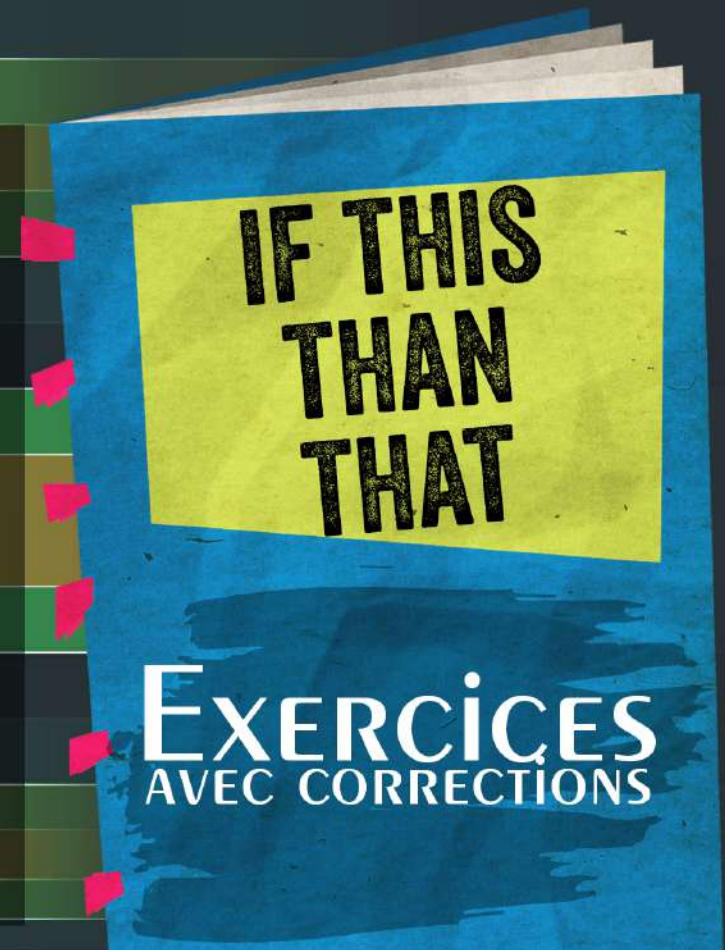


TECHNIQUES DES RÉSEAUX INFORMATIQUES

ALGORITHMIQUE

Sommaire :

- LES TESTS
- LES BOUCLES
- LES TABLEAUX
- LES STRUCTURES
- LES PROCÉDURES
- LES FONCTIONS



Source : <http://pise.info/algo/>

Énoncé 1 :

Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif (on laisse de côté le cas où le nombre vaut zéro).

Corrigé 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

Énoncé 2 :

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif (on laisse de côté le cas où le produit est nul). Attention toutefois : on ne doit **pas** calculer le produit des deux nombres.

Corrigé 2 :

Variables m, n en Entier

Début

Ecrire "Entrez deux nombres : "

Lire m, n

Si $(m > 0 \text{ ET } n > 0) \text{ OU } (m < 0 \text{ ET } n < 0)$ **Alors**

Ecrire "Leur produit est positif"

Sinon

Ecrire "Leur produit est négatif"

Finsi

Fin

Les Tests (e01-e05)

Énoncé 3 :

Ecrire un algorithme qui demande trois noms à l'utilisateur et l'informe ensuite s'ils sont rangés ou non dans l'ordre alphabétique.

Corrigé 3 :

Variables a, b, c en Caractère

Début

Ecrire "Entrez successivement trois noms : "

Lire a, b, c

Si $a < b \text{ ET } b < c$ **Alors**

Ecrire "Ces noms sont classés alphabétiquement"

Sinon

Ecrire "Ces noms ne sont pas classés"

Finsi

Fin

Énoncé 4 :

Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif (on inclut cette fois le traitement du cas où le nombre vaut zéro).

Corrigé 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

Énoncé 5 :

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si le produit est négatif ou positif (on inclut cette fois le traitement du cas où le produit peut être nul). Attention toutefois, on ne doit pas calculer le produit !

Corrigé 5 :

Variables m, n en Entier

Début

Ecrire "Entrez deux nombres : "

Lire m, n

Si $m = 0 \text{ OU } n = 0$ **Alors**

Ecrire "Le produit est nul"

SinonSi $(m < 0 \text{ ET } n < 0) \text{ OU } (m > 0 \text{ ET } n > 0)$ **Alors**

Ecrire "Le produit est positif"

Sinon

Ecrire "Le produit est négatif"

Finsi

Fin

Source : <http://pise.info/algo/>

Énoncé 1 :

Ecrire un algorithme qui déclare et remplisse un tableau de 7 valeurs numériques en les mettant toutes à zéro.

Corrigé 1 :

Tableau Truc(6) en Numérique

Variable i en Numérique

Debut

Pour i ← 0 à 6

Truc(i) ← 0

i Suivant

Fin

Énoncé 2 :

Ecrire un algorithme qui déclare et remplisse un tableau contenant les six voyelles de l'alphabet latin.

Corrigé 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

Énoncé 3 :

Ecrire un algorithme qui déclare un tableau de 9 notes, dont on fait ensuite saisir les valeurs par l'utilisateur.

Les Tableaux (e01-e05)

Corrigé 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

Énoncé 4 :

Ecrivez la fin de l'algorithme 3 afin que le calcul de la moyenne des notes soit effectué et affiché à l'écran.

Corrigé 4 :

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

Énoncé 5 :

Ecrivez un algorithme permettant à l'utilisateur de saisir un nombre quelconque de valeurs, qui devront être stockées dans un tableau. L'utilisateur doit donc commencer par entrer le nombre de valeurs qu'il compte saisir. Il effectuera ensuite cette saisie. Enfin, une fois la saisie terminée, le programme affichera le

nombre de valeurs négatives et le nombre de valeurs positives.

Corrigé 5 :

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

Source : <http://pise.info/algo/>

Énoncé 1 :

Ecrivez un algorithme calculant la somme des valeurs d'un tableau (on suppose que le tableau a été préalablement saisi)

Corrigé 1 :

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

Énoncé 2 :

Ecrivez un algorithme constituant un tableau, à partir de deux tableaux de même longueur préalablement saisis. Le nouveau tableau sera la somme des éléments des deux tableaux de départ.

Tableau 1 :

4	8	7	9	1	5	4	6
---	---	---	---	---	---	---	---

Tableau 2 :

7	6	5	2	1	3	7	4
---	---	---	---	---	---	---	---

Tableau à constituer :

11	14	12	11	2	8	11	10
----	----	----	----	---	---	----	----

Les Tableaux (e06-e10)

Corrigé 2 :

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

Énoncé 3 :

Toujours à partir de deux tableaux précédemment saisis, écrivez un algorithme qui calcule le schtroumpf des deux tableaux. Pour calculer le schtroumpf, il faut multiplier chaque élément du tableau 1 par chaque élément du tableau 2, et additionner le tout. Par exemple si l'on a :

Tableau 1 :

4	8	7	12
---	---	---	----

Tableau 2 :

3	6
---	---

Le Schtroumpf sera :

$3 * 4 + 3 * 8 + 3 * 7 + 3 * 12 + 6 * 4 + 6 * 8 + 6 * 7 + 6 * 12 = 279$

Corrigé 3 :

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

Tableaux T1(), T2() **en Numérique**

Debut

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

Énoncé 4 :

Ecrivez un algorithme qui permette la saisie d'un nombre quelconque de valeurs, sur le principe de l'ex 6.8. Toutes les valeurs doivent être ensuite augmentées de 1, et le nouveau tableau sera affiché à l'écran.

Corrigé 4 :

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

Source : <http://pise.info/algo/>

Énoncé 1 :

Soit un tableau T à deux dimensions (12, 8) préalablement rempli de valeurs numériques. Écrire un algorithme qui recherche la plus grande valeur au sein de ce tableau.

Corrigé 1 :

Variables i, j, iMax, jMax **en Numérique**
Tableau T(12, 8) **en Numérique**
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
```

Énoncé 2 :

Écrire l'algorithme effectuant le décalage des éléments d'un tableau.

Exemple :

Tableau initial D E C A L A G E •

Tableau modifié(décalage a gauche)) E C A L A G E D

Les Tableaux (e10-e15)

Corrigé 2 :

Procédure Decalage_gauche (T: Tableau de caractère, N: entier)
VAR tmp: caractère i: entier
Debut
 tmp ← T[1]
Pour i ← 1 a N-1 **Faire**
 T[i] ← T[i+1]
Fin Pour
 T[N] ← tmp
Fin

Énoncé 3 :

Toujours à partir de deux tableaux précédemment saisis, écrivez un algorithme qui calcule le schtroumpf des deux tableaux. Pour calculer le schtroumpf, il faut multiplier chaque élément du tableau 1 par chaque élément du tableau 2, et additionner le tout. Par exemple si l'on a :

Tableau 1 :

4	8	7	12
---	---	---	----

Tableau 2 :

3	6
---	---

Le Schtroumpf sera :

$3 * 4 + 3 * 8 + 3 * 7 + 3 * 12 + 6 * 4 + 6 * 8 + 6 * 7 + 6 * 12 = 279$

Corrigé 3 :

Variables i, j, N1, N2, S **en Numérique**
Tableaux T1(), T2() **en Numérique**
Debut

```
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
```

Énoncé 4 :

Ecrivez un algorithme qui permette la saisie d'un nombre quelconque de valeurs, sur le principe de l'ex 6.8. Toutes les valeurs doivent être ensuite augmentées de 1, et le nouveau tableau sera affiché à l'écran.

Corrigé 4 :

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

Source : <http://pise.info/algo/>

Énoncé 1 :

Écrivez une fonction qui renvoie la somme de cinq nombres fournis en argument.

Corrigé 1 :

Fonction Sum(a, b, c, d, e)
Renvoyer a + b + c + d + e
FinFonction

Énoncé 2 :

Écrivez une fonction qui renvoie le nombre de voyelles contenues dans une chaîne de caractères passée en argument. Au passage, notez qu'une fonction a tout à fait le droit d'appeler une autre fonction.

Corrigé 1 :

Fonction NbVoyelles(Mot en Caractère)
Variables i, nb **en Numérique**
 nb ← 0
Pour i ← 1 à Len(Mot)
 Si Trouve("aeiouy", Mid(Mot, i, 1)) <> 0 **Alors**
 nb ← nb + 1
FinSi
i suivant
Renvoyer nb
FinFonction

Énoncé 3 :

Réécrivez la fonction Trouve, vue précédemment, à l'aide des fonctions Mid et Len (comme quoi, Trouve,

à la différence de Mid et Len, n'est pas une fonction indispensable dans un langage).

Procédres-et-Fonctions(e01-e05)

Corrigé 3 :

Fonction Trouve(a, b)
Variable i **en Numérique**
Début
 i ← 1
TantQue i < Len(a) - Len(b) et b <> Mid(a, i, Len(b))
 i ← i + 1
FinTantQue
Si b <> Mid(a, i, Len(b)) **Alors**
 Renvoyer 0
Sinon
 Renvoyer i
FinFonction

Énoncé 4 :

Ecrire un traitement qui effectue le tri d'un tableau envoyé en argument (on considère que le code appelant devra également fournir le nombre d'éléments du tableau).

Corrigé 4 :

Fonction TriTableau(T, n)
Variables i, posmini, temp **en Numérique**
Début
Pour i ← 0 à n-2
 posmini ← i
 Pour j ← i + 1 à n-1
 Si t(j) < t(posmini) **Alors**
 posmini ← j
 Finsi
 j suivant
 temp ← T(posmini)

T(posmini) ← T(i)
 T(i) ← temp
i suivant
Renvoyer T
FinFonction

Énoncé 5 :

reprenre l'exercice 11.6, mais cette fois la procédure comprendra un troisième paramètre, de type booléen. VRAI, celui-ci indiquera que le tri devra être effectué dans l'ordre croissant, FAUX dans l'ordre décroissant.

Corrigé 5 :

Fonction TriTableau(T, n, Croissant)
Variables i, pos, temp **en Numérique**
Début
Pour i ← 0 à n-2
 pos ← i
 Pour j ← i + 1 à n-1
 Si Croissant **Alors**
 Si t(j) < t(pos) **Alors**
 pos ← j
 Finsi
 Sinon
 Si t(j) > t(pos) **Alors**
 pos ← j
 Finsi
 Finsi
 j suivant
 temp ← T(pos)
 T(pos) ← T(i)
 T(i) ← temp
i suivant
Renvoyer T
FinFonction

Énoncé 1 :

Écrire une fonction ou procédure qui permet d'avoir un nombre entier positif et afficher son image miroir.
Exemple le nombre est 3524, on doit afficher 4253.

Corrigé 1 :

Procédure Miroir (x : entier) ;
Variable a, b : entier ;
Debut
Tantque x \neq 0 Faire
Debut
a \leftarrow x Mod 10 ;
Ecrire (a) ;
x \leftarrow x Div 10 ;
Fin
Fintantque
Fin ;

Énoncé 2 :

Écrire une fonction ou procédure qui permet de calculer la multiplication de deux nombres A et B entiers en utilisant l'addition.

Corrigé 2 :

Fonction multiple (A, B : entier) : entier ;
Variable Res, Y : Entier ;
Debut
Res \leftarrow 0 ;
Si B < 0 Alors
Y \leftarrow -B
Sinon
Y \leftarrow B
FinSi
Tantque Y > 0 Faire

Procédres-et-Fonctions(e06-e10)

Debut
Res \leftarrow Res + A
Y \leftarrow Y - 1
Fin
FinTantque ;
Si B < 0 Alors
Res \leftarrow - Res
FinSi ;
multi

Énoncé 3 :

Écrire une fonction ou procédure qui affiche tous les nombres pairs compris entre deux valeurs entières positives lue x et y

Corrigé 3 :

Procédure calcul ;
Variable x, y, z : entier ;
Debut
Lire (x, y);
Si x > y Alors
Debut
z \leftarrow x ;
x \leftarrow y ;
y \leftarrow z ;
Fin
FinSi ;
Tantque x \leq y Faire
Si x mod 2 = 0 Alors
Ecrire (x)
FinSi ;
x \leftarrow x + 1
FinTantque
Fin ;

Énoncé 4 :

Écrire une fonction ou procédure qui affiche si un nombre est premier ou non

Corrigé 4 :

Procédure premier (a : entier) ;
Variable b : booleen ; d : entier ;
Debut
b \leftarrow vrai ;
d \leftarrow 2 ;
Tantque (d \leq a/2) et (b = vrai) Faire
Si a mod d = 0 Alors
b \leftarrow Faux
Sinon
d \leftarrow d + 1
FinSi ;
FinTantque
Si d = vrai Alors
Ecrire (a, 'est premier')
Sinon
Ecrire (a, 'n'est pas premier')
FinSi
Fin ;

Énoncé 5 :

Écrire une procédure qui affiche la nature d'un nombre entier (pair ou impair).

Corrigé 5 :

PROCEDURE A#cheN (E n : ENTIER)
SI nmod2 = 0 ALORS
ecrire('pair')
SINON
ecrire('impair')
FIN SI

Source : <http://pise.info/algo/>

Énoncé 1 :

Ecrivez une fonction qui purge une chaîne d'un caractère, la chaîne comme le caractère étant passés en argument. Si le caractère spécifié ne fait pas partie de la chaîne, celle-ci devra être retournée intacte. Par exemple :

- Purgé("Bonjour","o") renverra "Bnjur"
- Purgé("J'ai horreur des espaces"," ") renverra "J'aihorreurdesespaces"
- Purgé("Moi, je m'en fous", "y") renverra "Moi, je m'en fous"

Corrigé 1 :

Fonction PurgéSimple(a, b)
Variable Sortie en Caractère
Variable i en Numérique

Début
Sortie ← "
Pour i ← 1 à Len(a)
 Si Mid(a, i, 1) <> b **Alors**
 Sortie ← Sortie & Mid(a, i, 1)
 FinSi
i suivant
Renvoyer Sortie
FinFonction

Énoncé 2 :

Même question que précédemment, mais cette fois, on doit pouvoir fournir un nombre quelconque de caractères à supprimer en argument.

Procédres-et-Fonctions(e11-e15)

Corrigé 2 :

Fonction PurgéMultiple(a, b)
Variable Sortie en Caractère
Variable i en Numérique
Début
Sortie ← "
Pour i ← 1 à Len(a)
 Si Trouve(b, Mid(a, i, 1)) = 0 **Alors**
 Sortie ← Sortie & Mid(a, i, 1)
 FinSi
i suivant
Renvoyer Sortie
FinFonction

Énoncé 3 :

Écrire un traitement qui inverse le contenu de deux valeurs passées en argument.

Corrigé 3 :

Procédure Inversion(X en Numérique par référence, Y en Numérique par référence)
Variable Temp en Numérique
Début
Temp ← X
X ← Y
Y ← Temp
FinProcédure

Énoncé 4 :

Écrire un traitement qui informe si un tableau envoyé en argument est formé ou non d'éléments tous rangés en ordre croissant

Corrigé 4 :

Fonction TableauCroissant(T, n)
Variable i en Numérique
Variable Flag en Booléen
Début
Flag ← VRAI
i ← 0
TantQue Flag **et** i < n-1
 Flag ← T(i) < T(i+1)
 i ← i+1
FinTantQue
Renvoyer Flag
FinFonction

Écrire une fonction ou procédure qui permet d'entrer deux valeurs M et N et d'afficher toutes les valeurs paires entre M et N si M < N.

Procédure calcul ;
Declaration
Variable M, N : entier ;
Debut
Lire (M, N);
Si M ≥ N **Alors**
 Ecrire ('Pas d'affichage')
Sinon
 Tantque M < N **Faire**
 Si M mod 2 = 0 **Alors**
 Ecrire (M)
 FinSi ;
 M ← M + 1
 Fin
 FinTantque
 FinSi
Fin

Source : <http://pise.info/algo/>

Enoncé :

On va à présent réaliser une application complète, en utilisant une architecture sous forme de sous-procédures et de fonction. Cette application a pour tâche de générer des grilles de Sudoku. Une telle grille est formée de 81 cases (9 x 9), contenant un chiffre entre 1 et 9, et dans laquelle aucune ligne, aucune colonne et aucune "sous-grille" de 3x3, ne contient deux fois le même chiffre.

Pour parvenir à nos fins, on va utiliser une méthode particulièrement barbare et inefficace : la génération aléatoire des 81 valeurs de la grille. On vérifiera alors que la grille satisfait aux critères ; si tel n'est pas le cas... on recommence la génération jusqu'à ce que la grille convienne. En pratique, la probabilité de générer une grille adéquate est si faible que cette méthode prendra sans doute beaucoup de temps, mais passons. Tout le truc est de piger que vérifier que les neuf cases d'une ligne, d'une colonne, ou d'une sous-grille, sont toutes différentes, c'est en réalité du pareil au même. On va donc factoriser le code procédant à cette vérification sous la forme d'une fonction booléenne **TousDifférents**, à qui on passera un tableau de 9 valeurs en argument. La fonction renverra donc VRAI si les 9 valeurs du tableau sont toutes différentes, et FAUX sinon.

a. Ecrire la fonction **TousDifférents**
Maintenant, bien que ce ne soit pas indispensable (car ce code n'est pas spécialement répété), on choisit également par pure commodité de confier la génération au hasard de la grille de 81 cases à un module dédié, **RemplitGrille**. (ce module, à qui on passera notre tableau de 81 cases en argument, est forcément une procédure, puisqu'il a pour tâche d'en modifier les 81 valeurs).

b. Ecrire la procédure **RemplitGrille**

Procédres-et-Fonctions(e16-e20)

Il faut à présent vérifier que l'ensemble des lignes correspond à la condition voulue, à savoir qu'il n'y existe pas de doublons. On réalise donc une fonction, **VerifLignes**, qui va vérifier les neuf lignes de notre grille une par une (en utilisant bien sûr la fonction **TousDifférents**, déjà écrite) et renvoyer VRAI si toutes les lignes sont correctes, FAUX dans le cas contraire.

c. Ecrire la fonction **Veriflignes**

On procède alors de même avec une fonction chargée de vérifier les colonnes, **VérifColonnes**.

d. Ecrire la fonction **Verifcolonnes**

...et encore à nouveau, avec cette fois la vrification des neuf "sous-grilles" 3x3.

e. Ecrire la fonction **VerifSousGrilles**

Il ne reste plus qu'à écrire la procédure principale, et l'affaire est dans le sac !

f. Ecrire la procédure principale de l'application

a- Corrigé :

Fonction TriTableau(T, n, Croissant)

Variables i, pos, temp **en Numérique**

Début

Pour i ← 0 à n-2

pos ← i

Pour j ← i + 1 à n-1

Si Croissant **Alors**

Si t(j) < t(pos) **Alors**

pos ← j

Finsi

Sinon

Si t(j) > t(pos) **Alors**

pos ← j

Finsi

Finsi

j suivant

temp ← T(pos)

T(pos) ← T(i)

T(i) ← temp

i suivant

Renvoyer T

FinFonction

b- Corrigé :

Fonction TousDifférents(T(8) en num) en booléen

Pour i ← 0 à 7

Pour j ← i+1 à 8

Si T(i) = T(j) **Alors**

Renvoyer Faux

FinSi

j suivant

i suivant **FinFonction**

c- Corrigé :

Procédure RemplitGrille(T(8, 8) en num par référence)

Pour i ← 0 à 8

Pour j ← 0 à 8

T(i, j) ← Ent(Alea()*9)+1

j suivant

i suivant **FinProcédure**

d-Corrigé :

Fonction VerifLignes(Grille(8, 8) en num) en booléen

Tableau Ligne(8) **en Num**

Pour i ← 0 à 8

Pour j ← 0 à 8

Ligne(j) ← Grille(i, j)

j suivant

Si TousDifferent(Ligne()) = faux **Alors**

Renvoyer Faux

FinSi

i suivant

Renvoyer Vrai
FinFonction

e- Corrigé

Fonction VerifColonnes(Grille(8, 8) en num) en booléen

Tableau Colonne(8) en Num

Pour j \leftarrow 0 à 8

Pour i \leftarrow 0 à 8

 Ligne(i) \leftarrow Grille(i, j)

 i suivant

Si TousDifferentes(Ligne()) = faux **Alors**

Renvoyer Faux

FinSi

j suivant

Renvoyer Vrai

FinFonction

f- Corrigé :

Fonction VerifSousGrilles(Grille(8, 8) en num) en booléen

Tableau SousGrille(8) en Num

Pour ancrei \leftarrow 0 à 6 pas 3

Pour ancrej \leftarrow 0 à 6 pas 3

Pour decali \leftarrow 0 à 2

Pour decalj \leftarrow 0 à 2

 SousGrille(decali*3 + decalj) \leftarrow Grille(ancrei + decali, ancrej + decalj)

 decalj suivant

 decali suivant

Si TousDifferentes(SousGrille()) = faux **Alors**

Renvoyer Faux

FinSi

 ancrej suivant

ancrei suivant

Renvoyer Vrai
FinFonction

Procédure principale()

Tableau Grille(8, 8) en Num

Appeler RemplitGrille(Grille())

Tant Que Non VerifLignes(Grille()) ou Non

VerifColonnes(Grille()) ou Non

VerifSousGrilles(Grille())

Appeler RemplitGrille(Grille())

FinTantQue>

FinProcédure

CORRECTION EXERCICES ALGORITHME 1

Mr KHATORY
(GIM 1° A)

1

EXERCICES ALGORITHME

**Ecrire un algorithme permettant de résoudre une équation du second degré.
Afficher les solutions !**

$$ax^2 + bx + c = 0; \quad \text{solution: } x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Solution:

ALGORITHME seconddegré

VAR a, b, c, delta : REEL

DEBUT

ECRIRE (" saisissez les valeurs a, b et c de l'équation $ax^2+bx+c=0$: ")

LIRE (a, b, c)

SI (a=0)

ALORS

ECRIRE (" équation du premier degré ")

SI (b≠0)

ALORS **ECRIRE** ("solution est ", -c/b)

SINON **ECRIRE** (" Pas de solution")

FINSI

SINON

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

SI (delta > 0)

ALORS

ECRIRE ("les solutions sont ", $\frac{-b - \text{racine}(\text{delta})}{2 \cdot a}$ et ", $\frac{-b + \text{racine}(\text{delta})}{2 \cdot a}$)

SINON

SI delta =0 **ALORS** **ECRIRE** ("Solution est", -b/(2a))

SINON **ECRIRE** ("pas de solutions réelles !!")

FINSI

FINSI

FINSI

FIN

Fonction
standard

2

EXERCICES ALGORITHME

Ecrire le même algorithme avec des selon-que :

ALGORITHME seconddegré

VAR a, b, c, delta: REEL

DEBUT

 ECRIRE ("saisissez les valeurs de a, b et c de l'équation ax^2+bx+c ")

 LIRE (a, b, c)

SI (a=0)

ALORS

 ECRIRE ("équation du premier degré ")

SI (b<>0)

ALORS ECRIRE ("solution est ", -c/b)

SINON ECRIRE (" Pas de solution")

FINSI

SINON

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

SELONQUE

 delta = 0 : ECRIRE ("la solution unique est:", -b/(2a)

 delta > 0 : ECRIRE (" les deux solutions sont ", $\frac{-b - \text{racine}(\text{delta})}{2*a}$, " et " , $\frac{-b + \text{racine}(\text{delta})}{2*a}$)

SINON ECRIRE (" pas de solution réelle ")

FINSELON

FINSI

FIN

3

EXERCICES ALGORITHME

Ecrire un algorithme qui donne la durée de vol en heure minute connaissant l'heure de départ et l'heure d'arrivée.

On considère que le départ et l'arrivée ont lieu le même jour !

Données: h1,m1,h2 et m2

On suppose que $h2 > h1$!!

Cas possibles pour m1 et m2  2 cas (m1<m2 ou m1 >m2)

4

EXERCICES ALGORITHME

Ecrire un algorithme qui donne la durée de vol en heure minute connaissant l'heure de départ et l'heure d'arrivée.

On considère que le départ et l'arrivée ont lieu le même jour

Solution:**ALGORITHME DuréeVol**

VAR h1, h2, m1, m2: ENTIER
hd, md : ENTIER

DEBUT

```
    ECRIRE (" entrer horaire de départ: h min")
    LIRE (h1, m1)
    ECRIRE (" entrer horaire d'arrivée: h min")
    LIRE (h2, m2)
    SI (m2 > m1 )
    ALORS
        hd ⇐ h2-h1
        md ⇐ m2-m1
        ECRIRE (" la durée de vol est : ", hd , ': ', md)
    SINON
        hd ⇐ h2-h1-1
        md ⇐ m2+60-m1
        ECRIRE (" la durée de vol est : ", hd , ': ', md)
    FINSI
```

FIN

5

EXERCICES ALGORITHME

Ecrire un algorithme qui donne la durée de vol en heure minute connaissant l'heure de départ et l'heure d'arrivée.

On considère que le départ et l'arrivée ont lieu le même jour

Solution n° 2:**ALGORITHME DureeVol1**

VAR h1, h2, m1, m2: ENTIER
hd, md : ENTIER

DEBUT :


```
    ECRIRE (" entrer horaire de départ: h min")
    LIRE (h1, m1)
    ECRIRE (" entrer horaire d'arrivée: h min")
    LIRE (h2, m2)
    md ⇐ [h2*60+m2] - [h1*60+m1]
    hd ⇐ md div 60      (* division entière ( / )*)
    md ⇐ md mod 60      (*reste de la division entière (%)*)
    ECRIRE (" la durée de vol est : ", hd , ': ', md)
```

FIN

6

EXERCICES ALGORITHME

On suppose que la durée de vol est inférieure à 24 heures mais peut avoir lieu le lendemain.

Etudier les différents cas ! 

Données: $h1, m1, h2$ et $m2$

Exemple1:
Départ : 8h23 min
Arrivée: 13h 30 min

($*m1 < m2*$)

$h1 < h2$

Exemple2:
Départ : 8h23 min
Arrivée: 13h 15 min

($*m1 > m2*$)

Exemple3:
Départ : 17h30 min
Arrivée: 2h 40 min

($*m1 < m2*$)

$h1 > h2$

Exemple4:
Départ : 17h30 min
Arrivée: 2 h 25 min

($*m1 > m2*$)

➤ Comparer $h1$ et $h2$! (2 cas)

➤ Pour chaque cas: comparer $m1$ et $m2$! (2 cas)

 4 cas en tout !!

7

EXERCICES ALGORITHME

On suppose que la durée de vol est inférieure à 24 heures mais peut avoir lieu le lendemain.

ALGORITHME DureeVol2

VAR $h1, h2, m1, m2$: ENTIER

hd, md : ENTIER

DEBUT

ECRIRE (" entrer horaire de départ et d'arrivée: $h1 \ m1 \ h2 \ m2$ ")

LIRE ($h1, m1, h2, m2$)

SI ($h2 > h1$)

ALORS

SI ($m2 > m1$)

ALORS

hd \Leftarrow $h2 - h1$
md \Leftarrow $m2 - m1$
ECRIRE (hd, md)

SINON

hd \Leftarrow $h2 - h1 - 1$
md \Leftarrow $m2 + 60 - m1$
ECRIRE (hd, md)

FINSI

SINON

SI ($m2 > m1$)

ALORS

hd \Leftarrow $h2 - h1 + 24$
md \Leftarrow $m2 - m1$
ECRIRE (hd, md)

SINON

hd \Leftarrow $h2 - h1 + 24 - 1$
md \Leftarrow $m2 + 60 - m1$
ECRIRE (hd, md)

FINSI

FINSI

FIN

Exemple:
Départ : 8h23 min
Arrivée: 13h 30 min

Exemple:
Départ : 8h23min
Arrivée: 13h 15 min

Exemple:
Départ : 17h30min
Arrivée: 2h 40min

Exemple:
Départ : 17h30min
Arrivée: 2h 25 min

8

EXERCICES ALGORITHME

Ecrire un algorithme qui lit trois valeurs entières (A, B et C) et qui permet de les trier par échanges successifs Et enfin les afficher dans l'ordre

ALGORITHME TriSuccessif

VAR A, B, C : ENTIER

DEBUT

ECRIRE (" entrer Les valeurs A , B et C ")

LIRE(A,B,C)

SI (A > B) ALORS

échange (A,B)

SI B > C ALORS

échange (B,C)

SI A > B ALORS

échange (A,B)

FINSI

FINSI

SINON

SI B > C ALORS

échange (B,C)

SI A > B ALORS

échange (A,B)

FINSI

FINSI

FINSI

ECRIRE ("Les valeurs A , B et C sont d

Finalelement $A < B < C$

FIN

9

EXERCICES ALGORITHME

Ecrire un algorithme calculatrice permettant la saisie du premier entier (a) de l'opération (+ ou - ou * ou / : sont des caractères) et du deuxième entier (b) et qui affiche le résultat

ALGORITHME calculatrice

VAR a, b : ENTIER

op : CARACTERE

DEBUT

ECRIRE (" saisissez le premier entier ")

LIRE (a)

ECRIRE (" saisissez l'opération ")

LIRE (op)

ECRIRE (" saisissez le deuxième entier")

LIRE (b)

SELONQUE :

Op = '+' : ECRIRE ("la somme de ",a, "et de ",b, "est égale",a+b)

Op = '*' : ECRIRE ("le produit de ",a, "et de ",b, "est égale",a*b)

Op = '/' : SI (b= 0) ALORS ECRIRE (" division impossible ")

SINON ECRIRE ("la division de ",a, "par ",b, "est égale", a/b)

FINSI

Op = '-' : ECRIRE ("la soustraction de ",a, "et de ",b, "est égale", a-b)

SINON: ECRIRE(" Opération invalide ")

FINSELONQUE

FIN

10

EXERCICES ALGORITHME

1. Ecrire un algorithme qui demande un nombre de départ, et qui calcule la **somme** des entiers jusqu'à ce nombre. Par exemple si l'on tape 4, l'algorithme doit calculer: $1 + 2 + 3 + 4 = 10$

BOUCLE POUR**ALGORITHME Somme_Nombres**

```

VAR i, S : ENTIER
    val : ENTIER
DEBUT
    ECRIRE (" Entrer un nombre ")
    LIRE (val)
    S ← 0
    POUR i DE 1 A val FAIRE
        S ← S+i
    FINPOUR
    ECRIRE (" La somme des nombres de
    1 à ", val, "est ", S)
FIN

```

Equivalent
POUR**BOUCLE TANT QUE****Algorithme Somme_Nombres**

```

Var i, S : ENTIER
    Val : ENTIER
DEBUT
    ECRIRE (" Entrer un nombre entier:")
    LIRE(val)
    S ← 0
    i ← 1
    TANTQUE i ≤ val FAIRE
        S ← S+i
        i ← i+1
    FINTANTQUE
    ECRIRE (" La somme des nombres de 1 à ",
    val, "est ", S)
FIN

```

11

EXERCICES ALGORITHME

1. Ecrire un algorithme qui demande un nombre de départ, et qui calcule la **moyenne** des entiers jusqu'à ce nombre. Par exemple si l'on tape 4, l'algorithme doit calculer: $1 + 2 + 3 + 4 = 10/4 = 2.5$

BOUCLE POUR**ALGORITHME Moyenne_Nombres**

```

Var i, S : ENTIER
    Val : ENTIER
    Moyenne : REEL
DEBUT
    S ← 0
    LIRE (val)
    POUR i DE 1 A val FAIRE
        S ← S+i
    FINPOUR
    Moyenne ← S / val
    ECRIRE (" La moyenne des nombres de 1 à
    ", val, "est ", Moyenne)
FIN

```

Equivalent
POUR**BOUCLE TANT QUE****ALGORITHME Moyenne_Nombres**

```

Var i, S : ENTIER
    Val : ENTIER
    Moyenne : REEL
DEBUT
    S ← 0
    i ← 1
    Lire(val)
    TANTQUE i ≤ val FAIRE
        S ← S+i
        i ← i+1
    FINTANTQUE
    Moyenne ← S / val
    Ecrire (" La moyenne des nombres de
    1 à ", val, "est ", Moyenne)
FIN

```

12

EXERCICES ALGORITHME

Ecrire l'algorithme qui affiche la somme des prix d'une suite d'articles en DH (entiers) saisies par l'utilisateur et se terminant par zéro.

BOUCLE TANTQUE

```

ALGORITHME Somme_Prix
VAR p, S : ENTIER
DEBUT
S ← 0
ECRIRE("Entrer le prix du 1° article:")
LIRE(p)
TANTQUE (p≠0)
FAIRE
    S ← S+p
    ECRIRE("Entrer le prix de l'article suivant( 0 si Fin):")
    LIRE(p)
FINTANTQUE
ECRIRE (" La somme des prix des articles est ", S)
FIN

```

BOUCLE REPETER

```

ALGORITHME Somme_Prix
VAR p, S : ENTIER
DEBUT
S ← 0
ECRIRE("Entrer le prix du 1° article:")
LIRE(p)
REPETER
    S ← S+p
    ECRIRE("Entrer le prix de l'article suivant( 0 si Fin):")
    LIRE(p)
JUSQU'A (p =0)
ECRIRE (" La somme des prix des articles est ", S)
FIN

```

13

EXERCICES ALGORITHME

Ecrire l'algorithme qui affiche la somme des prix d'une suite d'articles en DH (entiers) saisies par l'utilisateur et se terminant par zéro.

Cas d'entrée à la boucle:

Si au départ $p=0$

→ choisir la boucle **TANTQUE**

Cas particulier ($p=0$) (Boucle **REPETER**) :
On peut changer l'algorithme :

```

ALGORITHME Somme_Prix
VAR p, S : ENTIER
DEBUT
S ← 0
REPETER
    ECRIRE("Entrer le prix de l'article ( 0 si Fin):")
    LIRE(p)
    S ← S+p
JUSQU'A p =0
ECRIRE (" La somme des prix des articles est ", S)
FIN

```



```

ALGORITHME Somme_Prix
VAR p, S : ENTIER
DEBUT
S ← 0
ECRIRE("Entrer le prix du 1° article:")
LIRE(p)
REPETER
    S ← S+p
    ECRIRE("Entrer le prix de l'article suivant( 0 si Fin):")
    LIRE(p)
JUSQU'A p =0
ECRIRE (" La somme des prix des articles est ", S)
FIN

```

14

EXERCICES ALGORITHME

Écrire un algorithme qui demande successivement 10 nombres à l'utilisateur, et qui affiche à la fin le plus grand de ces 10 nombres Et aussi son rang

Exemple :

Entrez le nombre numéro 1 : 13
 Entrez le nombre numéro 2 : 17

 Entrez le nombre numéro 10 : 5

 Le plus grand de ces nombres est : 17
 C'était le 2 ème nombre saisi

15

EXERCICES ALGORITHME

ALGORITHME Somme_10Nombres

```

CONST  NBRE=10;
VAR    indice , val : ENTIER
      Indice_grand, PLUSGRAND :ENTIER

DEBUT
  ECRIRE("Entrez le 1er nombre : ")
  LIRE (Val)
  PLUSGRAND ⇐ val
  Indice_grand ⇐ 1
  indice ⇐ 2
  TANTQUE (indice ≤ NBRE)
  FAIRE
    ECRIRE("Entrez le nombre numéro : ", indice)
    LIRE (Val)
    SI (val > PLUSGRAND) ALORS
      indice_grand ⇐ indice
      PLUSGRAND ⇐ val
    FINSI
    indice ⇐ indice+1
  FINTANTQUE
  ECRIRE("le plus grand de ces nombres est.", PLUSGRAND)
  ECRIRE(" c'était le ",indice_grand, " ème nombre saisi)
FIN
  
```

POUR indice DE 2 A NBRE
 Ecrire("Entrez le nombre numéro", indice)
 Lire (Val)
 Si val > PLUSGRAND alors
 Indice_grand ⇐ indice
 PLUSGRAND ⇐ val
 Finsi
 FINPOUR

16

EXERCICES ALGORITHME

Ecrire un programme mettant en œuvre le jeu suivant :

Le premier utilisateur saisi un entier que le second doit deviner. Pour cela, il a le droit à autant de tentatives qu'il souhaite. A chaque échec, le programme lui indique si l'entier cherché est plus grand ou plus petit que sa proposition.

Un score indiquant le nombre de coups joués est mis à jour et affiché lorsque l'entier est trouvé.

ALGORITHME devinette

VAR a, n, t : ENTIER

DEBUT

ECRIRE(" Entrez le nombre à deviner")

LIRE (a)

ECRIRE("entrez le nombre (premier essai")

LIRE (n)

t ← 1

TANTQUE (a ≠ n)

FAIRE

SI n > a **ALORS** ECRIRE (" nombre cherché plus petit que : ", n)

SINON ECRIRE (" nombre cherché plus grand que ", n)

FINSI

t ← t + 1

ECRIRE("entrez un autre nombre (tentative N° ", t, ")")

LIRE (n)

FINTANTQUE

ECRIRE (" c'est gagné : le nombre de tentatives est" ,t)

FIN

17

EXERCICES ALGORITHME

Ecrire un programme mettant en œuvre le jeu suivant :

Le premier utilisateur saisi un entier que le second doit deviner. Pour cela, il a le droit à autant de tentatives qu'il souhaite. A chaque échec, le programme lui indique si l'entier cherché est plus grand ou plus petit que sa proposition.

Un score indiquant le nombre de coups joués est mis à jour et affiché lorsque l'entier est trouvé.

ALGORITHME devinette

VAR a, n, t : ENTIER

DEBUT

ECRIRE(" Entrez le nombre à deviner")

LIRE (a)

t ← 1

REPETER

ECRIRE("Entrez un nombre (tentative N° : ", t, ")")

LIRE (n)

SELONQUE

n > a : ECRIRE (" nombre cherché plus petit que : ", n)

n < a : ECRIRE (" nombre cherché plus grand que ", n)

n = a : ECRIRE (" c'est gagné : le nombre de tentatives est" ,t)

FINSELONQUE

t ← t + 1

JUSQU'A (a=n)**FIN****BOUCLE REPETER**

18

EXERCICES ALGORITHME

Écrire une fonction **F_PGCD** qui retourne le PGCD de deux nombres en utilisant l'astuce suivante: soustrait le plus petit des deux entiers du plus grand jusqu'à ce qu'ils soient égaux.

Exemple:

a=24

b=36

Le PGCD ??

Boucle :

1. **a** < b (24 < 36) → **b** = 36 - 24 = 12

2. **b** < a (12 < 24) → **a** = 24 - 12 = 12

a=b=12 on s'arrête donc le PGCD est **12**

19

EXERCICES ALGORITHME

Écrire une fonction **F_PGCD** qui retourne le PGCD de deux nombres en utilisant l'astuce suivante: soustrait le plus petit des deux entiers du plus grand jusqu'à ce qu'ils soient égaux.

FONCTION F_PGCD(a,b :ENTIER) :ENTIER

DEBUT

REPETER

SELONQUE

a > b : a ← a - b

a < b : b ← b - a

FINSELON

JUSQU'À a = b

retourner(a)

FIN

20

EXERCICES ALGORITHME

Ecrire la même fonction en utilisant l'algorithme d'Euclide: Utiliser les structures TANTQUE puis REPETER JUSQU'A

Boucle TANT QUE :

```

FUNCTION F_PGCD(a,b :ENTIER) :ENTIER
VAR r :ENTIER
DEBUT
r ← a%b /* reste de la division entière*/
TANTQUE r≠0
FAIRE
    a ← b
    b ← r
    r ← a%b
FINTANTQUE
retourner(b)
FIN

```

Boucle REPETER JUSQU'A :

```

FUNCTION F_PGCD(a,b :ENTIER) :ENTIER
VAR r :ENTIER
DEBUT
REPETER
    r ← a%b
    a ← b
    b ← r
JUSQU'A r=0
retourner(b)
FIN

```

21

Merci
&
FIN

22