

# PARTIE 1

## CORRIGES DES EXERCICES

### Exercice 1.1

Après La valeur des variables est :

A ← 1	A = 1	B = ?
B ← A + 3	A = 1	B = 4
A ← 3	A = 3	B = 4

### Exercice 1.2

Après La valeur des variables est :

A ← 5	A = 5	B = ?	
C = ?			
B ← 3	A = 5	B = 3	
C = ?			
C ← A + B	A = 5	B = 3	
C = 8			
A ← 2	A = 2	B = 3	
C = 8			
C ← B - A	A = 2	B = 3	C = 1

### Exercice 1.3

Après La valeur des variables est :

A ← 5	A = 5	B = ?
B ← A + 4	A = 5	B = 9
A ← A + 1	A = 6	B = 9
B ← A - 4	A = 6	B = 2

### Exercice 1.4

Après La valeur des variables est :

A ← 3	A = 3	B = ?	
C = ?			
B ← 10	A = 3	B = 10	
C = ?			
C ← A + B	A = 3	B = 10	
C = 13			
B ← A + B	A = 3	B = 13	
C = 13			
A ← C	A = 13	B = 13	C = 13

### Exercice 1.5

Après La valeur des variables est :

A ← 5	A = 5	B = ?
B ← 2	A = 5	B = 2
A ← B	A = 2	B = 2
B ← A	A = 2	B = 2

Les deux dernières instructions ne permettent donc pas d'échanger les deux valeurs de B et A, puisque l'une des deux valeurs (celle de A) est ici écrasée. Si l'on inverse les deux dernières instructions, cela ne changera rien du tout, hormis le fait que cette fois c'est la valeur de B qui sera écrasée.

### Exercice 1.6

#### Début

...  
C ← A  
A ← B  
B ← C

#### Fin

On est obligé de passer par une variable dite temporaire (la variable C).

### Exercice 1.7

#### Début

...  
D ← C  
C ← B  
B ← A  
A ← D

#### Fin

En fait, quel que soit le nombre de variables, une seule variable temporaire suffit...

### Exercice 1.8

Il ne peut produire qu'une erreur d'exécution, puisqu'on ne peut pas additionner des caractères.

### Exercice 1.9

...En revanche, on peut les concaténer. A la fin de l'algorithme, C vaudra donc "42312".

## PARTIE 2

### Exercice 2.1

On verra apparaître à l'écran 231, puis 462 (qui vaut  $231 * 2$ )

### Exercice 2.2

Variables nb, carr en Entier

#### Début

Ecrire "Entrez un nombre :"

#### Lire nb

carr ← nb \* nb

Ecrire "Son carré est : ", carr

#### Fin

En fait, on pourrait tout aussi bien économiser la variable carr en remplaçant les deux avant-dernières lignes par :

Ecrire "Son carré est : ", nb\*nb

C'est une question de style ; dans un cas, on privilégie la lisibilité de l'algorithme, dans l'autre, on privilégie l'économie d'une variable.

### Exercice 2.3

Variables nb, pht, ttva, pttc en Numérique

#### Début

Ecrire "Entrez le prix hors taxes :"

#### Lire pht

Ecrire "Entrez le nombre d'articles :"

#### Lire nb

Ecrire "Entrez le taux de TVA :"

#### Lire ttva

pttc ← nb \* pht \* (1 + ttva)

Ecrire "Le prix toutes taxes est : ", pttc

#### Fin

Là aussi, on pourrait squeezer une variable et une ligne en écrivant directement. :

Ecrire "Le prix toutes taxes est : ", nb \* pht \* (1 + ttva)

C'est plus rapide, plus léger en mémoire, mais un peu plus difficile à relire (et à écrire !)

### Exercice 2.4

Variables t1, t2, t3, t4 en Caractère

#### Début

t1 ← "belle Marquise"

```

t2 ← "vos beaux yeux"
t3 ← "me font mourir"
t4 ← "d'amour"
Ecrire t1 & " " & t2 & " " & t3 & " " & t4
Ecrire t3 & " " & t2 & " " & t4 & " " & t1
Ecrire t2 & " " & t3 & " " & t1 & " " & t4
Ecrire t4 & " " & t1 & " " & t2 & " " & t3
Fin

```

## PARTIE 3

### Exercice 3.1

Variable n en Entier

Début

Ecrire "Entrez un nombre : "

Lire n

Si n > 0 Alors

Ecrire "Ce nombre est positif"

Sinon

Ecrire "Ce nombre est négatif"

Finsi

Fin

### Exercice 3.2

Variables m, n en Entier

Début

Ecrire "Entrez deux nombres : "

Lire m, n

Si (m > 0 ET n > 0) OU (m < 0 ET n < 0)

Alors

Ecrire "Leur produit est positif"

Sinon

Ecrire "Leur produit est négatif"

Finsi

Fin

### Exercice 3.3

Variables a, b, c en Caractère

Début

Ecrire "Entrez successivement trois noms : "

Lire a, b, c

Si a < b ET b < c Alors

Ecrire "Ces noms sont classés  
alphabétiquement"

Sinon

Ecrire "Ces noms ne sont pas classés"

Finsi

Fin

### Exercice 3.4

Variable n en Entier

Début

Ecrire "Entrez un nombre : "

Lire n

Si n < 0 Alors

Ecrire "Ce nombre est négatif"

SinonSi n = 0 Alors

Ecrire "Ce nombre est nul"

Sinon

Ecrire "Ce nombre est positif"

Finsi

Fin

### Exercice 3.5

Variables m, n en Entier

Début

Ecrire "Entrez deux nombres : "

Lire m, n

Si m = 0 OU n = 0 Alors

Ecrire "Le produit est nul"

SinonSi (m < 0 ET n < 0) OU (m > 0 ET n > 0)

Alors

Ecrire "Le produit est positif"

Sinon

Ecrire "Le produit est négatif"

Finsi

Fin

Si on souhaite simplifier l'écriture de la condition lourde du SinonSi, on peut toujours passer par des variables booléennes intermédiaires. Une astuce de sioux consiste également à employer un Xor (c'est l'un des rares cas dans lesquels il est pertinent)

### Exercice 3.6

Variable age en Entier

Début

Ecrire "Entrez l'âge de l'enfant : "

Lire age

Si age >= 12 Alors

Ecrire "Catégorie Cadet"

SinonSi age >= 10 Alors

Ecrire "Catégorie Minime"

SinonSi age >= 8 Alors

Ecrire "Catégorie Pupille"

SinonSi age >= 6 Alors

Ecrire "Catégorie Poussin"

Finsi

Fin

On peut évidemment écrire cet algorithme de différentes façons, ne serait-ce qu'en commençant par la catégorie la plus jeune.

## PARTIE 4

### Exercice 4.1

Aucune difficulté, il suffit d'appliquer la règle de la transformation du OU en ET vue en cours (loi de Morgan). Attention toutefois à la rigueur dans la transformation des conditions en leur contraire...

Si Tutu <= Toto + 4 ET Tata <> "OK" Alors

Tutu ← Tutu - 1

Sinon

Tutu ← Tutu + 1

Finsi

### Exercice 4.2

Variables h, m en Numérique

Début

Ecrire "Entrez les heures, puis les minutes : "

Lire h, m

m ← m + 1

Si m = 60 Alors

m ← 0

```

    h ← h + 1
FinSi
Si h = 24 Alors
    h ← 0
FinSi
Ecrire "Dans une minute il sera ", h,
"heure(s) ", m, "minute(s)"
Fin

```

#### Exercice 4.3

**Variables** h, m, s **en Numérique**

**Début**

Ecrire "Entrez les heures, puis les minutes,  
puis les secondes : "

Lire h, m, s

s ← s + 1

Si s = 60 Alors

s ← 0

m ← m + 1

FinSi

Si m = 60 Alors

m ← 0

h ← h + 1

FinSi

Si h = 24 Alors

h ← 0

FinSi

Ecrire "Dans une seconde il sera ", h, "h",  
m, "m et ", s, "s"

Fin

#### Exercice 4.4

**Variables** n, p **en Numérique**

**Début**

Ecrire "Nombre de photocopies : "

Lire n

Si n ≤ 10 Alors

p ← n \* 0,1

SinonSi n ≤ 30 Alors

p ← 10 \* 0,1 + (n - 10) \* 0,09

Sinon

p ← 10 \* 0,1 + 20 \* 0,09 + (n - 30) \* 0,08

FinSi

Ecrire "Le prix total est: ", p

Fin

#### Exercice 4.5

**Variable** sex **en Caractère**

**Variable** age **en Numérique**

**Variables** C1, C2 **en Booléen**

**Début**

Ecrire "Entrez le sexe (M/F) : "

Lire sex

Ecrire "Entrez l'âge: "

Lire age

C1 ← sex = "M" ET age > 20

C2 ← sex = "F" ET (age > 18 ET age < 35)

Si C1 ou C2 Alors

Ecrire "Imposable"

Sinon

Ecrire "Non Imposable"

FinSi

Fin

#### Exercice 4.6

Cet exercice, du pur point de vue algorithmique, n'est pas très méchant. En revanche, il représente dignement la catégorie des énoncés piégés.

En effet, rien de plus facile que d'écrire : si le candidat a plus de 50%, il est élu, sinon s'il a plus de 12,5 %, il est au deuxième tour, sinon il est éliminé. Hé hé hé... mais il ne faut pas oublier que le candidat peut très bien avoir eu 20 % mais être tout de même éliminé, tout simplement parce que l'un des autres a fait plus de 50 % et donc qu'il n'y a pas de deuxième tour !...

Moralité : ne jamais se jeter sur la programmation avant d'avoir soigneusement mené l'analyse du problème à traiter.

**Variables** A, B, C, D **en Numérique**

**Début**

Ecrire "Entrez les scores des quatre prétendants :"

Lire A, B, C, D

C1 ← A > 50

C2 ← B > 50 ou C > 50 ou D > 50

C3 ← A ≥ B et A ≥ C et A ≥ D

C4 ← A ≥ 12,5

Si C1 Alors

Ecrire "Elu au premier tour"

SinonSi C2 ou Non(C4) Alors

Ecrire "Battu, éliminé, sorti !!!"

SinonSi C3 Alors

Ecrire "Ballotage favorable"

Sinon

Ecrire "Ballotage défavorable"

FinSi

Fin

#### Exercice 4.7

Là encore, on illustre l'utilité d'une bonne analyse. Je propose deux corrigés différents. Le premier suit l'énoncé pas à pas. C'est juste, mais c'est vraiment lourd. La deuxième version s'appuie sur une vraie compréhension d'une situation pas si embrouillée qu'elle n'en a l'air.

Dans les deux cas, un recours aux variables booléennes aère sérieusement l'écriture.

Donc, premier corrigé, on suit le texte de l'énoncé pas à pas :

**Variables** age, perm, acc, assur **en Numérique**

**Variables** C1, C2, C3 **en Booléen**

**Variable** situ **en Caractère**

**Début**

Ecrire "Entrez l'âge: "

Lire age

Ecrire "Entrez le nombre d'années de permis: "

"

Lire perm

Ecrire "Entrez le nombre d'accidents: "

Lire acc

Ecrire "Entrez le nombre d'années

```

d'assurance: "
Lire assur
C1 ← age >= 25
C2 ← perm >= 2
C3 ← assur > 1
Si Non(C1) et Non(C2) Alors
    Si acc = 0 Alors
        situ ← "Rouge"
    Sinon
        situ ← "Refusé"
    FinSi
SinonSi ((Non(C1) et C2) ou (C1 et Non(C2)))
Alors
    Si acc = 0 Alors
        situ ← "Orange"
    SinonSi acc = 1 Alors
        situ ← "Rouge"
    Sinon
        situ ← "Refusé"
    FinSi
Sinon
    Si acc = 0 Alors
        situ ← "Vert"
    SinonSi acc = 1 Alors
        situ ← "Orange"
    SinonSi acc = 2 Alors
        situ ← "Rouge"
    Sinon
        situ ← "Refusé"
    FinSi
FinSi
Si C3 Alors
    Si situ = "Rouge" Alors
        situ ← "Orange"
    SinonSi situ = "Orange" Alors
        situ ← "Orange"
    SinonSi situ = "vert" Alors
        situ ← "Bleu"
    FinSi
FinSi
Ecrire "votre situation : ", situ
Fin

```

Vous trouvez cela compliqué ? Oh, certes oui, ça l'est ! Et d'autant plus qu'en lisant entre les lignes, on pouvait s'apercevoir que ce galimatias de tarifs recouvre en fait une logique très simple : un système à points. Et il suffit de comptabiliser les points pour que tout s'éclaire... Reprenons juste après l'affectation des trois variables booléennes C1, C2, et C3. On écrit :

```

P ← 0
Si Non(C1) Alors
    P ← P + 1
FinSi
Si Non(C2) Alors
    P ← P + 1
FinSi
P ← P + acc
Si P < 3 et C3 Alors
    P ← P - 1

```

```

FinSi
Si P = -1 Alors
    situ ← "Bleu"
SinonSi P = 0 Alors
    situ ← "Vert"
SinonSi P = 1 Alors
    situ ← "Orange"
SinonSi P = 2 Alors
    situ ← "Rouge"
Sinon
    situ ← "Refusé"
FinSi
Ecrire "votre situation : ", situ
Fin
Cool, non ?

```

### Exercice 4.8

En ce qui concerne le début de cet algorithme, il n'y a aucune difficulté. C'est de la saisie bête et même pas méchante:

**variables** J, M, A, JMax **en Numérique**  
**variables** VJ, VM, B **en Booléen**

**Début**  
**Ecrire** "Entrez le numéro du jour"  
**Lire** J  
**Ecrire** "Entrez le numéro du mois"  
**Lire** M  
**Ecrire** "Entrez l'année"  
**Lire** A

C'est évidemment ensuite que les ennuis commencent... La première manière d'aborder la chose consiste à se dire que fondamentalement, la structure logique de ce problème est très simple. Si nous créons deux variables booléennes VJ et VM, représentant respectivement la validité du jour et du mois entrés, la fin de l'algorithme sera d'une simplicité biblique (l'année est valide par définition, si on évacue le débat byzantin concernant l'existence de l'année zéro) :

```

Si VJ et VM alors
    Ecrire "La date est valide"
Sinon
    Ecrire "La date n'est pas valide"
FinSi

```

Toute la difficulté consiste à affecter correctement les variables VJ et VM, selon les valeurs des variables J, M et A. Dans l'absolu, VJ et VM pourraient être les objets d'une affectation monstrueuse, avec des conditions atrocement composées. Mais franchement, écrire ces conditions en une seule fois est un travail de bénédictin sans grand intérêt. Pour éviter d'en arriver à une telle extrémité, on peut sérier la difficulté en créant deux variables supplémentaires :

**B** : variable booléenne qui indique s'il s'agit d'une année bissextile  
**JMax** : variable numérique qui indiquera le dernier jour valable pour le mois entré.

Avec tout cela, on peut y aller et en ressortir vivant.  
On commence par initialiser nos variables booléennes, puis on traite les années, puis les mois, puis les jours.

On note "dp" la condition "divisible par" :

$B \leftarrow A \text{ dp } 400 \text{ ou } (\text{non}(A \text{ dp } 100) \text{ et } A \text{ dp } 4)$

$J_{\text{max}} \leftarrow 0$

$VM \leftarrow M \geq 1 \text{ et } M \leq 12$

**Si** VM **Alors**

**Si** M = 2 et B **Alors**

$J_{\text{Max}} \leftarrow 29$

**SinonSi** M = 2 **Alors**

$J_{\text{Max}} \leftarrow 28$

**SinonSi** M = 4 ou M = 6 ou M = 9 ou M = 11

**Alors**

$J_{\text{Max}} \leftarrow 30$

**Sinon**

$J_{\text{Max}} \leftarrow 31$

**FinSi**

$VJ \leftarrow J \geq 1 \text{ et } J \leq J_{\text{max}}$

**FinSi**

Cette solution a le mérite de ne pas trop compliquer la structure des tests, et notamment de ne pas répéter l'écriture finale à l'écran. Les variables booléennes intermédiaires nous épargnent des conditions composées trop lourdes, mais celles-ci restent néanmoins sérieuses.

Une approche différente consisterait à limiter les conditions composées, quitte à le payer par une structure beaucoup plus exigeante de tests imbriqués. Là encore, on évite de jouer les extrémistes et l'on s'autorise quelques conditions composées lorsque cela nous simplifie l'existence. On pourrait aussi dire que la solution précédente "part de la fin" du problème (la date est elle valide ou non ?), alors que celle qui suit "part du début" (quelles sont les données entrées au clavier ?) :

**Si** M < 1 ou M > 12 **Alors**

**Ecrire** "Date Invalide"

**SinonSi** M = 2 **Alors**

**Si** A dp 400 **Alors**

**Si** J < 1 ou J > 29 **Alors**

**Ecrire** "Date Invalide"

**Sinon**

**Ecrire** "Date valide"

**FinSi**

**SinonSi** A dp 100 **Alors**

**Si** J < 1 ou J > 28 **Alors**

**Ecrire** "Date Invalide"

**Sinon**

**Ecrire** "Date valide"

**FinSi**

**SinonSi** A dp 4 **Alors**

**Si** J < 1 ou J > 28 **Alors**

**Ecrire** "Date Invalide"

**Sinon**

**Ecrire** "Date valide"

**FinSi**

**Sinon**

**Si** J < 1 ou J > 28 **Alors**

**Ecrire** "Date Invalide"

**Sinon**

**Ecrire** "Date valide"

**FinSi**

**FinSi**

**SinonSi** M = 4 ou M = 6 ou M = 9 ou M = 11

**Alors**

**Si** J < 1 ou J > 30 **Alors**

**Ecrire** "Date Invalide"

**Sinon**

**Ecrire** "Date valide"

**FinSi**

**Sinon**

**Si** J < 1 ou J > 31 **Alors**

**Ecrire** "Date Invalide"

**Sinon**

**Ecrire** "Date valide"

**FinSi**

**FinSi**

On voit que dans ce cas, l'alternative finale (Date valide ou invalide) se trouve répétée un grand nombre de fois. Ce n'est en soi ni une bonne, ni une mauvaise chose. C'est simplement une question de choix stylistique. Personnellement, j'avoue préférer assez nettement la première solution, qui fait ressortir beaucoup plus clairement la structure logique du problème (il n'y a qu'une seule alternative, autant que cette alternative ne soit écrite qu'une seule fois).

Il convient enfin de citer une solution très simple et élégante, un peu plus difficile peut-être à imaginer du premier coup, mais qui avec le recul apparaîtrait comme très immédiate. Sur le fond, cela consiste à dire qu'il y a quatre cas pour qu'une date soit valide : celui d'un jour compris entre 1 et 31 dans un mois à 31 jours, celui d'un jour compris entre 1 et 30 dans un mois à 30 jours, celui d'un jour compris entre 1 et 29 en février d'une année bissextile, et celui d'un jour de février compris entre 1 et 28. Ainsi :

$B \leftarrow (A \text{ dp } 4 \text{ et } \text{Non}(A \text{ dp } 100)) \text{ ou } A \text{ dp } 400$

$K1 \leftarrow (m=1 \text{ ou } m=3 \text{ ou } m=5 \text{ ou } m=7 \text{ ou } m=8 \text{ ou } m=10 \text{ ou } m=12) \text{ et } (J \geq 1 \text{ et } J \leq 31)$

$K2 \leftarrow (m=4 \text{ ou } m=6 \text{ ou } m=9 \text{ ou } m=11) \text{ et } (J \geq 1 \text{ et } J \leq 30)$

$K3 \leftarrow m=2 \text{ et } B \text{ et } J \geq 1 \text{ et } J \leq 29$

$K4 \leftarrow m=2 \text{ et } J \geq 1 \text{ et } J \leq 28$

**Si** K1 ou K2 ou K3 ou K4 **Alors**

**Ecrire** "Date valide"

**Sinon**

**Ecrire** "Date non valide"

**FinSi**

**Fin**

Tout est alors réglé avec quelques variables booléennes et quelques conditions composées, en un minimum de lignes de code.

La morale de ce long exercice - et non moins long corrigé, c'est qu'un problème de test un peu compliqué admet une pléiade de solutions justes...  
...Mais que certaines sont plus astucieuses que d'autres !

## PARTIE 5

### Exercice 5.1

**Variable** N **en Entier**

**Debut**

N ← 0

**Ecrire** "Entrez un nombre entre 1 et 3"

**TantQue** N < 1 ou N > 3

**Lire** N

**Si** N < 1 ou N > 3 **Alors**

**Ecrire** "Saisie erronée. Recommencez"

**FinSi**

**FinTantQue**

**Fin**

### Exercice 5.2

**Variable** N **en Entier**

**Debut**

N ← 0

**Ecrire** "Entrez un nombre entre 10 et 20"

**TantQue** N < 10 ou N > 20

**Lire** N

**Si** N < 10 **Alors**

**Ecrire** "Plus grand !"

**SinonSi** N > 20 **Alors**

**Ecrire** "Plus petit !"

**FinSi**

**FinTantQue**

**Fin**

### Exercice 5.3

**Variables** N, i **en Entier**

**Debut**

**Ecrire** "Entrez un nombre : "

**Lire** N

**Ecrire** "Les 10 nombres suivants sont : "

**Pour** i ← N + 1 à N + 10

**Ecrire** i

**i Suivant**

**Fin**

### Exercice 5.4

**Variables** N, i **en Entier**

**Debut**

**Ecrire** "Entrez un nombre : "

**Lire** N

**Ecrire** "La table de multiplication de ce nombre est : "

**Pour** i ← 1 à 10

**Ecrire** N, " x ", i, " = ", n\*i

**i Suivant**

**Fin**

### Exercice 5.5

**Variables** N, i, Som **en Entier**

**Debut**

**Ecrire** "Entrez un nombre : "

**Lire** N

Som ← 0

**Pour** i ← 1 à N

    Som ← Som + i

**i Suivant**

**Ecrire** "La somme est : ", Som

**Fin**

### Exercice 5.6

**Variables** N, i, F **en Entier**

**Debut**

**Ecrire** "Entrez un nombre : "

**Lire** N

F ← 1

**Pour** i ← 2 à N

    F ← F \* i

**i Suivant**

**Ecrire** "La factorielle est : ", F

**Fin**

### Exercice 5.7

**Variables** N, i, PG **en Entier**

**Debut**

PG ← 0

**Pour** i ← 1 à 20

**Ecrire** "Entrez un nombre : "

**Lire** N

**Si** i = 1 ou N > PG **Alors**

        PG ← N

**FinSi**

**i Suivant**

**Ecrire** "Le nombre le plus grand était : ", PG

**Fin**

En ligne 3, on peut mettre n'importe quoi dans PG, il suffit que cette variable soit affectée pour que le premier passage en ligne 7 ne provoque pas d'erreur.

Pour la version améliorée, cela donne :

**Variables** N, i, PG, IPG **en Entier**

**Debut**

PG ← 0

**Pour** i ← 1 à 20

**Ecrire** "Entrez un nombre : "

**Lire** N

**Si** i = 1 ou N > PG **Alors**

        PG ← N

        IPG ← i

**FinSi**

**i Suivant**

**Ecrire** "Le nombre le plus grand était : ", PG

**Ecrire** "Il a été saisi en position numéro ", IPG

**Fin**

### Exercice 5.8

**Variables** N, i, PG, IPG **en Entier**

**Debut**

N ← 1

i ← 0

PG ← 0

**TantQue** N <> 0

**Ecrire** "Entrez un nombre : "

**Lire** N

```

i ← i + 1
Si i = 1 ou N > PG Alors
    PG ← N
    IPG ← i
FinSi
FinTantQue
Ecrire "Le nombre le plus grand était : ",
PG
Ecrire "Il a été saisi en position numéro ",
IPG
Fin

```

### Exercice 5.9

**Variables** FF, somdue, M, IPG, Reste, Nb10F, Nb5F **En Entier**

**Debut**

E ← 1

somdue ← 0

**TantQue** E <> 0

**Ecrire** "Entrez le montant : "

**Lire** E

    somdue ← somdue + E

**FinTantQue**

**Ecrire** "Vous devez :", E, " euros"

**Ecrire** "Montant versé :"

**Lire** M

Reste ← M - E

Nb10E ← 0

**TantQue** Reste >= 10

    Nb10E ← Nb10E + 1

    Reste ← Reste - 10

**FinTantQue**

Nb5E ← 0

**Si** Reste >= 5

    Nb5E ← 1

    Reste ← Reste - 5

**FinSi**

**Ecrire** "Rendu de la monnaie :"

**Ecrire** "Billets de 10 E : ", Nb10E

**Ecrire** "Billets de 5 E : ", Nb5E

**Ecrire** "Pièces de 1 E : ", reste

**Fin**

### Exercice 5.10

Spontanément, on est tenté d'écrire l'algorithme suivant :

Variables N, P, i, Numé, Déno1, Déno2 en Entier

Debut **Ecrire** "Entrez le nombre de chevaux partants : "

**Lire** N

**Ecrire** "Entrez le nombre de chevaux joués : "

**Lire** P

Numé ← 1

**Pour** i ← 2 à N

    Numé ← Numé \* i

**i Suivant**

Déno1 ← 1

**Pour** i ← 2 à N-P

    Déno1 ← Déno1 \* i

**i Suivant**

Déno2 ← 1

**Pour** i ← 2 à P

    Déno2 ← Déno2 \* i

**i Suivant**

**Ecrire** "Dans l'ordre, une chance sur ", Numé / Déno1

**Ecrire** "Dans le désordre, une sur ", Numé / (Déno1 \* Déno2)

**Fin**

Cette version, formellement juste, comporte tout de même deux faiblesses.

La première, et la plus grave, concerne la manière dont elle calcule le résultat final. Celui-ci est le quotient d'un nombre par un autre ; or, ces nombres auront rapidement tendance à être très grands. En calculant, comme on le fait ici, d'abord le numérateur, puis ensuite le dénominateur, on prend le risque de demander à la machine de stocker des nombres trop grands pour qu'elle soit capable de les coder (cf. le préambule). C'est d'autant plus bête que rien ne nous oblige à procéder ainsi : on n'est pas obligé de passer par la division de deux très grands nombres pour obtenir le résultat voulu.

La deuxième remarque est qu'on a programmé ici trois boucles successives. Or, en y regardant bien, on peut voir qu'après simplification de la formule, ces trois boucles comportent le même nombre de tours ! (si vous ne me croyez pas, écrivez un exemple de calcul et biffez les nombres identiques au numérateur et au dénominateur). Ce triple calcul (ces trois boucles) peut donc être ramené(es) à un(e) seul(e). Et voilà le travail, qui est non seulement bien plus court, mais aussi plus performant :

**Variables** N, P, i, O, F **en Entier**

**Debut**

**Ecrire** "Entrez le nombre de chevaux partants : "

**Lire** N

**Ecrire** "Entrez le nombre de chevaux joués : "

**Lire** P

A ← 1

B ← 1

**Pour** i ← 1 à P

    A ← A \* (i + N - P)

    B ← B \* i

**i Suivant**

**Ecrire** "Dans l'ordre, une chance sur ", A

**Ecrire** "Dans le désordre, une chance sur ", A / B

**Fin**

# PARTIE 6

## Exercice 6.1

**Tableau** Truc(6) en Numérique

**Variable** i en Numérique

**Debut**

**Pour** i ← 0 à 6

Truc(i) ← 0

**i Suivant**

**Fin**

## Exercice 6.2

**Tableau** Truc(5) en Caractère

**Debut**

Truc(0) ← "a"

Truc(1) ← "e"

Truc(2) ← "i"

Truc(3) ← "o"

Truc(4) ← "u"

Truc(5) ← "y"

**Fin**

## Exercice 6.3

**Tableau** Notes(8) en Numérique

**Variable** i en Numérique

**Pour** i ← 0 à 8

Ecrire "Entrez la note numéro ", i + 1

Lire Notes(i)

**i Suivant**

**Fin**

## Exercice 6.4

Cet algorithme remplit un tableau avec six valeurs :

0, 1, 4, 9, 16, 25.

Il les écrit ensuite à l'écran. Simplification :

**Tableau** Nb(5) en Numérique

**Variable** i en Numérique

**Début**

**Pour** i ← 0 à 5

Nb(i) ← i \* i

Ecrire Nb(i)

**i Suivant**

**Fin**

## Exercice 6.5

Cet algorithme remplit un tableau avec les sept

valeurs : 1, 3, 5, 7, 9, 11, 13.

Il les écrit ensuite à l'écran. Simplification :

**Tableau** N(6) en Numérique

**Variables** i, k en Numérique

**Début**

N(0) ← 1

Ecrire N(0)

**Pour** k ← 1 à 6

N(k) ← N(k-1) + 2

Ecrire N(k)

**k Suivant**

**Fin**

## Exercice 6.6

Cet algorithme remplit un tableau de 8 valeurs : 1, 1,

2, 3, 5, 8, 13, 21

## Exercice 6.7

**Variable** s en Numérique

**Tableau** Notes(8) en Numérique

**Debut**

s ← 0

**Pour** i ← 0 à 8

Ecrire "Entrez la note n° ", i + 1

Lire Notes(i)

s ← s + Notes(i)

**i Suivant**

Ecrire "Moyenne :", s/9

**Fin**

## Exercice 6.8

**Variables** Nb, Nbpos, Nbneg en Numérique

**Tableau** T() en Numérique

**Debut**

Ecrire "Entrez le nombre de valeurs :"

Lire Nb

Redim T(Nb-1)

Nbpos ← 0

Nbneg ← 0

**Pour** i ← 0 à Nb - 1

Ecrire "Entrez le nombre n° ", i + 1

Lire T(i)

**Si** T(i) > 0 **alors**

Nbpos ← Nbpos + 1

**Sinon**

Nbneg ← Nbneg + 1

**Finsi**

**i Suivant**

Ecrire "Nombre de valeurs positives : ",  
Nbpos

Ecrire "Nombre de valeurs négatives : ",  
Nbneg

**Fin**

## Exercice 6.9

**Variables** i, Som, N en Numérique

**Tableau** T() en Numérique

**Debut**

... (on ne programme pas la saisie du tableau, dont on  
suppose qu'il compte N éléments)

Redim T(N-1)

...

Som ← 0

**Pour** i ← 0 à N - 1

Som ← Som + T(i)

**i Suivant**

Ecrire "Somme des éléments du tableau : ",  
Som

**Fin**

## Exercice 6.10

**Variables** i, N en Numérique

**Tableaux** T1(), T2(), T3() en Numérique

**Debut**

... (on suppose que T1 et T2 comptent N éléments, et  
qu'ils sont déjà saisis)

Redim T3(N-1)

...

**Pour** i ← 0 à N - 1

T3(i) ← T1(i) + T2(i)



**i Suivant**

**Fin**

#### **Exercice 6.11**

**Variables i, j, N1, N2, S en Numérique**

**Tableaux T1(), T2() en Numérique**

**Debut**

... On ne programme pas la saisie des tableaux T1 et T2.

On suppose que T1 possède N1 éléments, et que T2 en possède T2)

...

**S ← 0**

**Pour i ← 0 à N1 - 1**

**Pour j ← 0 à N2 - 1**

**S ← S + T1(i) \* T2(j)**

**j Suivant**

**i Suivant**

**Ecrire "Le schtroumpf est : ", S**

**Fin**

#### **Exercice 6.12**

**Variables Nb, i en Numérique**

**Tableau T() en Numérique**

**Debut**

**Ecrire "Entrez le nombre de valeurs : "**

**Lire Nb**

**Redim T(Nb-1)**

**Pour i ← 0 à Nb - 1**

**Ecrire "Entrez le nombre n° ", i + 1**

**Lire T(i)**

**i Suivant**

**Ecrire "Nouveau tableau : "**

**Pour i ← 0 à Nb - 1**

**T(i) ← T(i) + 1**

**Ecrire T(i)**

**i Suivant**

**Fin**

#### **Exercice 6.13**

**Variables Nb, Posmaxi en Numérique**

**Tableau T() en Numérique**

**Ecrire "Entrez le nombre de valeurs :"**

**Lire Nb**

**Redim T(Nb-1)**

**Pour i ← 0 à Nb - 1**

**Ecrire "Entrez le nombre n° ", i + 1**

**Lire T(i)**

**i Suivant**

**Posmaxi ← 0**

**Pour i ← 0 à Nb - 1**

**Si T(i) > T(Posmaxi) alors**

**Posmaxi ← i**

**Finsi**

**i Suivant**

**Ecrire "Element le plus grand : ",**

**T(Posmaxi)**

**Ecrire "Position de cet élément : ", Posmaxi**

**Fin**

#### **Exercice 6.14**

**Variables Nb, i, Som, Moy, Nbsup en**

**Numérique**

**Tableau T() en Numérique**

**Debut**

**Ecrire "Entrez le nombre de notes à saisir : "**

**Lire Nb**

**Redim T(Nb-1)**

**Pour i ← 0 à Nb - 1**

**Ecrire "Entrez le nombre n° ", i + 1**

**Lire T(i)**

**i Suivant**

**Som ← 0**

**Pour i ← 0 à Nb - 1**

**Som ← Som + T(i)**

**i Suivant**

**Moy ← Som / Nb**

**Nbsup ← 0**

**Pour i ← 0 à Nb - 1**

**Si T(i) > Moy Alors**

**Nbsup ← Nbsup + 1**

**Finsi**

**i Suivant**

**Ecrire Nbsup, " élèves dépassent la moyenne de la classe"**

**Fin**

## **PARTIE 7**

#### **Exercice 7.1**

**Variables Nb, i en Entier**

**Variable Flag en Booleen**

**Tableau T() en Entier**

**Debut**

**Ecrire "Entrez le nombre de valeurs :"**

**Lire Nb**

**Redim T(Nb-1)**

**Pour i ← 0 à Nb - 1**

**Ecrire "Entrez le nombre n° ", i + 1**

**Lire T(i)**

**i Suivant**

**Flag ← Vrai**

**Pour i ← 1 à Nb - 1**

**Si T(i) <> T(i - 1) + 1 Alors**

**Flag ← Faux**

**Finsi**

**i Suivant**

**Si Flag Alors**

**Ecrire "Les nombres sont consécutifs"**

**Sinon**

**Ecrire "Les nombres ne sont pas consécutifs"**

**Finsi**

**Fin**

Cette programmation est sans doute la plus spontanée, mais elle présente le défaut d'examiner la totalité du tableau, même lorsqu'on découvre dès le départ deux éléments non consécutifs. Aussi, dans le cas d'un grand tableau, est-elle dispendieuse en temps de traitement. Une autre manière de procéder serait de sortir de la boucle dès que deux éléments non consécutifs sont détectés. La deuxième partie de l'algorithme deviendrait donc :

```

i ← 1
TantQue T(i) = T(i - 1) + 1 et i < Nb - 1
    i ← i + 1
FinTantQue
Si T(i) = T(i - 1) + 1 Alors
    Ecrire "Les nombres sont consécutifs"
Sinon
    Ecrire "Les nombres ne sont pas
consécutifs"
Finsi

```

### Exercice 7.2

On suppose que N est le nombre d'éléments du tableau. Tri par insertion :

```

...
Pour i ← 0 à N - 2
    posmaxi = i
    Pour j ← i + 1 à N - 1
        Si t(j) > t(posmaxi) alors
            posmaxi ← j
        Finsi
    j suivant
    temp ← t(posmaxi)
    t(posmaxi) ← t(i)
    t(i) ← temp
i suivant
Fin

```

Tri à bulles :

```

...
Yapermut ← Vrai
TantQue Yapermut
    Yapermut ← Faux
    Pour i ← 0 à N - 2
        Si t(i) < t(i + 1) Alors
            temp ← t(i)
            t(i) ← t(i + 1)
            t(i + 1) ← temp
        Yapermut ← Vrai
    Finsi
    i suivant
FinTantQue
Fin

```

### Exercice 7.3

On suppose que n est le nombre d'éléments du tableau préalablement saisi

```

...
Pour i ← 0 à (N-1)/2
    Temp ← T(i)
    T(i) ← T(N-1-i)
    T(N-1-i) ← Temp
i suivant
Fin

```

### Exercice 7.4

```

...
Ecrire "Rang de la valeur à supprimer ?"
Lire s
Pour i ← s à N-2
    T(i) ← T(i+1)
i suivant
Redim T(N-1)
Fin

```

### Exercice 7.5

N est le nombre d'éléments du tableau Dico(), contenant les mots du dictionnaire, tableau préalablement rempli.

**Variables** Sup, Inf, Comp **en Entier**

**Variables** Fini **en Booléen**

**Début**

**Ecrire** "Entrez le mot à vérifier"

**Lire** Mot

On définit les bornes de la partie du tableau à considérer

Sup ← N - 1

Inf ← 0

Fin ← Faux

**TantQue** Non Fini

Comp désigne l'indice de l'élément à comparer. En bonne rigueur, il faudra veiller à ce que Comp soit bien un nombre entier, ce qui pourra s'effectuer de différentes manières selon les langages.

Comp ← (Sup + Inf)/2

Si le mot se situe avant le point de comparaison, alors la borne supérieure change, la borne inférieure ne bouge pas.

**Si** Mot < Dico(Comp) **Alors**

Sup ← Comp - 1

**Sinon**, c'est l'inverse

**Sinon**

Inf ← Comp + 1

**Finsi**

Fin ← Mot = Dico(Comp) ou Sup < Inf

**FinTantQue**

**Si** Mot = Dico(Comp) **Alors**

**Ecrire** "le mot existe"

**Sinon**

**Ecrire** "Il n'existe pas"

**Finsi**

**Fin**

## PARTIE 8

### Exercice 8.1

**Tableau** Truc(5, 12) **en Entier**

**Debut**

**Pour** i ← 0 à 5

**Pour** j ← 0 à 12

Truc(i, j) ← 0

j **Suivant**

i **Suivant**

**Fin**

### Exercice 8.2

Cet algorithme remplit un tableau de la manière suivante:

X(0, 0) = 1

X(0, 1) = 2

X(0, 2) = 3

X(1, 0) = 4

X(1, 1) = 5

X(1, 2) = 6

Il écrit ensuite ces valeurs à l'écran, dans cet ordre.

### Exercice 8.3

Cet algorithme remplit un tableau de la manière suivante:

```
x(0, 0) = 1
x(1, 0) = 4
x(0, 1) = 2
x(1, 1) = 5
x(0, 2) = 3
x(1, 2) = 6
```

Il écrit ensuite ces valeurs à l'écran, dans cet ordre.

### Exercice 8.4

Cet algorithme remplit un tableau de la manière suivante:

```
T(0, 0) = 0
T(0, 1) = 1
T(1, 0) = 1
T(1, 1) = 2
T(2, 0) = 2
T(2, 1) = 3
T(3, 0) = 3
T(3, 1) = 4
```

Il écrit ensuite ces valeurs à l'écran, dans cet ordre.

### Exercice 8.5

Version a : cet algorithme remplit un tableau de la manière suivante:

```
T(0, 0) = 1
T(0, 1) = 2
T(1, 0) = 3
T(1, 1) = 4
T(2, 0) = 5
T(2, 1) = 6
T(3, 0) = 7
T(3, 1) = 8
```

Il écrit ensuite ces valeurs à l'écran, dans cet ordre.

Version b : cet algorithme remplit un tableau de la manière suivante:

```
T(0, 0) = 1
T(0, 1) = 5
T(1, 0) = 2
T(1, 1) = 6
T(2, 0) = 3
T(2, 1) = 7
T(3, 0) = 4
T(3, 1) = 8
```

Il écrit ensuite ces valeurs à l'écran, dans cet ordre.

### Exercice 8.6

**Variables** i, j, iMax, jMax **en Numérique**

**Tableau** T(12, 8) **en Numérique**

Le principe de la recherche dans un tableau à deux dimensions est strictement le même que dans un tableau à une dimension, ce qui ne doit pas nous étonner. La seule chose qui change, c'est qu'ici le balayage requiert deux boucles imbriquées, au lieu d'une seule.

**Debut**

```
...
iMax ← 0
jMax ← 0
```

**Pour** i ← 0 à 12

**Pour** j ← 0 à 8

**Si** T(i, j) > T(iMax, jMax) **Alors**

iMax ← i

jMax ← j

**FinSi**

j **Suivant**

i **Suivant**

**Ecrire** "Le plus grand élément est ", T(iMax, jMax)

**Ecrire** "Il se trouve aux indices ", iMax, ";", jMax

**Fin**

### Exercice 8.7

**Variables** i, j, posi, posj, i2, j2 **en**

**Entier**

**Variables** Correct, MoveOK **en Booléen**

**Tableau** Damier(7, 7) **en Booléen**

**Tableau** Mouv(3, 1) **en Entier**

Le damier contenant un seul pion, on choisit de le coder à l'économie, en le représentant par un tableau de booléens à deux dimensions. Dans chacun des emplacements de ce damier, Faux signifie l'absence du pion, Vrai sa présence.

Par ailleurs, on emploie une méchante astuce, pas obligatoire, mais bien pratique dans beaucoup de situations. L'idée est de faire correspondre les choix possibles de l'utilisateur avec les mouvements du pion. On entre donc dans un tableau Mouv à deux dimensions, les déplacements du pion selon les quatre directions, en prenant soin que chaque ligne du tableau corresponde à une saisie de l'utilisateur. La première valeur étant le déplacement en i, la seconde le déplacement en j. Ceci nous épargnera par la suite de faire quatre fois les mêmes tests.

**Debut**

Choix 0 : pion en haut à droite

Mouv(0, 0) ← -1

Mouv(0, 1) ← -1

Choix 1 : pion en haut à gauche

Mouv(1, 0) ← -1

Mouv(1, 1) ← 1

Choix 2 : pion en bas à gauche

Mouv(2, 0) ← 1

Mouv(2, 1) ← -1

Choix 3 : pion en bas à droite

Mouv(3, 0) ← 1

Mouv(3, 1) ← 1

Initialisation du damier; le pion n'est pour le moment nulle part

**Pour** i ← 0 à 7

**Pour** j ← 0 à 7

Damier(i, j) ← Faux

j **suivant**

i **suivant**

Saisie de la coordonnée en i ("posi") avec contrôle de saisie

Correct ← Faux

**TantQue** Non Correct

**Ecrire** "Entrez la ligne de votre pion: "

**Lire** posi

**Si** posi >= 0 et posi <= 7 **Alors**

        Correct ← vrai

**Finsi**

**Fintantque**

Saisie de la coordonnée en j ("posj") avec contrôle de saisie

Correct ← Faux

**TantQue** Non Correct

**Ecrire** "Entrez la colonne de votre pion: "

**Lire** posj

**Si** posj >= 0 et posj <= 7 **Alors**

        Correct ← vrai

**Finsi**

**Fintantque**

Positionnement du pion sur le damier virtuel.

Damier(posi, posj) ← vrai

Saisie du déplacement, avec contrôle

**Ecrire** "Quel déplacement ?"

**Ecrire** " - 0: en haut à gauche"

**Ecrire** " - 1: en haut à droite"

**Ecrire** " - 2: en bas à gauche"

**Ecrire** " - 3: en bas à droite"

Correct ← Faux

**TantQue** Non Correct

**Lire** Dep

**Si** Dep >= 0 et Dep <= 3 **Alors**

        Correct ← vrai

**Finsi**

**FinTantQue**

i2 et j2 sont les futures coordonnées du pion. La variable booléenne MoveOK vérifie la validité de ce futur emplacement

i2 ← posi + Mouv(Dep, 0)

j2 ← posj + Mouv(Dep, 1)

MoveOK ← i2 >= 0 et i2 <= 7 et j2 >= 0 et j2 <= 7

Cas où le déplacement est valide

**Si** MoveOK **Alors**

    Damier(posi, posj) ← Faux

    Damier(i2, j2) ← vrai

Affichage du nouveau damier

**Pour** i ← 0 à 7

**Pour** j ← 0 à 7

**Si** Damier(i, j) **Alors**

**Ecrire** " o ";

**Sinon**

**Ecrire** " x ";

**Finsi**

        j suivant

**Ecrire** ""

    i suivant

**Sinon**

Cas où le déplacement n'est pas valide

**Ecrire** "Mouvement impossible"

**Finsi**

**Fin**

## PARTIE 9

### Exercice 9.1

A ← Sin(B)

Aucun problème

A ← Sin(A + B \* C)

Aucun problème

B ← Sin(A) - Sin(D)

Erreur ! D est en caractère

D ← Sin(A / B)

Aucun problème... si B est différent de zéro

C ← Cos(Sin(A))

Erreur ! Il manque une parenthèse fermante

### Exercice 9.2

Vous étiez prévenus, c'est bête comme chou ! Il suffit de se servir de la fonction Len, et c'est réglé :

**Variable** Mot en Caractère

**Variable** Nb en Entier

**Debut**

**Ecrire** "Entrez un mot : "

**Lire** Mot

    Nb ← Len(Mot)

**Ecrire** "Ce mot compte ", Nb, " lettres"

**Fin**

### Exercice 9.3

Là, on est obligé de compter par une boucle le nombre d'espaces de la phrase, et on en déduit le nombre de mots. La boucle examine les caractères de la phrase un par un, du premier au dernier, et les compare à l'espace.

**Variable** Bla en Caractère

**Variables** Nb, i en Entier

**Debut**

**Ecrire** "Entrez une phrase : "

**Lire** Bla

    Nb ← 0

**Pour** i ← 1 à Len(Bla)

**Si** Mid(Bla, i, 1) = " " **Alors**

            Nb ← Nb + 1

**Finsi**

    i suivant

**Ecrire** "Cette phrase compte ", Nb + 1, " mots"

**Fin**

### Exercice 9.4

Solution 1 : pour chaque caractère du mot, on pose une très douloureuse condition composée. Le moins que l'on puisse dire, c'est que ce choix ne se distingue pas par son élégance. Cela dit, il marche, donc après tout, pourquoi pas.

**Variable** Bla en Caractère

**Variables** Nb, i, j en Entier

**Debut**

**Ecrire** "Entrez une phrase : "

**Lire** Bla

    Nb ← 0

**Pour** i ← 1 à Len(Bla)

**Si** Mid(Bla, i, 1) = "a" ou Mid(Bla, i, 1) = "e" ou Mid(Bla, i, 1) = "i" ou Mid(Bla, i, 1) = "o" ou Mid(Bla, i, 1) = "u" ou Mid(Bla, i, 1) = "y" **Alors**

```

    Nb ← Nb + 1
  FinSi
i suivant
Ecrire "Cette phrase compte ", Nb, "
voyelles"
Fin

```

Solution 2 : on stocke toutes les voyelles dans une chaîne. Grâce à la fonction Trouve, on détecte immédiatement si le caractère examiné est une voyelle ou non. C'est nettement plus sympathique...

**Variables** Bla, Voy **en Caractère**  
**Variables** Nb, i, j **en Entier**  
**Début**  
 Ecrire "Entrez une phrase : "  
 Lire Bla  
 Nb ← 0  
 Voy ← "aeiouy"  
**Pour** i ← 1 à Len(Bla)  
   **Si** Trouve(Voy, Mid(Bla, i, 1)) <> 0 **Alors**  
     Nb ← Nb + 1  
**FinSi**  
 i suivant  
 Ecrire "Cette phrase compte ", Nb, "  
 voyelles"  
**Fin**

### Exercice 9.5

Il n'existe aucun moyen de supprimer directement un caractère d'une chaîne... autrement qu'en procédant par collage. Il faut donc concaténer ce qui se trouve à gauche du caractère à supprimer, avec ce qui se trouve à sa droite. Attention aux paramètres des fonctions Mid, ils n'ont rien d'évident !

**Variable** Bla **en Caractère**  
**Variables** Nb, i, j **en Entier**  
**Début**  
 Ecrire "Entrez une phrase : "  
 Lire Bla  
 Ecrire "Entrez le rang du caractère à supprimer : "  
 Lire Nb  
 L ← Len(Bla)  
 Bla ← Mid(Bla, 1, Nb - 1) & Mid(Bla, Nb + 1, L - Nb)  
 Ecrire "La nouvelle phrase est : ", Bla  
**Fin**

### Exercice 9.6

Sur l'ensemble des exercices de cryptographie, il y a deux grandes stratégies possibles :

- soit transformer les caractères en leurs codes ASCII. L'algorithme revient donc ensuite à traiter des nombres. Une fois ces nombres transformés, il faut les reconvertir en caractères.

- soit en rester au niveau des caractères, et procéder directement aux transformations à ce niveau. C'est cette dernière option qui est choisie ici, et pour tous les exercices de cryptographie à venir.

Pour cet exercice, il y a une règle générale : pour chaque lettre, on détecte sa position dans l'alphabet, et on la remplace par la lettre occupant la position suivante. Seul cas particulier, la vingt-sixième lettre (le Z) doit être codée par la première (le A), et non par la vingt-septième, qui n'existe pas !

**Variables** Bla, Cod, Alpha **en Caractère**

**Variables** i, Pos **en Entier**

**Début**

Ecrire "Entrez la phrase à coder : "

Lire Bla

Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

Cod ← ""

**Pour** i ← 1 à Len(Bla)

  Let ← Mid(Bla, i, 1)

**Si** Let <> "Z" **Alors**

    Pos ← Trouve(Alpha, Let)

    Cod ← Cod & Mid(Alpha, Pos + 1, 1)

**Sinon**

    Cod ← Cod & "A"

**FinSi**

i **Suivant**

Bla ← Cod

Ecrire "La phrase codée est : ", Bla

**Fin**

### Exercice 9.7

Cet algorithme est une généralisation du précédent. Mais là, comme on ne connaît pas d'avance le décalage à appliquer, on ne sait pas a priori combien de "cas particuliers", à savoir de dépassements au-delà du Z, il va y avoir.

Il faut donc trouver un moyen simple de dire que si on obtient 27, il faut en réalité prendre la lettre numéro 1 de l'alphabet, que si on obtient 28, il faut en réalité prendre la lettre 2, etc. Ce moyen simple existe : il faut considérer le reste de la division par 26, autrement dit le modulo.

Il y a une petite ruse supplémentaire à appliquer, puisque 26 doit rester 26 et ne pas devenir 0.

**Variable** Bla, Cod, Alpha **en Caractère**

**Variables** i, Pos, Décal **en Entier**

**Début**

Ecrire "Entrez le décalage à appliquer : "

Lire Décal

Ecrire "Entrez la phrase à coder : "

Lire Bla

Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

Cod ← ""

**Pour** i ← 1 à Len(Bla)

  Let ← Mid(Bla, i, 1)

  Pos ← Trouve(Alpha, Let)

  NouvPos ← Mod(Pos + Décal, 26)

**Si** NouvPos = 0 **Alors**

    NouvPos ← 26

**FinSi**

  Cod ← Cod & Mid(Alpha, NouvPos, 1)

i **Suivant**

Bla ← Cod

**Ecrire** "La phrase codée est : ", Bla  
**Fin**

### Exercice 9.8

Là, c'est assez direct.

**Variable** Bla, Cod, Alpha **en Caractère**

**Variables** i, Pos, Décal **en Entier**

**Début**

**Ecrire** "Entrez l'alphabet clé : "

**Lire** clé

**Ecrire** "Entrez la phrase à coder : "

**Lire** Bla

Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

Cod ← ""

**Pour** i ← 1 à Len(Bla)

Let ← Mid(Bla, i, 1)

Pos ← Trouve(Alpha, Let)

Cod ← Cod & Mid(clé, Pos, 1)

**i Suivant**

Bla ← Cod

**Ecrire** "La phrase codée est : ", Bla

**Fin**

### Exercice 9.9

Le codage de Vigenère n'est pas seulement plus difficile à briser; il est également un peu plus raide à programmer. La difficulté essentielle est de comprendre qu'il faut deux boucles: l'une pour parcourir la phrase à coder, l'autre pour parcourir la clé. Mais quand on y réfléchit bien, ces deux boucles ne doivent surtout pas être imbriquées. Et en réalité, quelle que soit la manière dont on l'écrit, elle n'en forment qu'une seule.

**Variables** Alpha, Bla, Cod, clé, Let **en Caractère**

**Variables** i, Pos, PosClé, Décal **en Entier**

**Début**

**Ecrire** "Entrez la clé : "

**Lire** clé

**Ecrire** "Entrez la phrase à coder : "

**Lire** Bla

Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

Cod ← ""

PosClé ← 0

**Pour** i ← 1 à Len(Bla)

On gère la progression dans la clé. J'ai effectué cela "à la main" par une boucle, mais un joli emploi de la fonction Modulo aurait permis une programmation en une seule ligne!

PosClé ← PosClé + 1

**Si** PosClé > Len(clé) **Alors**

PosClé ← 1

**FinSi**

On détermine quelle est la lettre clé et sa position dans l'alphabet

LetClé ← Mid(clé, PosClé, 1)

PosLetClé ← Trouve(Alpha, LetClé)

On détermine la position de la lettre à coder et le décalage à appliquer. Là encore, une solution alternative aurait été d'employer Mod : cela nous aurait épargné le Si...

Let ← Mid(Bla, i, 1)

Pos ← Trouve(Alpha, Let)

NouvPos ← Pos + PosLetClé

**Si** NouvPos > 26 **Alors**

NouvPos ← NouvPos - 26

**FinSi**

Cod ← Cod & Mid(Alpha, NouvPos, 1)

**i Suivant**

Bla ← Cod

**Ecrire** "La phrase codée est : ", Bla

**Fin**

### Exercice 9.10

On en revient à des choses plus simples...

**Variable** Nb **en Entier**

**Ecrire** "Entrez votre nombre : "

**Lire** Nb

**Si** Nb/2 = Ent(Nb/2) **Alors**

**Ecrire** "Ce nombre est pair"

**Sinon**

**Ecrire** "Ce nombre est pair"

**FinSi**

**Fin**

### Exercice 9.11

a) Glup ← Alea() \* 2

b) Glup ← Alea() \* 2 - 1

c) Glup ← Alea() \* 0,30 + 1,35

d) Glup ← Ent(Alea() \* 6) + 1

e) Glup ← Alea() \* 17 - 10,5

f) Glup ← Ent(Alea()\*6) + Ent(Alea()\*6) + 2

## PARTIE 10

### Exercice 10.1

Cet algorithme écrit l'intégralité du fichier

"Exemple.txt" à l'écran

### Exercice 10.2

**Variable** Truc **en Caractère**

**Variable** i **en Entier**

**Debut**

**Ouvrir** "Exemple.txt" sur 5 **en Lecture**

**Tantque** Non EOF(5)

**LireFichier** 5, Truc

**Pour** i ← 1 à Len(Truc)

**Si** Mid(Truc, i, 1) = "/" **Alors**

**Ecrire** " "

**Sinon**

**Ecrire** Mid(Truc, i, 1)

**FinSi**

**i Suivant**

**FinTantQue**

**Fermer** 5

### Exercice 10.3

**Variables** Nom \* 20, Prénom \* 17, Tel \* 10,

Mail \* 20, Lig **en Caractère**

**Debut**

**Ecrire** "Entrez le nom : "

**Lire** Nom

**Ecrire** "Entrez le prénom : "

**Lire** Prénom

**Ecrire** "Entrez le téléphone : "

**Lire** Tel

**Ecrire** "Entrez le nom : "

**Lire** Mail

Lig ← Nom & Prénom & Tel & Mail

**Ouvrir** "Adresse.txt" sur 1 **pour Ajout**

**EcrireFichier** 1, Lig

**Fermer** 1

**Fin**

#### Exercice 10.4

Là, comme indiqué dans le cours, on passe par un tableau de structures en mémoire vive, ce qui est la technique la plus fréquemment employée. Le tri - qui est en fait un simple test - sera effectué sur le premier champ (nom).

**Structure** Bottin

Nom **en Caractère** \* 20

Prénom **en Caractère** \* 15

Tel **en Caractère** \* 10

Mail **en Caractère** \* 20

**Fin Structure**

**Tableau** Mespotes() **en Bottin**

**Variables** MonPote, Nouveau **en Bottin**

**Variables** i, j **en Numérique**

**Debut**

**Ecrire** "Entrez le nom : "

**Lire** Nouveau.Nom

**Ecrire** "Entrez le prénom : "

**Lire** Nouveau.Prénom

**Ecrire** "Entrez le téléphone : "

**Lire** Nouveau.Tel

**Ecrire** "Entrez le mail : "

**Lire** Nouveau.Mail

On recopie l'intégralité de "Adresses" dans MesPotes().

Et après tout, c'est l'occasion : quand on tombe au bon endroit, on insère subrepticement notre nouveau copain dans le tableau.

**Ouvrir** "Adresse.txt" sur 1 **pour Lecture**

i ← -1

inséré ← Faux

**Tantque** Non EOF(1)

i ← i + 1

**Redim** MesPotes(i)

**LireFichier** 1, MonPote

**Si** MonPote.Nom > Nouveau.Nom **et** Non Inséré

**Alors**

MesPotes(i) ← Nouveau

Inséré ← Vrai

i ← i + 1

**Redim** MesPotes(i)

**FinSi**

MesPotes(i) ← MonPote

**FinTantQue**

**Fermer** 1

Et le tour est quasiment joué. Il ne reste plus qu'à rebalancer tel quel l'intégralité du tableau MesPotes dans le fichier, en écrasant l'ancienne version.

**Ouvrir** "Adresse.txt" sur 1 **pour Ecriture**

**Pour** j ← 0 à i

**EcrireFichier** 1, MesPotes(j)

j **suivant**

**Fermer** 1

**Fin**

#### Exercice 10.5

C'est un peu du même tonneau que ce qu'on vient de faire, à quelques variantes près. Il y a essentiellement une petite gestion de flag pour faire bonne mesure.

**Structure** Bottin

Nom **en Caractère** \* 20

Prénom **en Caractère** \* 15

Tel **en caractère** \* 10

Mail **en Caractère** \* 20

**Fin Structure**

**Tableau** Mespotes() **en Bottin**

**Variables** MonPote **en Bottin**

**Variables** Ancien, Nouveau **en Caractère**\*20

**Variables** i, j **en Numérique**

**Variable** Trouvé **en Booléen**

**Debut**

**Ecrire** "Entrez le nom à modifier : "

**Lire** Ancien

**Ecrire** "Entrez le nouveau nom : "

**Lire** Nouveau

On recopie l'intégralité de "Adresses" dans Fic, tout en recherchant le clampin. Si on le trouve, on procède à la modification.

**Ouvrir** "Adresse.txt" sur 1 **pour Lecture**

i ← -1

Trouvé ← Faux

**Tantque** Non EOF(1)

i ← i + 1

**Redim** MesPotes(i)

**LireFichier** 1, MonPote

**Si** MonPote.Nom = Ancien.Nom **Alors**

Trouvé ← Vrai

MonPote.Nom ← Nouveau

**FinSi**

MesPotes(i) ← MonPote

**FinTantQue**

**Fermer** 1

On recopie ensuite l'intégralité de Fic dans "Adresse"

**Ouvrir** "Adresse.txt" sur 1 **pour Ecriture**

**Pour** j ← 0 à i

**EcrireFichier** 1, MesPotes(j)

j **suivant**

**Fermer** 1

Et un petit message pour finir !

**Si** Trouvé **Alors**

**Ecrire** "Modification effectuée"

**Sinon**

**Ecrire** "Nom inconnu. Aucune modification effectuée"

**FinSi**

**Fin**

#### Exercice 10.6

Là, c'est un tri sur un tableau de structures, rien de plus facile. Et on est bien content de disposer des structures, autrement dit de ne se coltiner qu'un seul tableau...

**Structure** Bottin Nom **en Caractère** \* 20

Prénom **en Caractère** \* 15

Tel **en caractère** \* 10

```

Mail en Caractère * 20
Fin Structure
Tableau Mespotes() en Bottin
Variables Mini en Bottin
Variables i, j en Numérique
Debut
On recopie l'intégralité de "Adresses" dans MesPotes...
Ouvrir "Adresse.txt" sur 1 pour Lecture
i ← -1
Tantque Non EOF(1)
    i ← i + 1
    Redim MesPotes(i)
    LireFichier 1, MesPotes(i)
FinTantQue
Fermer 1
On trie le tableau selon l'algorithme de tri par
insertion déjà étudié, en utilisant le champ Nom de la
structure :
Pour j ← 0 à i - 1
    Mini ← MesPotes(j)
    posmini ← j
    Pour k ← j + 1 à i
        Si MesPotes(k).Nom < Mini.Nom Alors
            mini ← MesPotes(k)
            posmini ← k
        Finsi
    k suivant
    MesPotes(posmini) ← MesPotes(j)
    MesPotes(j) ← Mini
j suivant
On recopie ensuite l'intégralité du tableau dans
"Adresse"
Ouvrir "Adresse.txt" sur 1 pour Ecriture
Pour j ← 0 à i
    EcrireFichier 1, MesPotes(j)
j suivant
Fermer 1
Fin

```

#### Exercice 10.7

Bon, celui-là est tellement idiot qu'on n'a même pas besoin de passer par des tableaux en mémoire vive.

**Variable Lig en Caractère**

```

Début
Ouvrir "Tutu.txt" sur 1 pour Ajout
Ouvrir "Toto.txt" sur 2 pour Lecture
Tantque Non EOF(2)
    LireFichier 2, Lig
    EcrireFichier 1, Lig
FinTantQue
Fermer 2
Ouvrir "Tata.txt" sur 3 pour Lecture
Tantque Non EOF(3)
    LireFichier 2, Lig
    EcrireFichier 1, Lig
FinTantQue
Fermer 3
Fermer 1
Fin

```

#### Exercice 10.8

On va éliminer les mauvaises entrées dès la recopie : si l'enregistrement ne présente pas un mail valide, on l'ignore, sinon on le copie dans le tableau.

```

Structure Bottin
    Nom en Caractère * 20
    Prénom en Caractère * 15
    Tel en caractère * 10
    Mail en Caractère * 20
Fin Structure
Tableau Mespotes() en Bottin
Variable MonPote en Bottin
Variables i, j en Numérique
Debut
On recopie "Adresses" dans MesPotes en testant le
mail...
Ouvrir "Adresse.txt" sur 1 pour Lecture
i ← -1
Tantque Non EOF(1)
    LireFichier 1, MonPote
    nb ← 0
    Pour i ← 1 à Len(MonPote.Mail)
        Si Mid(MonPote.Mail, i, 1) = "@" Alors
            nb ← nb + 1
        Finsi
    i suivant
    Si nb = 1 Alors
        i ← i + 1
        Redim MesPotes(i)
        MesPotes(i) ← MonPote
    Finsi
FinTantQue
Fermer 1
On recopie ensuite l'intégralité de Fic dans "Adresse"
Ouvrir "Adresse.txt" sur 1 pour Ecriture
Pour j ← 0 à i
    EcrireFichier 1, MesPotes(j)
j Suivant
Fermer 1
Fin

```

#### Exercice 10.9

Une fois de plus, le passage par un tableau de structures est une stratégie commode. Attention toutefois, comme il s'agit d'un fichier texte, tout est stocké en caractère. Il faudra donc convertir en numérique les caractères représentant les ventes, pour pouvoir effectuer les calculs demandés. Pour le traitement, il y a deux possibilités. Soit on recopie le fichier à l'identique dans un premier tableau, et on traite ensuite ce tableau pour faire la somme par vendeur. Soit on fait le traitement directement, dès la lecture du fichier. C'est cette option qui est choisie dans ce corrigé.

```

Structure Vendeur
    Nom en Caractère * 20
    Montant en Numérique
Fin Structure
Tableau MesVendeurs() en Vendeur
Variables NomPrec * 20, Lig, Nom en

```



**caractère**

**Variables** Somme, Vente **en Numérique**

On balaye le fichier en faisant nos additions.

Dès que le nom a changé (on est passé au vendeur suivant), on range le résultat et on remet tout à zéro

**Debut**

**Ouvrir** "Ventes.txt" sur 1 **pour Lecture**

$i \leftarrow -1$

Somme  $\leftarrow 0$

NomPréc  $\leftarrow ""$

**Tantque** Non EOF(1)

**LireFichier** 1, Lig

    Nom  $\leftarrow$  Mid(Lig, 1, 20)

    Vente  $\leftarrow$  CNum(Mid(Lig, 21, 10))

**Si** Nom = NomPréc **Alors**

        Somme  $\leftarrow$  Somme + Vente

**Sinon**

$i \leftarrow i + 1$

**Redim** MesVendeurs(i)

        MesVendeurs(i).Nom  $\leftarrow$  NomPréc

        MesVendeurs(i).Montant  $\leftarrow$  Somme

        Somme  $\leftarrow 0$

        NomPréc  $\leftarrow$  Nom

**FinSi**

**FinTantQue**

Et n'oublions pas un petit tour de plus pour le dernier de ces messieurs...

$i \leftarrow i + 1$

**Redim** MesVendeurs(i)

MesVendeurs(i).Nom  $\leftarrow$  NomPréc

MesVendeurs(i).Montant  $\leftarrow$  Somme

**Fermer** 1

Pour terminer, on affiche le tableau à l'écran

**Pour** j  $\leftarrow 0$  à i

**Ecrire** MesVendeurs(j)

j **suivant**

**Fin**

## PARTIE 11

### Exercice 11.1

Voilà un début en douceur...

**Fonction** Sum(a, b, c, d, e)

**Renvoyer** a + b + c + d + e

**FinFonction**

### Exercice 11.2

**Fonction** NbVoyelles(Mot en Caractère)

**Variables** i, nb **en Numérique**

**Pour** i  $\leftarrow 1$  à Len(Mot)

**Si** Trouve("aeiouy", Mid(Mot, i, 1))  $\neq 0$

**Alors**

    nb  $\leftarrow$  nb + 1

**FinSi**

i **suivant**

**Renvoyer** nb

**FinFonction**

### Exercice 11.3

**Fonction** Trouve(a, b)

**Variable** i **en Numérique**

**Début**

i  $\leftarrow 1$

**TantQue** i < Len(a) - Len(b) et b  $\neq$  Mid(a, i, Len(b))

$i \leftarrow i + 1$

**FinTantQue**

**Si** b  $\neq$  Mid(a, i, Len(b)) **Alors**

**Renvoyer** 0

**Sinon**

**Renvoyer** i

**FinFonction**

**Fonction** ChoixDuMot

Quelques explications : on lit intégralement le fichier contenant la liste des mots. Au fur et à mesure, on range ces mots dans le tableau Liste, qui est redimensionné à chaque tour de boucle. Un tirage aléatoire intervient alors, qui permet de renvoyer un des mots au hasard.

**Fonction** ChoixDuMot()

**Tableau** Liste() **en Caractère**

**Variables** Nbmots, Choisi **en Numérique**

**Ouvrir** "Dico.txt" sur 1 **en Lecture**

Nbmots  $\leftarrow -1$

**Tantque** Non EOF(1)

    Nbmots  $\leftarrow$  Nbmots + 1

**Redim** Liste(Nbmots)

**LireFichier** 1, Liste(Nbmots)

**FinTantQue**

**Fermer** 1

Choisi  $\leftarrow$  Ent(Alea() \* Nbmots)

**Renvoyer** Liste(Choisi)

**FinFonction**

**Fonction** PartieFinie

On commence par vérifier le nombre de mauvaises réponses, motif de défaite. Ensuite, on regarde si la partie est gagnée, traitement qui s'apparente à une gestion de Flag : il suffit que l'une des lettres du mot à deviner n'ait pas été trouvée pour que la partie ne soit pas gagnée. La fonction aura besoin, comme arguments, du tableau Verif, de son nombre d'éléments et du nombre actuel de mauvaises réponses.

**Fonction** PartieFinie(t() **en Booléen**, n, x **en Numérique**)

**Variables** i, issue **en Numérique**

**Si** x = 10 **Alors**

**Renvoyer** 2

**Sinon**

    Issue  $\leftarrow 1$

**Pour** i  $\leftarrow 0$  à n

**Si** Non t(i) **Alors**

            Issue  $\leftarrow 0$

**FinSi**

    i **suivant**

**Renvoyer** Issue

**FinSi**

**FinFonction**

**Procédure** AffichageMot

Une même boucle nous permet de considérer une par une les lettres du mot à trouver (variable m), et de savoir si ces lettres ont été identifiées ou non.

**Procédure** AffichageMot(**m en Caractère par Valeur**, **t()** **en Booléen par Valeur**)

**Variable** Aff **en Caractere**

**Variable** i **en Numérique**

Aff ← ""

**Pour** i ← 0 à len(m) - 1

**Si** Non t(i) **Alors**

        Aff ← Aff & "-"

**Sinon**

        Aff ← Aff & Mid(mot, i + 1, 1)

**FinSi**

i suivant

**Ecrire** Aff

**FinProcédure**

**Remarque** : cette procédure aurait également pu être écrite sous la forme d'une fonction, qui aurait renvoyé vers la procédure principale la chaîne de caractères Aff. L'écriture à l'écran de cette chaîne Aff aurait alors été faite par la procédure principale. Voilà donc une situation où on peut assez indifféremment opter pour une sous-procédure ou pour une fonction.

**Procédure** SaisieLettre

On vérifie que le signe entré (paramètre b) est bien une seule lettre, qui ne figure pas dans les propositions précédemment effectuées (paramètre a)

**Procédure** SaisieLettre(a, b **en Caractère par Référence**)

**Variable** Correct **en Booléen**

**Variable** Alpha **en Caractere**

**Début**

Correct ← Faux

Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

**TantQue** Non Correct

**Ecrire** "Entrez la lettre proposée : "

**Lire** b

**Si** Trouve(alpha, b) = 0 Ou len(b) <> 1

**Alors**

**Ecrire** "Ce n'est pas une lettre !"

**SinonSi** Trouve(a, b) <> 0 **Alors**

**Ecrire** "Lettre déjà proposée !"

**Sinon**

        Correct ← Vrai

        a ← a & b

**FinSi**

**FinTantQue**

**Fin Procédure**

**Procédure** VerifLettre

Les paramètres se multiplient... L est la lettre proposée, t() le tableau de booléens, M le mot à trouver et N le nombre de mauvaises propositions. Il n'y a pas de difficulté majeure dans cette procédure : on examine les lettres de M une à une, et on en tire les conséquences. Le flag sert à savoir si la lettre proposée faisait ou non partie du mot à deviner.

**Procédure** VerifLettre(L, M **en Caractère par Valeur**, t() **en Booléen par Référence**, N **en Numérique par Référence**)

**Variable** Correct **en Booléen**

**Début**

Correct ← Faux

**Pour** i ← 1 à Len(M)

**Si** Mid(M, i, 1) = L **Alors**

        Correct ← Vrai

        T(i - 1) ← Vrai

**FinSi**

**FinTantQue**

**Si** Non Correct **Alors**

    N ← N + 1

**FinSi**

**Fin Procédure**

**Procédure** Epilogue

**Procédure** Epilogue(M **en Caractère par Valeur**, N **en Numérique par Valeur**)

**Début**

**Si** N = 2 **Alors**

**Ecrire** "Une mauvaise proposition de trop... Partie terminée !"

**Ecrire** "Le mot à deviner était : ", M

**Sinon**

**Ecrire** "Bravo ! Vous avez trouvé !"

**FinSi**

**Fin Procédure**

**Procédure** Principale

**Procédure** Principale

**Variables** Lettre, Mot, Propos **en Caractere**

**Variables** g i, MovRep **en Numérique**

**Tableau** verif() **en Booléen**

**Début**

Mot ← ChoixDuMot()

Propos ← ""

Lettre ← ""

**Redim** Verif(Len(Mot)-1)

**Pour** i ← 0 à Len(Mot)-1

    Verif(i) ← Faux

i suivant

k ← 0

**Tantque** k = 0

    AffichageMot(Mot, Verif())

    SaisieLettre(Propos, Lettre)

    VerifLettre(Lettre, Mot, Verif(), MovRep)

    k ← PartieFinie(Verif(), len(mot), MovRep)

**FinTantQue**

Epilogue(Mot, k)

**Fin**