftarrow$  f \* i (*\* Pour chaque parcours on multiplie l'ancienne valeur de f par i \**)

**Fin Pour**

Ecrire(n,"!=" ,f)

**Fin.**

#### ▪ Version Tant que... Faire

**Algorithme** Facto

**Var**

n,i,f : Entier

**Début**

Ecrire("Entrer un entier positif="), Lire(n)

f  $\leftarrow$  1 (*\* initialisation de la factorielle à 1 puisque 1!=1 \**)

i  $\leftarrow$  2 (*\* initialisation du compteur i \**)

**Tant que**(i  $\leq$  n) **Faire**

f  $\leftarrow$  f \* i (*\* Pour chaque parcours on multiplie l'ancienne valeur de f par i \**)

i  $\leftarrow$  i + 1 (*\* incrémentation du compteur i \**)

**Fin Pour**

Ecrire(n,"!=" ,f)

**Fin.**

▪ **Version Répéter... Jusqu'à**

**Algorithme** Facto

**Var**

n,i,f : Entier

**Début**

Ecrire("Entrer un entier positif="), Lire(n)

$f \leftarrow 1$  (\* initialisation de la factorielle à 1 puisque  $1!=1$  \*)

$i \leftarrow 2$  (\* initialisation du compteur i \*)

**Répéter**

$f \leftarrow f * i$  (\* Pour chaque parcours on multiplie l'ancienne valeur de f par i \*)

$i \leftarrow i + 1$  (\* incrémentation du compteur i \*)

**Jusqu'à** (i=n)

Ecrire(n,"!=" ,f)

**Fin.**

**Exercice 3**

**Algorithme** Notes

**Var**

n, i: Entier

note, min, max, s : Réel

**Début**

Ecrire("Entrer le nombre de notes=")

Lire(a) (\* **On suppose que n est toujours supérieur à zéro** \*)

$s \leftarrow 0$

$\min \leftarrow 0$  (\* initialisation de la note minimale à 0 \*)

$\max \leftarrow 20$

**Pour i de 1 à n Faire**

Ecrire("Entrer une note="), Lire(note)

$s \leftarrow s + \text{note}$  (\* additionner la nouvelle note \*)

**Si** ( note < min ) **Alors**

$\min \leftarrow \text{note}$  (\* mémorisation de la nouvelle valeur minimale \*)

**Fin Si**

**Fin Pour**

Ecrire("Meilleur note = ",max)

Ecrire("Mauvaise note = ",max)

Ecrire("Moyenne des notes = ",s/n)

**Fin.**

**Exercice 4**

**Algorithme** Puissance

**Var**

a,c : Réel

b,i: Entier

**Début**

Ecrire("Entrer la valeur de a="), Lire(a)

Ecrire("Entrer la valeur de b="), Lire(b)

```

c ← 1 (* initialisation du résultat du produit *)
Pour i de 1 à Abs(b) Faire
    c ← c * a      (*produit de axa b fois *)
Fin Pour
Si ( b < 0 ) Alors (* si b est négative alors le résultat sera 1/c *)
    c ← 1 / c
Fin Si
Ecrire(a," à la puissance ",b,"=",c)
Fin.

```

### Exercice 5

**Algorithme Premier**

**Var**

n,i,nb\_div: Entier

**Début**

```

Ecrire("Entrer un entier positif="), Lire(n)
nb_div ← 0 (* initialisation du nombre de diviseurs*)
i ← 1
Tant que (i <= n) Faire
    Si ( n Mod i = 0 ) Alors
        nb_div ← nb_div + 1 (* incrémentation du nombre de diviseurs *)
    FinSi
    i ← i +1
Fin Tant que
Si (nb_div <= 2) Alors
    Ecrire("C'est un nombre premier")
Sinon
    Ecrire("Ce n'est pas un nombre premier")
Fin Si

```

**Fin.**

### Exercice 6

**Algorithme PGCD**

**Var**

a,b: Entier

**Début**

```

Ecrire("Entrer la valeur de a ="), Lire(a)
Ecrire("Entrer la valeur de b ="), Lire(b)
Répéter
    Si ( a > b ) Alors
        a ← a - b
    FinSi
    Si ( a < b ) Alors
        b ← b - a
    FinSi
Jusqu'à ( a =b )
Ecrire("Le PGCD =",a)

```

**Fin.**

### Exercice 7

#### Algorithme PPCM

##### Var

a,b,i,x: Entier

##### Début

Ecrire("Entrer la valeur de a ="), Lire(a)

Ecrire("Entrer la valeur de b ="), Lire(b)

**Si** ( a < b ) **Alors**

    x ← a   (\*-----\*)

    a ← b   (\* Permutation \*)

    b ← x   (\*-----\*)

**FinSi**

i ← 1

**Tant que** (((i\*a) Mod b) # 0) **Faire**

    i ← i + 1

**Fin Tant que**

Ecrire("PPCM =",i\*a)

**Fin.**

### Exercice 8

#### Algorithme Fibo

##### Var

f0,f1,f,i: Entier

##### Début

f0 ← 1

Ecrire("f0 = ",f0)

f1 ← 1

Ecrire("f1 = ",f1)

**Pour** i de 2 à 9 **Faire**

    f ← f0 + f1

    Ecrire("f",i," = ",f)

    f0 ← f1

    f1 ← f

**Fin Pour**

**Fin.**

### Exercice 9

#### Algorithme Somme

##### Var

n,i: Entier

s : Réel

##### Début

Ecrire("Entrer la valeur de n = "), Lire(n)

s ← 0 (\* initialisation de la somme \*)

**Pour** i de 1 à n **Faire**

    s ← s + 1/i (\* incrémenter la somme avec la nouvelle valeur de 1/i \*)

**Fin Pour**

Ecrire("somme=",s)

**Fin.**



### Exercice 10

Algorithme cubique

Var

i, centaine, dizaine, unite: Entier

Début

Pour i de 150 à 410 Faire

centaine  $\leftarrow$  i Div 100 (\* ex :  $i=150 \rightarrow$  centaine = 1 \*)

dizaine  $\leftarrow$  ( i Mod 100) Div 10 (\* ex :  $i=150 \rightarrow$  dizaine = 5 \*)

unite  $\leftarrow$  (i Mod 100) Mod 10 (\* ex :  $i=150 \rightarrow$  unite = 0 \*)

Si (( centaine<sup>3</sup> + dizaine<sup>3</sup>+unite<sup>3</sup>) = i) Alors

Ecrire(i, " est un nombre cubique")

Fin Si

Fin Pour

Fin.

Remarque : les nombres cubiques sont : 153, 370, 371 et 407

### Exercice 11

Algorithme parfaits

Var

i, n, s, j: Entier

Début

Pour i de 1 à 1000 Faire

s  $\leftarrow$  0

Pour j de 1 à ( i Div 2 ) Faire

Si (i Mod j = 0) Alors

s  $\leftarrow$  s + j

Fin Si

Pour

Fin Pour

Si ( s= i) Alors

Ecrire(i, " est un nombre parfait")

Fin Si

Fin.

Remarque : les nombres parfait inférieurs à 1000 sont : 6, 28, 496.



Faculté des Sciences Juridiques,  
Economiques et de Gestion de Jendouba

Année Universitaire : 2006/2007 – Semestre 1

Module : Algorithmique et structures de données I

Classe : 1<sup>ère</sup> LFIAG

Enseignants: Riadh IMED FERREH & Riadh BOUSLIMI

## TD n° 4

(Les chaînes de caractères)

### Objectif

✓ Construire des algorithmes qui traitent des caractères et des chaînes de caractères.

### Exercice 1

Ecrire un algorithme qui lit un caractère au clavier puis affiche son prédécesseur, son successeur et le code ASCII de son équivalent en majuscule.

### Exercice 2

Ecrire un algorithme qui lit une lettre au clavier puis affiche s'il s'agit d'une consonne ou d'une voyelle.

**Remarque :** Les voyelles sont : "A", "a", "E", "e", "I", "i", "O", "o", "U", "u", "Y", "y".

### Exercice 3

Ecrire un algorithme "**Palind**" qui lit une chaîne de caractères et vérifie si cette chaîne est un palindrome ou non.

Un palindrome est un mot qui peut être lu indifféremment de droite à gauche ou de gauche à droite (Exemple: "AZIZA", "LAVAL", "RADAR",...).

### Exercice 4

Ecrire un algorithme qui lit une chaîne de caractères puis affiche son inverse.

**Exemple:** Si la chaîne entrée est "algo", l'algorithme doit afficher "ogla"

### Exercice 5

Ecrire un algorithme qui lit une chaîne de caractères et renvoie son équivalent en majuscules.

### Exercice 6

Ecrire un algorithme qui permet de compter le nombre de mots dans une phrase.

La phrase commence obligatoirement par une lettre et les mots sont séparés par des espaces.

### Exercice 7

Ecrire un algorithme qui détermine et affiche le mot le plus long dans une phrase donnée.

### Exercice 8

Ecrire un algorithme qui lit :

- Un mot (chaîne de caractère formée uniquement de lettres)
- Une lettre

Puis affiche le nombre d'apparitions de la lettre dans le mot.

## Correction du TD 4

### Exercice 1

**Algorithme** Caract

**Var**

c : Caractère

**Début**

Ecrire("Entrer un caractère="), Lire(c)

Ecrire(pred(c)) (\* affichage du prédécesseur du caractère c \*)

Ecrire(succ(c)) (\* affichage du successeur du caractère c \*)

Ecrire(asc(majus(c)) (\* affichage du code ASCII du majuscule de caractère c \*))

**Fin.**

### Exercice 2

**Algorithme** Cons\_Voy

**Var**

c : Caractère

**Début**

**Répéter**

Ecrire("Entrer une lettre:"), Lire(c) (\* saisie contrôler d'une lettre \*)

**Jusqu'à** ( c >= "A" **ET** c <= "Z") **OU** ( c >= "a" **ET** c <= "z")

**Si** (Majus(c) = "A") **OU** (Majus(c) = "E") **OU** (Majus(c) = "I")

**OU** (Majus(c) = "O") **OU** (Majus(c) = "U") **OU** (Majus(c) = "Y") **Alors**

Ecrire(c, " est une voyelle")

**Sinon**

Ecrire(c, " est une consonne")

**Fin Si**

**Fin.**

### Exercice 3

**Algorithme** Palind

**Var**

ch: Chaîne

i, L : Entier

Pal : Booléen

**Début**

Ecrire("Entrer une chaîne non vide="), Lire(ch)

L ← long(ch) (\* longueur de la chaîne \*)

Pal ← Vrai (\* on suppose initialement que la chaîne est palindrome \*)

i ← 1 (\* initialisation du compteur i \*)

**Tant que** ( i <= L Div 2 ) **ET** (Pal) **Faire** (\* Parcours de la chaîne jusqu'à la moitié \*)

**Si** ( ch[i] = ch[L-i+1] ) **Alors** (\* s'il sont égaux alors on incrémente \*)

i ← i + 1 (\* incrémentation \*)

**Sinon**

Pal ← Faux (\* s'il y'a deux lettres différentes alors on s'arrête \*)

**Fin Si**

**Fin Tant que**

**Si (Pal) Alors**  
 Ecrire(ch, " est un palindrome")  
**Sinon**  
 Ecrire(ch, " n'est pas un palindrome")  
**Fin Si**

**Fin.**

#### **Exercice 4**

**Algorithme** inverse

**Var**

i, L : Réel  
 ch1, ch2: Chaîne

**Début**

Ecrire("Entrer une chaîne :"), Lire(ch1)  
 L  $\leftarrow$  Long(ch1) (\* Longueur de la chaîne \*)  
 ch2  $\leftarrow$  "" (\* initialisation de la chaîne inverse \*)  
**Pour i de 1 à L Faire**  
 ch2  $\leftarrow$  ch1[i] + ch2 (\* insertion du caractère au début la chaîne \*)  
**Fin Pour**  
 Ecrire("Inverse de la chaîne=", ch2)

**Fin**

#### **Exercice 5**

**Algorithme** Majuscule

**Var**

i, L : Entier  
 ch1, ch2: Chaîne

**Début**

Ecrire("Entrer une chaîne :"), Lire(ch1)  
 L  $\leftarrow$  Long(ch1) (\* Longueur de la chaîne \*)  
 ch2  $\leftarrow$  "" (\* initialisation de la nouvelle chaîne au vide \*)  
**Pour i de 1 à L Faire**  
 ch2  $\leftarrow$  ch2 + Majus(ch1[i]) (\* conversion de chaque lettre en majuscule \*)  
**Fin Pour**  
 Ecrire("Chaîne en majuscule=", ch2)

**Fin**

#### **Exercice 6**

**Algorithme** Comptage\_Mots

**Var**

i, L, nb\_mot : Entier  
 phase: Chaîne

**Début**

Ecrire("Entrer une phrase non vide :"), Lire(phase)  
 L  $\leftarrow$  Long(phase) (\* longueur de la phrase \*)  
 Nb\_mot  $\leftarrow$  1 (\* initialisation du compteur de mot // par défaut on un seul mot \*)  
**Pour i de 1 à L Faire**  
 Si (phase[i] = " ") **Alors**  
 nb\_mot  $\leftarrow$  nb\_mot + 1 (\* si on trouve on espace on incrémente le Nb\_mot \*)  
**Fin Si**  
**Fin Pour**  
 Ecrire("Nombre de mots =", nb\_mot)

**Fin**

### Exercice 7

**Algorithme** Plus\_Long\_Mot

**Var**

i,j,L: Entier  
phrase, mot, motpl: Chaîne

**Début**

Ecrire("Entrer une phrase:"), Lire(phrase)  
L  $\leftarrow$  Long(phrase)  
motpl  $\leftarrow$  "" (*\* initialisation du mot le plus long \**)  
i  $\leftarrow$  1  
**Tant que** ( i  $\leq$  L ) **Faire**  
    mot  $\leftarrow$  "" (*\* initialisation du mot courant \**)  
    j  $\leftarrow$  i  
    **Tant que** ((j  $\leq$  L) ET (phrase[j]  $\neq$  " ")) **Faire**  
        mot  $\leftarrow$  mot + phrase[j] (*\* création du mot \**)  
        j  $\leftarrow$  j + 1  
    **Fin Tant que**  
    **Si** (long(mot) > long(motpl)) **Alors**  
        Motpl  $\leftarrow$  mot (*\* retenir le nouveau mot le plus grand \**)  
    **Fin Si**  
    i  $\leftarrow$  j+1  
**Fin Tant que**  
Ecrire("Le mot le plus grand est =", motpl)

**Fin**

### Exercice 8

**Algorithme** fréquence

**Var**

i,L,nb: Entier  
mot: Chaîne  
lettre : Caractère

**Début**

Ecrire("Entrer un mot:"), Lire(mot)  
Ecrire("Entrer une lettre:"), Lire(lettre)  
L  $\leftarrow$  Long(mot) (*\* longueur du mot \**)  
nb  $\leftarrow$  0 (*\* initialisation du compteur d'occurrence de la lettre cherchée \**)  
**Pour** i **de** 1 **à** L **Faire**  
    **Si** (mot[i] = lettre) **Alors**  
        nb  $\leftarrow$  nb + 1 (*\* incrémentation du nombre d'occurrence \**)  
    **Fin Si**  
**Fin Pour**  
Ecrire(lettre, " apparaît ", nb, " fois dans ", mot)

**Fin**



Faculté des Sciences Juridiques,  
Economiques et de Gestion de Jendouba

Année Universitaire : 2006/2007 – Semestre 1

Module : Algorithmique et structures de données I

Classe : 1<sup>ère</sup> LFIAG

Enseignants: Riadh IMED FERREH & Riadh BOUSLIMI

## TD n° 5

(Procédures et fonctions)

### Objectifs

- ✓ Appliquer la démarche de programmation modulaire pour construire des algorithmes structurés en procédures et fonctions
- ✓ Savoir le mode de passage de paramètre d'une procédure et une fonction
- ✓ Apprendre et distinguer l'appel d'une procédure et une fonction

### Exercice 1

Ecrire une procédure **puissance** qui calcule  $c = a^b = a \times a \times a \times \dots \times a$  (b fois); a et b étant des entiers positifs. Tester cette procédure

### Exercice 2

Ecrire une procédure **permut** qui permet d'échanger les valeurs de 2 entiers a et b. Tester cette procédure

### Exercice 3

Ecrire une fonction **minimum** qui retourne le minimum de 2 entiers a et b. Tester cette fonction

### Exercice 4

1. On appelle **bigramme** une suite de deux lettres. Ecrire une procédure qui calcule le nombre d'occurrences d'un bigramme dans une chaîne de caractères.
2. Peut-on transformer cette procédure en fonction? Si oui écrire cette fonction.

### Exercice 5

Ecrire une fonction **Triangle** qui permet de vérifier si les 3 nombres a,b et c peuvent être les mesures des côtés d'un triangle rectangle.

**Remarque:** D'après le théorème de Pythagore, si a, b et c sont les mesures des côtés d'un rectangle, alors  $a^2 = b^2 + c^2$  ou  $b^2 = a^2 + c^2$  ou  $c^2 = a^2 + b^2$

## Correction du TD 5

### Exercice 1



**Procédure** Puissance(a,b : Entier, **var** c : Entier)

**Var**

i : Entier

**Début**

c  $\leftarrow$  1

Pour i de 1 à b Faire

c  $\leftarrow$  c \* a

Fin Pour

**Fin**

**Algorithme** Test {programme principal}

**Var**

x,y,r : Entier

**Début**

Ecrire("Entrer un entier x="), Lire(x)

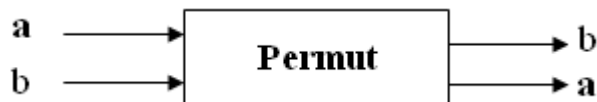
Ecrire("Entrer un entier y="), Lire(y)

**Puissance(x,y,r)** //appel de la procédure

Ecrire(x," à la puissance de ",y, " =",r)

**Fin.**

### Exercice 2



**Procédure** Permut(**var** a,b : Entier)

**Var**

aux : Entier

**Début**

aux  $\leftarrow$  a

A  $\leftarrow$  b

b  $\leftarrow$  aux

**Fin**

**Algorithme** Test {programme principal}

**Var**

x,y : Entier

**Début**

Ecrire("Entrer un entier x="), Lire(x)

Ecrire("Entrer un entier y="), Lire(y)

**Permut(x,y)** //appel de la procédure

Ecrire("La nouvelle valeur de x=",x, " et de y=",y)

**Fin.**

### Exercice 3

**Fonction** Minimum(a,b : Entier) : Entier

**Var**

min : Entier

**Début**

**Si** ( a<= b) **Alors**

min ← a

**Sinon**

min ← b

**Fin Si**

**Fin**

**Algorithme** Test {programme principal}

**Var**

x,y,m : Entier

**Début**

Ecrire("Entrer un entier x="), Lire(x)

Ecrire("Entrer un entier y="), Lire(y)

m ← **Minimum**(x,y) (\* appel de la fonction \*)

Ecrire("Le minimum est=",m)

**Fin.**

### Exercice 4

**Procédure** fréq(bigram : Chaîne[2]; chn:Chaîne; **var** nb: Entier)

**Var**

i, L : Entier

**Début**

L ← Long(chn)

nb ← 0

**Pour** i de 1 à (L-1) **Faire**

**Si** (chn[i]=bigram[1] **ET** (chn[i+1]=bigram[2]) **Alors**

nb ← nb + 1

**Fin Si**

**Fin Pour**

**Fin**

Cette procédure possède un seul paramètre résultat de type entier, donc elle peut être remplacée par une fonction.

**Procédure** fréq(bigram : Chaîne[2]; chn:Chaîne): Entier

**Var**

i, L : Entier

**Début**

L ← Long(chn)

nb ← 0

**Pour** i de 1 à (L-1) **Faire**

**Si** (chn[i]=bigram[1] **ET** (chn[i+1]=bigram[2]) **Alors**

nb ← nb + 1

**Fin Si**

**Fin Pour**

**fréq** ← nb (\*affectation du résultat à la fonction \*)

**Fin**



### **Exercice 5**

**Fonction** triangle(a,b,c : Réel) : Booléen

**Début**

**Si** (  $a^2=b^2+c^2$ ) **OU** ( $b^2=a^2+c^2$ ) **OU** ( $c^2=a^2+b^2$ ) **Alors**  
    triangle  $\leftarrow$  Vrai

**Sinon**  
    triangle  $\leftarrow$  Faux

**Fin Si**

**Fin**



Faculté des Sciences Juridiques,  
Economiques et de Gestion de Jendouba

Année Universitaire : 2006/2007 – Semestre 1

Module : Algorithmique et structures de données I

Classe : 1<sup>ère</sup> LFIAG

Enseignants: Riadh IMED FEREH & Riadh BOUSLIMI

## TD n° 6

(Les Tableaux)

### Objectifs

- ✓ Maîtriser la manipulation des tableaux à une ou à deux dimensions.
- ✓ Traitement des problèmes de recherche dans un tableau (Recherche séquentielle, Recherche dichotomique;
- ✓ Savoir les principaux algorithmes de tri d'un tableau.

### Exercice 1

Ecrire une procédure **Remplir** permettant le remplissage d'un tableau de n entiers.

### Exercice 2

Ecrire une procédure **Afficher** permettant l'afficher les éléments d'un tableau de n entiers.

### Exercice 3

Ecrire une fonction **Minimum** permettant de chercher le minimum dans un tableau T de n entiers.

### Exercice 4

Ecrire une fonction **Recherche\_seq** qui permet de chercher un élément x dans un tableau T. La fonction renvoie Vrai s'il est existant et Faux sinon.

### Exercice 5

Même exercice que n° 3 mais on suppose que le tableau est trié.

### Exercice 6

Donner le principe et l'algorithme de tri de :

- Tri à bulle
- Tri par sélection
- Tri par insertion

### Exercice 7

Ecrire une procédure qui permet de fusionner deux tableaux triés A et B contenant respectivement n et m éléments. Le résultat est un tableau trié C à (n+m) éléments.

**Exemple:**

A	1	20	41
---	---	----	----

B	19	23	27	54	91
---	----	----	----	----	----

C	1	20	41	19	23	27	54	91
---	---	----	----	----	----	----	----	----

### Exercice 8

1. Ecrire une procédure permettant de remplir une matrice;
2. Ecrire une procédure permettant d'afficher une matrice;
3. Ecrire une fonction qui renvoie la somme de deux matrices M1 et M2;
4. Ecrire une fonction qui renvoie le produit de deux matrices M1 et M2;

## Correction du TD 6

### Exemple de déclaration d'un tableau

**Constantes**

n = 100

**Type**

Tab = Tableau[1..n] de Entier

**Var**

T : Tab

### Exemple de déclaration d'une matrice

**Constantes**

n = 100

m = 100

**Type**

Mat = Tableau[1..n,1..m] de Entier

**Var**

Matrice : Mat

### Exercice 1

**Procédure** Remplir (var T : Tab; n: Entier)

**Var**

i : Entier

**Début**

**Pour i de 1 à n Faire**

Ecrire("T[" , i , "]" = "), Lire(T[i])

**Fin Pour**

**Fin**

### Exercice 2

**Procédure** Afficher (T : Tab; n: Entier)

**Var**

i : Entier

**Début**

**Pour i de 1 à n Faire**

Ecrire(T[i], " ")

**Fin Pour**

**Fin**

### Exercice 3

**Fonction** Minimum(T : Tab) : Entier

**Var**

min : Entier

**Début**

min  $\leftarrow$  T[1]

**Pour i de 2 à n Faire**

**Si** ( min > T[i] ) **Alors**

min  $\leftarrow$  a

**Fin Si**

**Fin Pour**

**Fin**

#### Exercice 4

##### Principe de la recherche séquentielle :

Comparer x aux différents éléments du tableau jusqu'à trouver x ou atteindre la fin du tableau.

- *Version avec Répéter... jusqu'à*

**Fonction** Recherche\_seq (T:Tab; n,x:Entier) : Entier

**Var**

i, : Entier

**Début**

i ← 0

**Répéter**

i ← i+1

**Jusqu'à** (T[i]=x) **OU** (i = n)

**Si**(T[i]=x) **Alors**

Recherche\_seq ← Vrai

**Sinon**

Recherche\_seq ← Faux

**Fin Si**

**Fin**

- *Version avec Tant que... Faire*

**Fonction** Recherche\_seq (T:Tab; n,x:Entier) : Entier

**Var**

i, : Entier

**Début**

i ← 1

**Tant que** (T[i] ≠ x) **ET** (i ≤ n) **Faire**

i ← i+1

**Fin Tant que**

**Si**(T[i]=x) **Alors**

Recherche\_seq ← Vrai

**Sinon**

Recherche\_seq ← Faux

**Fin Si**

**Fin**

#### Exercice 5

##### Principe de la recherche dichotomique :

Le but de la recherche dichotomique est de diviser l'intervalle de recherche par 2 à chaque itération. Pour cela, on procède de la façon suivante :

Soient premier et dernier les extrémités gauche et droite de l'intervalle dans lequel on cherche la valeur x, on calcule m, l'indice de l'élément médian :

➔  $m = (\text{premier} + \text{dernier}) \text{ Div } 2$

Il y a trois cas possibles :

- $x < T[m]$  : l'élément  $x$ , s'il existe, se trouve dans l'intervalle  $[premier..m-1]$ .
- $x > T[m]$  : l'élément  $x$ , s'il existe, se trouve dans l'intervalle  $[m+1\dots dernier]$ .
- $x = T[m]$  : l'élément de valeur  $x$  est trouvé, la recherche est terminée.

La recherche dichotomique consiste à itérer ce processus jusqu'à ce que l'on trouve  $x$  ou que l'intervalle de recherche soit vide.

**Fonction** Recherche\_dich( $T$ :Tab;  $n,x$ :Entier) : Entier

**Var**

premier,  $m$ , dernier : Entier  
trouve : Booléen

**Début**

premier  $\leftarrow 1$   
dernier  $\leftarrow n$   
trouve  $\leftarrow$  Faux

**Répéter**

$m \leftarrow (\text{premier} + \text{dernier}) \text{ Div } 2$

(1)

**Si** ( $x < T[m]$ ) **Alors**

dernier  $\leftarrow m - 1$

**Sinon Si** ( $x > T[m]$ ) **Alors**

premier  $\leftarrow m + 1$

**Sinon**

trouve  $\leftarrow$  Vrai

**Fin Si**

**Jusqu'à** (trouve=vrai) **OU** (premier>dernier)

Recherche\_dich  $\leftarrow$  trouve

**Fin**

(2)

## Exercice 6

- **Tri à bulle**

### Principe :

On répète le traitement suivant :

Parcourir les éléments du tableau de 1 à  $(n-1)$ ; si l'élément  $i$  est supérieur à l'élément  $(i+1)$ , alors on permute.

Le programme s'arrête lorsque aucune permutation n'est réalisable.

**Procédure** Tri\_Bulles( **var**  $T$ : Tab;  $n$ : Entier)

**Var**

$i$ ,  $aux$  : Entier  
échange : Booléen

**Début**

**Répéter**

échange  $\leftarrow$  Faux

**Pour**  $i$  **de** 1 **à**  $(n-1)$  **Faire**

**Si**( $T[i] > T[i+1]$ ) **Alors**

$aux \leftarrow T[i]$

$T[i] \leftarrow T[i+1]$

$T[i+1] \leftarrow aux$

échange  $\leftarrow$  Vrai

**Fin Si**

**Fin Pour**

**Jusqu'à** (échange = Faux)

**Fin**

### Optimisation de la procédure du Tri à bulle

**Procédure** Tri\_Bulles( **var**  $T$ : Tab;  $n$ : Entier)

**Var**

$i$ ,  $aux$  : Entier  
échange : Booléen

**Début**

**Répéter**

échange  $\leftarrow$  Faux

**Pour** i de 1 à (n-1) **Faire**

**Si**(T[i] > T[i+1]) **Alors**

aux  $\leftarrow$  T[i]

T[i]  $\leftarrow$  T[i+1]

T[i+1]  $\leftarrow$  aux

échange  $\leftarrow$  Vrai

**Fin Si**

**Fin Pour**

**n**  $\leftarrow$  n-1

**Jusqu'à** (échange = Faux) **OU** (n = 1)

**Fin**

**Trace d'exécution**

*Tableau initial*

*Après la 1<sup>ère</sup> itération*

*Après la 2<sup>ème</sup> itération*

*Après la 3<sup>ème</sup> itération*

*Après la 4<sup>ème</sup> itération*

6	4	2	1	4
4	2	1	4	6
2	1	4	4	6
1	2	4	4	6
1	2	4	4	6

▪ **Tri par sélection**

**Principe :**

Cette méthode consiste :

- chercher l'indice du plus petit élément du tableau T[1..n] et permuter l'élément correspondant avec l'élément d'indice 1;
- chercher l'indice du plus petit élément du tableau T[2..n] et permuter l'élément correspondant avec l'élément d'indice 2;
- ...
- chercher l'indice du plus petit élément du tableau T[n-1..n] et permuter l'élément correspondant avec l'élément d'indice (n-1);

**Procédure** Tri\_Selection( var T: Tab; n: Entier)

**Var**

i,j,aux,indmin: Entier

**Début**

**Pour** i de 1 à (n-1) **Faire**

indmin  $\leftarrow$  i

**Pour** j de (i+1) à n **Faire**

**Si**(T[j] < T[indmin]) **Alors**

indmin  $\leftarrow$  j

**Fin Si**

**Fin Pour**

aux  $\leftarrow$  T[i]

T[i]  $\leftarrow$  T[indmin]

T[indmin]  $\leftarrow$  aux

**Fin Pour**

**Fin**

### Trace d'exécution

Tableau initial	6	4	2	1	4
Après la 1 <sup>ère</sup> itération	1	4	2	6	4
Après la 2 <sup>ème</sup> itération	1	2	4	6	4
Après la 3 <sup>ème</sup> itération	1	2	4	6	4
Après la 4 <sup>ème</sup> itération	1	2	4	4	6

### ▪ Tri par insertion

#### Principe :

Cette méthode consiste à prendre les éléments de la liste un par un et insérer chacun dans sa bonne place de façon que les éléments traités forment une sous-liste triée.

Pour ce faire, on procède de la façon suivante :

- comparer et permuter si nécessaire T[1] et T[2] de façon à placer le plus petit dans la case d'indice 1
- comparer et permuter si nécessaire l'élément T[3] avec ceux qui le précèdent dans l'ordre (T[2] puis T[1]) afin de former une sous-liste triée T[1..3]
- ...
- comparer et permuter si nécessaire l'élément T[n] avec ceux qui le précèdent dans l'ordre (T[n-1] , T[n-2],...) afin d'obtenir un tableau trié.

**Procédure** Tri\_Insertion( **var** T: Tab; n: Entier)

**Var**

i,j,x,pos: Entier

**Début**

**Pour** i **de** 2 **à** n **Faire**

pos  $\leftarrow$  i - 1

**Tant que** (pos  $\geq$  1) **ET** (T[pos] > T[i]) **Faire**

pos  $\leftarrow$  pos - 1

**Fin Tant que**

pos  $\leftarrow$  pos + 1

x  $\leftarrow$  T[i]

**Pour** j **de** (i-1) **à** pos [pas = -1] **Faire**

T[j+1]  $\leftarrow$  T[j]

**Fin Pour**

T[pos]  $\leftarrow$  x

**Fin Pour**

**Fin**

### Trace d'exécution

Tableau initial	6	4	2	1	4
Après la 1 <sup>ère</sup> itération	4	6	2	6	4
Après la 2 <sup>ème</sup> itération	2	4	6	6	4
Après la 3 <sup>ème</sup> itération	1	2	4	6	4
Après la 4 <sup>ème</sup> itération	1	2	4	4	6

### Exercice 7

**Procédure** Fusion (A : Tab; B:Tab; **var** C:Tab; n, m:Entier; **var** k :Entier)

**Var**

i,j : Entier

**Début**

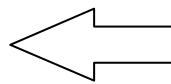
**Tant que** (i <=n) **ET** (j <= m) **Faire**

**Si**(A[i] <=B[j]) **Alors**

C[k] ← A[i]

i ← i + 1

k ← k+1



On a l'élément de A et plus **petit** que B alors on ajoute A dans C

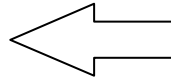
**Fin Si**

**Si** (B[j] <= A[i]) **Alors**

C[k] ← B[j]

j ← j + 1

k ← k+1



On a l'élément de A et plus **grand** que B alors on ajoute B dans C

**Fin Si**

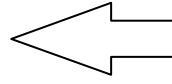
**Fin Tant que**

**Tant que** (i <=n) **Faire**

C[k] ← A[i]

i ← i + 1

k ← k+1



Si on a encore des éléments dans A alors on les rajoute successivement dans C

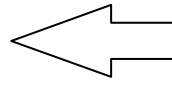
**Fin Tant que**

**Tant que** (j <=m) **Faire**

C[k] ← B[j]

j ← j + 1

k ← k+1



Si on a encore des éléments dans B alors on les rajoute successivement dans C

**Fin Tant que**

**Fin**



## Exercice 8

### 1. Remplissage d'une matrice

**Procédure** Remplir (**var** matrice : Mat; n,m: Entier)  
**Var**  
    i,j : Entier  
**Début**  
    **Pour** i **de** 1 **à** n **Faire**  
        **Pour** j **de** 1 **à** m **Faire**  
            Ecrire("Entrer un entier : "),Lire(T[i,j])  
        **Fin Pour**  
    **Fin Pour**  
**Fin**

### 2. Affichage d'une matrice

**Procédure** Afficher (matrice : Mat; n,m: Entier)  
**Var**  
    i,j : Entier  
**Début**  
    **Pour** i **de** 1 **à** n **Faire**  
        **Pour** j **de** 1 **à** m **Faire**  
            Ecrire(T[i,j])  
        **Fin Pour**  
    **Fin Pour**  
**Fin**

### 3. Somme de deux matrices

#### Principe:

Soient M1 et M2 deux matrices à n ligne et m colonnes.  
 $M3 = M1 + M2$

#### Exemple

M1= 

1	2	3
4	5	6

 M2= 

2	5	3
3	0	1

 M3= 

3	7	6
7	5	7

**Procédure** SomMat (M1,M2 : Mat ; **var** M3 : Mat; n,m: Entier)

**Var**  
    i,j : Entier  
**Début**  
    **Pour** i **de** 1 **à** n **Faire**  
        **Pour** j **de** 1 **à** m **Faire**  
             $M3[i,j] \leftarrow M1[i,j] + M2[i,j]$   
        **Fin Pour**  
    **Fin Pour**  
**Fin**

#### 4. Produit de deux matrices

##### Principe:

Soient M1 et M2 deux matrices à n ligne et m colonnes.

$$M3 = M1 * M2$$

$$M3_{i,j} = M1_{i,1} * M2_{1,j} + M1_{i,2} * M2_{2,j} + \dots + M1_{i,n} * M2_{m,j}$$

##### Exemple

$$\begin{matrix} M1= & \begin{bmatrix} 1 & 2 & 3 \\ 4 & 0 & 5 \end{bmatrix} & & \begin{bmatrix} 2 & 1 \\ 3 & 0 \\ 1 & 4 \end{bmatrix} & & \begin{bmatrix} 11 & 13 \\ 13 & 24 \end{bmatrix} & M2= \\ M3= & & & & & & \end{matrix}$$

**Procédure** ProdMat (M1,M2 : Mat ; **var** M3 : Mat; n,m: Entier; var k : Entier)

**Var**

    i,j : Entier

**Début**

**Pour** i **de** 1 **à** n **Faire**

**Pour** j **de** 1 **à** m **Faire**

            M3[i,j]  $\leftarrow$  0

**Pour** k **de** 1 **à** m **Faire**

                M3[i,j]  $\leftarrow$  M3[i,j] + M1[i,k] \* M2[k,j]

**Fin Pour**

**Fin Pour**

**Fin Pour**

**Fin**

## **B**ibliographie

- [1] Sébastien Rohaut, « *Algorithmique et Techniques fondamentale de programmation* », Edition Eni, 2007.
- [2] Patrice Lignelet et Jean Girerd, « *Algorithmique. Méthodes et modèles* », Paris : Masson, 1985.
- [3] ZITOUNI Baghdadi, « *Algorithmique et structures de données* », Centre de Publication Universitaire, 2003.