Résumé général pour préparer votre BAC

Matière : Algorithmique et Programmation

Section : Sciences de l'Informatique

Réalisé par :

Ghanmi Alaa eddine

(eleve 4eme SI)

Ghanmi Alaa Eddine

Les enregistrements

I. Déclaration (sur un exemple)

- **En analyse :**

TDNT

Type
ELEVE = **Enregistrement** Nom : Chaine [30] Prenom : Chaine [20] Sexe : caractère Moy : Réel Fin **ELEVE**

TDO

Objet	*Type/Nature*
E1	ELEVE

- **En Pascal :**

```
TYPE
ELEVE = Record
            Nom : String [30] ; Prenom :
            String [20] ; Sexe : Char ; Moy :
            Real ;
        End ;
VAR
  E1 : ELEVE ;
```

Remarques:

- Les types des champs peuvent être soit prédéfinis (Entier, réel, caractère, chaine, ...) soit définis par l'utilisateur.
- Le type enregistrement fait partie des types **structurés** (Avec Les types Tableau, matrice et fichier) **Un enregistrement ne doit jamais être le résultat d'une fonction.**

III. Utilisation des enregistrements (Sur des exemples)

III.1 Accès à un champ d'un enregistrement

L'accès à un champ d'une variable enregistrement se fait comme suit :

E1. Nom (En analyse, algorithme et Pascal)

III.2 Affectation

En analyse & algorithme	En Pascal
E1. Moy ← 15.23	E1.Moy := 15.23 ;

Ghomri Alaa Eddine

Remarques:

1. Il est possible d'affecter une variable enregistrement dans une autre à condition qu'ils aient la même structure.
   Exemple: **VAR** E1, E2: ELEVE
   Il est possible d'écrire E2 ← E1 (ou bien E1 ← E2)
2. Un champ a exactement les mêmes propriétés qu'une variable de même type.

III.3 Lecture

En Analyse	En algorithmique	En Pascal
E1.Nom = Donnée	Lire (E1.Nom)	Readln (E1.Nom) ;

III.4 Ecriture

En Analyse et en algorithmique	En Pascal
Ecrire (E1.Prenom)	Write (E1.Prenom) (ou **Writeln**)

III.5 Structure Avec … Faire

En Analyse et en algorithmique	En Pascal

Avec E1 **Faire**
Ecrire (Nom," ", Prenom," ",Sexe, " ", Moy)
Fin Avec
	With E1 **Do**
	Begin
	Writeln (Nom,' ', Prenom,' ',Sexe,' ',Moy);
	End;

III.6 Les vecteurs d'enregistrements

- **En analyse :**

TDNT

Type
Tab **=** **Tableau** de 30 **ELEVE**

TDO

Objet	*Type/Nature*
T	Tab

- **En Pascal :**
  TYPE Tab **=** **Array** [1..30] of **ELEVE;**
  VAR T: Tab;

Ghani Ass Eddine

Les Fichiers

I. Caractéristiques

Chaque fichier est désigné par deux noms : nom logique et nom physique

II. Types d'accès aux éléments

L'accès à un élément (une donnée) d'un fichier peut se faire de 2 manières :

- ✓ **Accès séquentiel**
- ✓ **Accès direct**

III. Types de fichiers :

Il existe 2 types de fichiers qui diffèrent selon l'organisation, le type des données et le mode d'accès :

- ✓ **Fichier typé**
- ✓ **Fichier texte**

NB: Un fichier typé peut être accessible d'une manière **séquentielle** ou **direct** tandis qu'un fichier texte n'est accessible que d'une manière **séquentielle**.

VI. Les fichiers de données (typés) :

VI.1 Déclaration : (Sur un exemple)

- **En analyse :**

Tableau de déclaration des nouveaux types (T.D.N.T)

Type
FICH_NOTES = **Fichier** de réel

Tableau de déclaration des objets (T.D.O)

Objet	*Type/Nature*
F	FICH_NOTES

- **Au niveau du Pascal:**

TYPE FICH_NOTES = **File** of **Real**;
VAR F: FICH_NOTES;

NB: Au contraire du type tableau, la taille d'un fichier **n'est pas fixée à l'avance**

VI.3 Traitement sur les fichiers : (Sur des exemples)

VI.3.1 Association : (Procédure)

❖ **Syntaxe :**

En Analyse et en algorithmique	En Pascal
Associer (F, "C:\4SI\Note_el.dat")	**ASSIGN** (F, 'C:\4SI\Note_el.dat') ;

VI.3.2 Ouverture :

1- Ouverture et création :

Ghanmi Alaa Eddine

En Analyse et en algorithmique	En Pascal
Recréer (F)	**Rewrite** (F);

2- Ouverture et emplacement du pointeur au début :

En Analyse et en algorithmique	En Pascal
Ouvrir (F)	**Reset** (F);

Etat du fichier	*Recréer*	*Ouvrir*
Existe	• Remplacer le fichier par un autre vide • Ouvrir le fichier	• Positionner le pointeur au début du fichier • Ouvrir le fichier
N'existe pas	• Créer un fichier vide • Ouvrir le fichier	Afficher un message d'erreur

VI.3.3 Ecriture dans un fichier :

En Analyse et en algorithmique	En Pascal
X ← 12.25 **Ecrire** (F, X)	**X:=12.25;** **Write** (F, X);

❖ **Remarques:**

- Avant d'utiliser l'instruction Ecrire il faut toujours affecter une valeur à la Variable
- La variable doit être de **même type** que le fichier.
- A chaque écriture d'une valeur ou d'un enregistrement, le pointeur du fichier avance automatiquement d'une position.

NB: on doit toujours mettre le mot clé VAR avant le nom du fichier dans les paramètres formels

VI.3.4 Lecture à partir d'un fichier :

En Analyse et en algorithmique	En Pascal
Lire (F, Y)	**Read** (F, Y);

→ Cette instruction permet de mettre le contenu du bloc courant de F dans une variable Y.

VI.3.5 Fermeture d'un fichier :

En Analyse et en algorithmique	En Pascal
Fermer (F)	**Close** (F);

Ghomri Aïda Edine

NB: *À la fin du traitement, on doit* **toujours** *fermer le fichier.*

VI.3.6 Test de fin de fichier : (Fonction Fin_fichier)

En Analyse et en algorithmique	En Pascal
Fin_fichier (F)	**EOF** (F)

Cette **fonction** permet de tester si on a atteint la fin du fichier ou non. Elle retourne un booléen

Application :

Soit **F1** un fichier qui contient les moyenne des élèves d'une classe. On veut afficher toutes les moyennes **supérieures ou égales à 12**. Proposer une solution au problème.

Solution :

```
DEF PROC AFFICHAGE (var F: FICH_NOTES)
Résultat = AFF
AFF = [] Tant que (Non (Fin_fichier (F)) Faire
   Lire (F1, Moy)
   Si Moy >= 12 Alors
                        Ecrire (Moy)
Fin Si
 Fin Tant que
      Fin AFFICHAGE
```

VI.4 Autre fonctions et procédures prédéfinies sur les fichiers :

En Analyse et en algorithmique	En Pascal	Rôle	Fonction / Procédure
Position_fichier (Nom_logique)	**FilePos** (Nom_logique);	Permet de déterminer la position actuelle du pointeur dans un fichier. Elle vaut 0 lorsque le pointeur se pointe sur le 1er bloc	**Fonction**
Effacer (Nom_logique)	**Erase** (Nom_logique);	Permet d'effacer un fichier à partir de son chemin d'accès.	**Procédure**
Renommer (Nom_logique, Nouveau_nom_physique)	**Rename** (Nom_logique, Nouveau_nom_physique);	Permet de renommer un fichier en lui donnant un nouveau nom physique. **NB:** Le fichier doit être **fermé** avant de le renommer	**Procédure**
Tronquer (Nom_logique)	**Truncate** (Nom_logique);	Permet de supprimer tout ce qui se trouve après la position courante du pointeur	**Procédure**

Grami Aïda Edine

VI.5 L'accès direct à un fichier :

VII. Les fichiers textes :

VII.1 Déclaration : (Sur un exemple)

En Analyse et en algorithmique	En Pascal	Rôle	Fonction / Procédure
Pointer (Nom_logique, Numéro)	**Seek** (Nom_logique, Numéro);	Permet de positionner le pointeur sur le bloc voulu. **NB :** Numéro désigne le numéro d'ordre du bloc auquel on veut accéder. **Il vaut 0 pour le 1er bloc**	**Procédure**
Taille_fichier (Nom_logique)	**FileSize** (Nom_logique);	Permet de savoir le nombre de blocs qui se trouvent dans un fichier	**Fonction**

- **En analyse :**

T.D.O

Objet	*Type/Nature*
FT	Texte

- **En Pascal :**

VAR
FT : **Text** ;

VII.2 Fonctions et procédures prédéfinies :

En analyse & Algorithme	En Pascal	Rôle
Fin_ligne (Nom_logique)	EoLn (Nom_logique) ;	**Fonction** qui retourne **Vrai** si le pointeur est en fin de ligne, **faux** sinon.
Chercher_fin_ligne (Nom_logique)	SeekEoLn (Nom_logique) ;	Identique à **EoLn** mais en supprimant les espaces et les caractères de tabulation avant d'effectuer le test.
Fin_fichier (Nom_logique)	Eof (Nom_logique) ;	**Fonction** qui retourne **Vrai** si le pointeur est en fin de fichier, **faux** sinon.
Chercher_fin_fichier (Nom_logique)	SeekEof (Nom_logique) ;	Identique à **Eof** mais en supprimant les espaces et les caractères de tabulation avant d'effectuer le test.
Ajouter (Nom_logique)	Append (Nom_logique) ;	**Procédure** qui ouvre le fichier et positionne son pointeur à la fin de ce dernier. Seul l'ajout d'éléments est alors possible.
Lire_nl (Nom_logique, Chaîne)	Readln (Nom_logique, chaîne) ;	**Procédure** qui met le contenu de la ligne courante du fichier dans la chaîne.
Ecrire_nl (Nom_logique, chaîne)	Writeln (Nom_logique, chaîne) ;	**Procédure** qui met le contenu de la chaîne dans ma ligne courante du fichier

Ghomri Aïda Edine

NB:

- Toutes les procédures et fonctions vues pour les fichiers typés sont valables pour les fichiers textes sauf : **Seek**, **FilePos**, **FileSize** et **Truncate**.
- Pour un fichier texte, la Procédure **Rewrite** permet l'ouverture en écriture et la procédure **Reset** permet l'ouverture en lecture.

ASUTUCE : (Remplissage d'un fichier qui s'arrête par la réponse "N" à la question "Voulez vous continuez (O/N)")

```
DEF PROC REMPLISSAGE (Var F: FICH)
Résultat = F
F = [Ouvrir(F)] Répéter
                    Avec E faire
                      Nom, Prenom, Sexe, Moy = Donnée
                    Fin Avec
                    Ecrire (F, E) {Ne pas oublier ça !!!!}
                    Répéter
                      Rep = Donnée ("Voulez vous continuez (O/N)")
                    Jusqu'à (Majus (Rep) Dans ["O","N"])
                  Jusqu'à (Majus (Rep) = "N")
Fermer (F)
Fin REMPLISSAGE
```

Ghanmi Alaa Eddine

La récursivité

I. Définition

La **récursivité** est une méthode algorithmique qui consiste à appeler un sous programme dans son propre corps.

II. Ecriture d'un module récursif :

Exemple 1 :

```
0)  DEF FN Factorielle (N: entier): entier long
1)  Si N=1 Alors Factorielle ← 1
Sinon Factorielle ← N * Factorielle (N – 1)
 Fin Si
2)  Fin Factorielle
```

Cas particulier

Cas général

Notez bien que N diminue de 1

Exemple 2: (Chaîne palindrome)

Prenons le mot "LAVAL"

On sait que chaque chaîne de longueur 0 ou 1 est palindrome

- **1ère étape:** Long ("LAVAL") = 5 ≥ 2 donc on compare le 1er et le dernier caractère de la chaîne. Et puisque **"L" = "L"** alors **Palindrome ("LAVAL") = Palindrome ("AVA")**
- **2ème étape:** Long ("AVA") = 3 ≥ 2 donc on compare le 1er et le dernier caractère et puisqu'ils sont égaux (**"A" = "A"**) alors **Palindrome ("AVA") = Palindrome ("V")**
- **3ème étape:** Long ("A") = 1 < 2 donc la chaîne **"V"** est palindrome et donc **on s'arrête**.

```
0)  DEF FN Palindrome (CH: chaîne): Booléen
1)  Si long (CH) < 2 Alors
        Palindrome ← Vrai
    Sinon Si CH [1] ≠ CH [long (CH)] alors
        Palindrome ← Faux

    Sinon Palindrome ← Palindrome (Sous-chaine (CH, 2, Long (CH) – 2)
    Fin Si
2)  Fin Palindrome
```

Cas particuliers

Notez bien que CH diminue de 2 caractères à chaque fois

Cas général

Ghomi Abd Edine

Les algorithmes de Tri

Tri par sélection :

```
0) DEF FN Posmin (T: Tab; N, Binf:Entier): Entier
1) P ← Binf
   Pour i de Binf+1 à N Faire
     Si T[i] < T[P] Alors
        P ← i
    Fin si
   Fin pour
2) Posmin ← P
3) Fin Posmin

0) DEF PROC Tri_selection (var T:tab; N: Entier)
1) Pour i de 1 à N-1 Faire
     M ← FN Posmin (T,N,i)
     Si T[i] ≠ T[M] Alors
       PROC Permut (T[M],T[i])
     Fin Si
   Fin pour
2) Fin Tri_selection
```

```
0) DEF PROC Permut (Var X,Y: Entier)
1) Aux ← X
2) X ← Y
3) Y ← Aux
4) Fin Permut
```

NB : Dans la procédure Permut, X et Y sont toujours de même type que les éléments du tableau

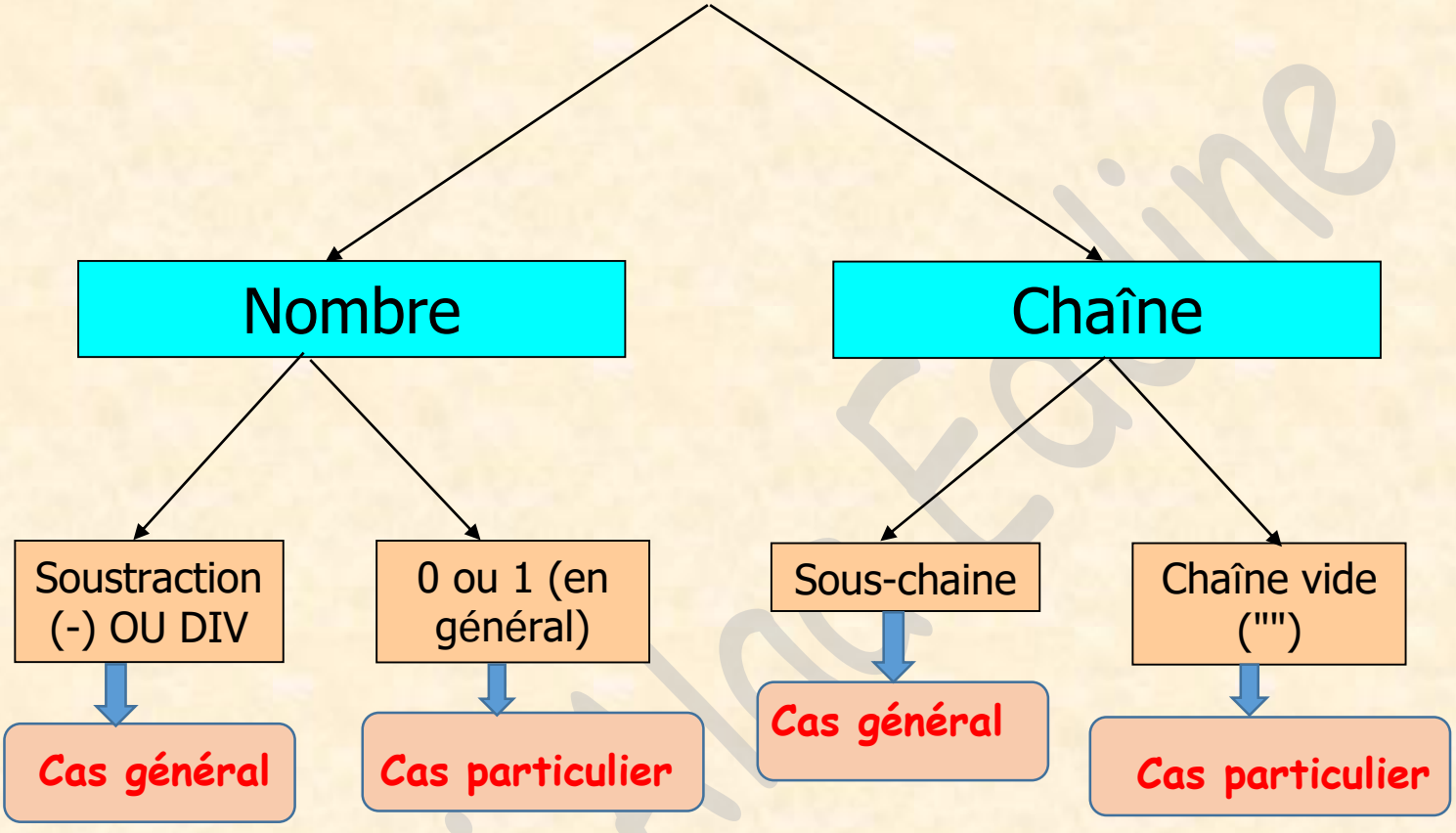
Tri à bulles :

```
0) DEF PROC Tri_bulles (var T: Tab ; N: Entier)
1) Répéter
   Echange ← Faux
   Pour i de 1 à N-1 Faire
   Si T[i] > T[i+1] Alors
      PROC Permut (T[i],T[i+1])
      Echange ← Vrai
   Fin Si
   Jusqu'à (Echange = Faux)
2) Fin Tri_bulles
```

Tri par insertion :

```
0) DEF PROC Tri_insertion (var T:tab; N: Entier)
1) Pour i de 2 à N Faire
     Aux ← T[i]
     j ← i
     Tant que (T[j-1]>Aux) ET (j>1) Faire
        T[j] ← T[j-1]
        j ← j-1
     Fin Tant que
     T[j] ← Aux
   Fin Pour
2) Fin Tri_insertion
```

Quelques trucs



Astuce

Une fonction booléenne comporte au moins 2 cas particuliers :

- **cas pour lequel la fonction retourne VRAI**
- **cas pour lequel la fonction retourne FAUX**

Ghanmi Abdelaziz

Tri Shell :

```
0) DEF PROC Tri_Shell (Var T:Tab ; N: Entier)
1) p← 0
     Tant que (p<N) Faire
       p← 3*p +1
     Fin tant que
2)  Tant que (p≠1) Faire
       p← p DIV 3
       Pour i de p+1 à N Faire
        Aux ← T[i]
        j ← i
        Tant que (j > p) ET (T[j - p] > Aux) Faire
          T [j] ← T [j – p]
          j ← j – p
        Fin Tant que
        T [j] ← Aux
      Fin Pour
     Fin Tant que
3)  Fin Tri_Shell
```

Tri décroissant

Les changement par rapport au tri croissant sont les suivants :

Méthode de tri	Ordre croissant	Ordre décroissant
Tri par sélection	T[i] **<** T[p]	T[i] **>** T[p]
	Fonction **Posmin**	Fonction **Posmax**
Tri à bulles	T[i] **>** T[i+1]	T[i] **<** T[i+1]
Tri par insertion	T[j-1] **>** Aux	T[j-1] **<** Aux
Tri Shell	T[j-p] **>** Aux	T[j-p] **<** Aux

Tri d'une chaîne

Les changement par rapport au tri d'un tableau sont les suivants :

Tableau	Chaîne
T	CH
N	Long (CH)
Paramètres (T,N)	Paramètre CH seulement

Trid'untableaud'enregistrements

Les changement par rapport à un tableau d'éléments simples sont les suivants : (Supposons que le tri se fait par rapport à un champ appelé **Moy** par exemple)

Méthode de tri	Tableau d'éléments simples	Tableau d'enregistrements
Tri par sélection	T[i] **<** T[p]	T[i]**.Moy** **<** T[p]**.Moy**
Tri à bulles	T[i] **>** T[i+1]	T[i]**.Moy** **>** T[i+1]**.Moy**
Tri par insertion	T[j-1] **>** Aux	T[j-1]**.Moy** **>** Aux**.Moy**
Tri Shell	T[j-p] **>** Aux	T[j-p] **.Moy** **>** Aux**.Moy**

Tri d'une partie d'un tableau

Les changement par rapport au tri d'un tableau complet sont les suivants : (Avec **Deb** = Début de la partie à trier et **Fin** = la fin de la partie à trier)

Tableau complet	Partie d'un tableau
1	**Deb**
N	**Fin**
Paramètres (T,N)	Paramètres (T,**Deb**,**Fin**)

Tri selon un critère

Supposons qu'on va trier le tableau selon le **nombre de chiffres pairs** dans chacune de ses cases

Les changement seront les suivants :

Méthode de tri	Tri habituel	Tri selon un critère
Tri par sélection	T[i] **<** T[p]	**FN Chiff_pairs** (T[i]) **<** **FN Chiff_pairs** (T[p])
Tri à bulles	T[i] **>** T[i+1]	**FN Chiff_pairs** (T[i]) **>** **FN Chiff_pairs** (T[i+1])
Tri par insertion	T[j-1] **>** Aux	**FN Chiff_pairs** (T[j-1]) **>** **FN Chiff_pairs** (Aux)
Tri Shell	T[j-p] **>** Aux	**FN Chiff_pairs** (T[j-p]) **>** **FN Chiff_pairs** (Aux)

Tri d'un fichier

Le tri d'un fichier se fait en 3 étapes :
1- Transfert du fichier vers un tableau
2- Tri du tableau
3- Transfert du tableau vers le fichier

Procédure transfert_Fich_Tab :

1. Nombre de bloc (ou lignes) dans le fichier connu :

```
DEF PROC TRANSFERT_FICH_TAB (Var F: Fich ; Var T: Tab; N: Entier)
Résultat = T
T, N = [Ouvrir (F)] Pour i de 1 à N Faire
                        Lire (F, T[i])
                    Fin Pour
Fin TRANSFERT_FICH_TAB
```

2. Nombre de bloc (ou lignes) dans le fichier inconnu :

```
DEF PROC TRANSFERT_FICH_TAB (Var F: Fich ; Var T: Tab; Var N: Entier)
Résultat = T, N
T, N = [Ouvrir (F), N ← 0] Tant que NON Fin_Fichier (F) Faire
                                N ← N + 1
                                Lire (F, T[N])
                            Fin Tant que
Fin TRANSFERT_FICH_TAB
```

Procédure transfert_Tab_Fich :

```
DEF PROC TRANSFERT_TAB_FICH (Var F: Fich ; T: Tab; N: Entier)
Résultat = F
F = [Récréer (F)] Pour i de 1 à N Faire
                        Ecrire (F, T[i])
                    Fin Pour
Fin TRANSFERT_TAB_FICH
```

Ghomri Abd Edine

Les algorithmes récurrents

1- Déterminer le $N^{ième}$ terme d'une suite d'ordre 1

Soit la suite U définie par :

$$\begin{cases} U_0 = 2 \\ U_n = 5 * U_{n-1} - 3 \qquad (\text{pour } n \geq 1) \end{cases}$$

Suite d'ordre 1 car pour calculer un terme de la suite U on a besoin d'un seul terme précédent.

```
DEF FN TERME_N (N: entier) : entier
Résultat = TERME_N
TERME_N ← U
T = [U1 ← 2] Pour i de 2 à N Faire
                    U ← 5 * U1 - 3
                    U1 ← U
               Fin Pour
Fin TERME_N
```

2- Déterminer le $N^{ième}$ terme d'une suite d'ordre 2

Soit la suite U définie par :

$$\begin{cases} U_0 = 3/2 \\ U_1 = 5 \\ U_n = 2 * U_{n-1} - 1/3 * U_{n-2} \qquad (\text{pour } n \geq 2) \end{cases}$$

Suite d'ordre 2 car pour calculer un terme de la suite U on a besoin de 2 termes précédents.

```
DEF FN TERME_N (N: entier) : Réel
Résultat = TERME_N
TERME_N ← U
U =  U1 ← 3/2      Pour i de 3 à N Faire
     U2 ← 5          U ← 2 * U2 - 1/3 * U1
                     U1 ← U2
                     U2 ← U
                   Fin Pour
Fin TERME_N
```

3-Suite de Fibonnacci

Soit F la suite de Fibonnacci définie par :

$$\begin{cases} F_1 = 1 \\ F_2 = 1 \\ F_n = F_{n-1} + F_{n-2} \qquad (\text{pour } n \geq 3) \end{cases}$$

Ghorri Alaa Eddine

Suite d'ordre 2 car pour calculer un terme de la suite de Fibonnacci on a besoin de 2 termes précédents.

```
DEF FN FIBO (N: entier) : Entier Résultat = FIBO
  FIBO ← F
  F =  ⎛F1 ← 1⎞      Pour i de 3 à N Faire
       ⎝F2 ← 1⎠          F ← F2 + F1
                         F1 ← F2
                         F2 ← F
                      Fin Pour
  Fin FIBO
```

4- Déterminer et afficher les N premiers termes d'une suite U

Soit la suite U définie par :

$$\begin{cases} U_0 = 3/2 \\ U_1 = 5 \\ U_n = 2 * U_{n-1} - 1/3 * U_{n-2} \quad \text{(pour } n \geq 2\text{)} \end{cases}$$

```
DEF PROC AFFICHE_TERME (N: entier)
Résultat = AFF
AFF =  ⎛U1 ← 3/2⎞          Pour i de 3 à N Faire
       ⎜U2 ← 5  ⎟              U ← 2 * U2 - 1/3 * U1
       ⎝Ecrire (U1, U2)⎠       U1 ← U2
                               U2 ← U
                               Ecrire (U)
                           Fin Pour
Fin AFFICHE_TERME
```

5- Déterminer si x est un terme d'une suite U ou non

Soit la suite U définie par :

$$\begin{cases} U_0 = 2 \\ U_1 = 5 \\ U_n = U_{n-1} + 3 * U_{n-2} \quad \text{(pour } n \geq 2\text{)} \end{cases}$$

Question : Ecrire une analyse d'un module **itératif** qui permet de déterminer si un entier **X > 5** est un terme de la suite U ou non (Sachant que U est une suite croissante).

Solution itérative :

```
DEF FN Chercher_X (X : Entier) : Booléen
Résultat = Chercher_X
Chercher_X ← (U = X)
U =  ⎛U1 ← 2⎞      Répéter
     ⎝U2 ← 5⎠        U ← U2 + 3 * U1
                     U1 ← U2
                     U2 ← U
                   Jusqu'à (U >= X)
Fin Chercher_X
```

Ghomri Aïa Eddine

Solution récursive :

```
DEF FN Chercher_X (U1, U2, X : Entier) : Booléen
Résultat = Chercher_X

Chercher_X = [ U ← U2 + 3 * U1 ]   Si (U = X) Alors
                                      Chercher_X ← Vrai
                                   Sinon Si (U > X) Alors
                                      Chercher_X ← Faux
                                   Sinon
                                      Chercher_X ← FN Chercher_X (U2, U, X)
                                   Fin si
Fin Chercher_X
```

Soit la suite U définie par :

$$\begin{cases} U_0 = 2 \\ U_1 = 5 \\ U_n = U_{n-1} + 3 * U_{n-2} \quad \text{(pour } n \geq 2\text{)} \end{cases}$$

Question : Ecrire une analyse d'un module **itératif** qui permet de déterminer si un entier X est un terme de la suite U ou non (Sachant que U est une suite croissante).

Solution itérative : (X peut être égale au premier ou au 2e terme)

```
DEF FN Chercher_X (X : Entier) : Booléen
Résultat = Chercher_X

Chercher_X = [ U1 ← 2 ]   Si (X = 2) Ou (X = 5) Alors
             [ U2 ← 5 ]      Chercher_X ← Vrai
                          Sinon
                             Répéter
                                U ← U2 + 3 * U1
                                U1 ← U2
                                U2 ← U
                             Jusqu'à (U >= X)
                          Chercher_X ← (U = X)
                       Fin si
Fin Chercher_X
```

6- Déterminer le N^{ième} terme d'une suite U d'une manière récursive

Soit la suite U définie par :

$$\begin{cases} U_0 = 2 \\ U_1 = 5 \\ U_n = U_{n-1} + 3 * U_{n-2} \quad \text{(pour } n \geq 2\text{)} \end{cases}$$

Question : Ecrire une analyse d'un module **récursif** qui permet de déterminer le N^{ième} terme de la suite U.

Ghanmi Abd Eddine

```
DEF FN TERME_N (N: entier) : entier
Résultat = TERME_N
TERME_N = [ ] Si (N=1) Alors
                TERME_N ← 2
              Sinon Si (N=2) Alors
                TERME_N ← 5
              Sinon
                TERME_N ← FN TERME_N (N-1) + 3 * FN TERME_N (N-2)
             Fin si
Fin TERME_N
```

7- Suites sur les chaînes - Suite de Thue Morse

Soit la suite de chaînes de caractères (appelée suite de Thue Morse) définie par :

U_0 = "0" { On choisir aussi U_0 = "1"}

U_n = La chaîne formée en remplaçant dans U_{n-1} toute occurrence de "0" par "01" et toute occurrence de "1" par "10"

Exemple :

U_0 = "0"
U_1 = "01"
U_2 = "0110"
U_3 = "01101001"

U_0 = "1"
U_1 = "10"
U_2 = "1001"
U_3 = "10010110"

Question : Ecrire une analyse d'un module **itératif** qui permet de déterminer le $N^{ième}$ terme de la suite U.

```
DEF FN Thue_Morse (N: entier) : Chaine
Résultat = Thue_Morse
Thue_Morse ← CH
CH = [CH ← "0" ] Pour i de 2 à N Faire
                    CH2 ← " " {Chaine vide}
                    Pour j de 1 à long (CH) Faire
                       Si CH[j] = "0" Alors
                          CH2 ← CH2 + "01"
                         Sinon CH2 ← CH2 + "10"
                       Fin si
                    Fin Pour
                    CH ← CH2
                 Fin Pour
Fin Thue_Morse
```

Ghorri Alaa Eddine

8- Les matrices (parcours - gestion des compteurs)

M	1	2	3	4	5	6	7
1	-5	4	6	2	3	8	12
2	7	0	-6	11	32	65	5
3	-8	8	-7	15	13	-1	20
4	-6	30	3	-3	16	15	2

Parcours de la 2ème ligne:

```
Pour j de 1 à NC Faire   {NC : nombre de colonnes (ici = 7) }
    S ← S + M [2, j]
Fin Pour
```

Parcours de la 3ème colonne:

```
Pour i de 1 à NL Faire   {NL : nombre de lignes (ici = 4) }
    P ← P * M [i, 3]
Fin Pour
```

M	1	2	3	4	5
1	-5	4	6	2	3
2	7	0	-6	11	32
3	-8	8	-7	15	13
4	-6	30	3	-3	16
5	5	12	6	1	-3

Parcours de la 1ère diagonale:

```
Pour i de 1 à N Faire   {N : nombre de lignes (ici = 5) }
    S ← S + M [i, i]
Fin Pour
```

Parcours de la 2ème diagonale :

```
Pour i de 1 à N Faire   {N : nombre de lignes (ici = 5) }
    P ← P * M [i, N-i+1]
Fin Pour
```

Ghamri Alaa Eddine

9- Triangle de pascal :

M	1	2	3	4	5
1	1				
2	1	1			
3	1	2	1		
4	1	3	3	1	
5	1	4	6	4	1

```
DEF PROC Triangle_Pascal (Var M : Mat ; N : Entier)
Résultat = M
M = [] Pour i de 1 à N Faire
        Pour j de 1 à i Faire
          Si (j = 1) ou (j = i) Alors
            M [i, j] ← 1
          Sinon
            M [i, j] ← M [i - 1, j] + M [i - 1, j - 1]
          Fin Si
        Fin Pour
      Fin Pour
Fin Triangle_Pascal
```

Ghanmi Alaa Eddine

Les algorithmes d'arithmétique

1. Calcul du PGCD : (Itératif)

1ère méthode:

```
0) DEF FN PGCD (A, B : Entier) : Entier
1) Tant que (A <> B) Faire
      Si (A > B) Alors
      A ← A – B
      Sinon
      B ← B – A
      FinSi
   Fin Tant que
2) PGCD ← A
3) Fin PGCD
```

2ème méthode:

```
0) DEF FN PGCD (A, B : entier) : entier
1) Tant que (B <> 0) Faire
      R ← A MOD B
      A ← B
      B ← R
   Fin Tant que
2) PGCD ← A
3) Fin PGCD
```

2. Calcul du PGCD : (Récursif)

1ère méthode:

```
0) DEF FN PGCD (A, B : Entier) : Entier
1) Si (A = B) Alors
      PGCD ← A
   Sinon Si (A > B) Alors PGCD ←
      PGCD (A – B, B)
   Sinon
      PGCD ← PGCD (A, B – A)
   FinSi
2) Fin PGCD
```

2ème méthode:

```
0) DEF FN PGCD (A, B : entier) : entier
1) Si (B = 0) Faire
      PGCD ← A
      Sinon
      PGCD ← PGCD (B, A MOD B)
   Fin Si
2) Fin PGCD
```

Cours Algo Eddine

3. Calcul du PPCM : (Itératif)

```
0) DEF FN PPCM (A, B : entier) : entier
1) Si (B > A) Alors C ← A
        A ← B B ← C
    Fin Si
2)  M ← A
    Tant que (M mod B ≠ 0) Faire M ← M
          + A
    Fin Tant que
3)  PPCM ← M
4)  Fin PPCM
```

4. Nombre Premier : (Itératif)

```
0) DEF FN Premier (X : Entier) : Booléen 1) j □
2
   Tant que (X MOD j ≠ 0) ET (j <= X DIV 2) Faire j □ j
      + 1
    Fin Tant que
2) Premier □ j > X DIV 2
3) Fin Premier
```

5. Décomposition en facteurs premiers :

```
0)  DEF FN Fact_prem (N : Entier) : Chaîne
1)  [K←1  ;  CH ← ""]
   Répéter
     K ← K + 1
     Si N MOD K = 0 Alors
          Convch (K, CH1)
          CH ← CH + "*" + CH1 N ← N
          DIV K
     Sinon    K ← K + 1 Fin Si
   Jusqu'à (N = 1)
2) Efface (CH,1,1)
3) Fin Fact_prem
```

6. Conversion entre les bases (B1 → 10) :

```
0) DEF FN Conv_B1_10 (CH : Chaîne; B1 : Entier) : Entier Long
1) Dec ← 0
   Pour i de 1 à Long (CH) Faire
      Si CH[i] Dans ["A".."F"] Alors N ← Ord (CH[i]) - 55
        Sinon Val (CH[i], N, e)
     FinSi
     Dec ← Dec + N * FN Puissance (B1, Long (CH) - i)
   FinPour
2) Conv_B1_10 ← Dec
3) Fin Conv_B1_10
```

Ghomri Abd Eddine

7. Conversion entre les bases (10 → B2):

```
0) DEF FN Conv_10_B2 (N : Entier Long; B2 : Entier) : Chaîne 1) CH
← " "
    Répéter
      R ← N MOD B2
      Si R ≥ 10
        Alors CH1 ← CHR (R + 55)
      Sinon Convch (R, CH1)
    Fin Si
    CH ← CH1 + CH N ← N
    DIV B2
    Jusqu'à (N = 0)
2) Conv_10_B2 ← CH
3) Fin Conv_10_B2
```

Ghanmi Alaa Eddine

Ghanmi Alaa Eddine

Les algorithmes d'approximation

1- Détermination et affichage du point fixe :

DEF PROC **Point_fixe** (E : Réel)
Résultat = Ecrire (" Le point fixe est", X1, " trouvé après ", i, " itérations")
X1, i = [i ← 0 ; X1 ← 1] Répéter
X0 ← X1
X1 ← f (X0)
i ← i + 1
Jusqu'à (ABS (X1 - X0) < E)
Fin **Point_fixe**

2- Détermination de la valeur approchée d'une constante :

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots$$

DEF FN **E_Fact** (epsilon : Réel) : Réel
Résultat = E_Fact
E_Fact ← S2
S2 = [i ← 0 ; S2 ← 1] Répéter
i ← i + 1
S1 ← S2
S2 ← S2 + 1 / **FN Fact (i)**
Jusqu'à (ABS (S2 - S1) < epsilon)
Fin **E_Fact**

$$\frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots$$

DEF FN Pi_Euler(E : Réel) : Réel Résultat = Pi_Euler
Pi_Euler ← RacineCarré (6*S2)
S2 = [i ← 1 ; S2 ← 1] Répéter
i ← i + 1
S1 ← S2
S2 ← S2 + 1 / **Carré (i)**
Jusqu'à (ABS (RacineCarré (6*S2) - RacineCarré (6*S1)) < E)
Fin **Pi_Euler**

$$\frac{\pi}{2} = \frac{3 * 5 *.7 * 9 * 11 * \ldots}{2 * 6 * 8 * 10 * 12 * \ldots}$$

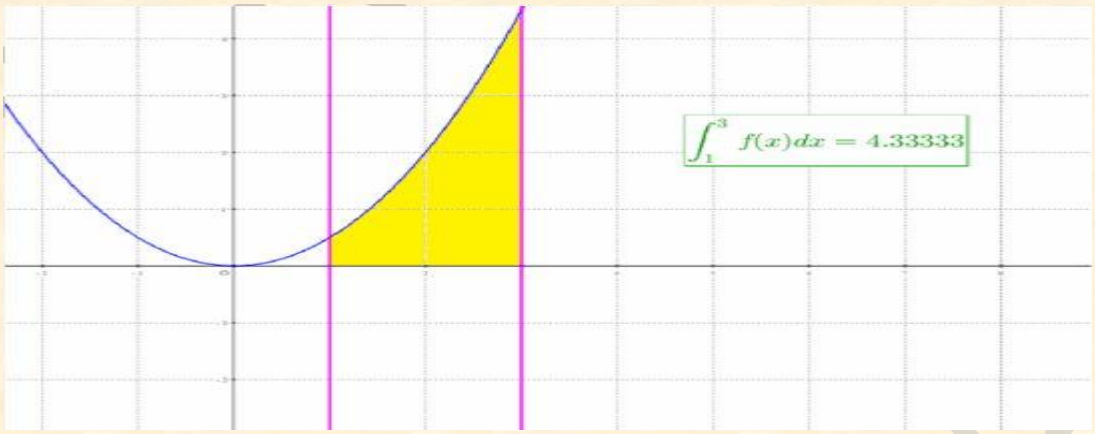
```
DEF PROC Pi_Euler2 (E : Réel)
Résultat = Ecrire (" Le valeur approchée de Pi est ", 2*P2)
P2 =      i ← 3          Répéter
          j ← 4              i ← i + 2
         P2 ← 3/2            j ← j + 2
                             P1 ← P2
                             P2 ← P2 * i / j
                         Jusqu'à (ABS (2*P2 - 2*P1) < E)
Fin Pi_Euler2
```

$$\frac{\pi}{4} = \frac{3}{4} + \frac{1}{2 * 3 * 4} - \frac{1}{4 * 5 * 6} + \frac{1}{6 * 7 * 8} \ldots$$

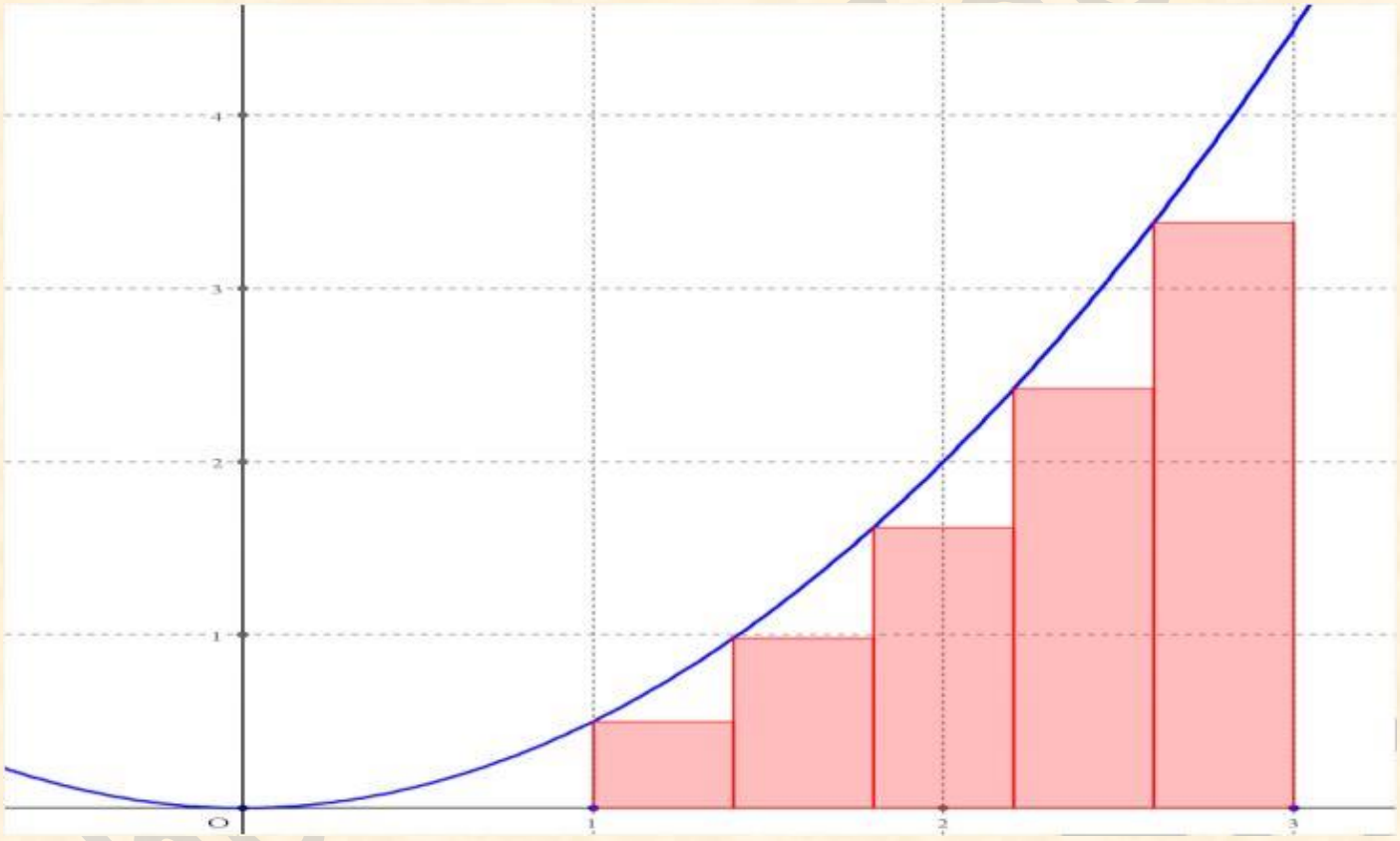
```
DEF FN Pi_APP (E : Réel): Réel
Résultat = Pi_APP
Pi_APP ← 4*S2
S2 =      i ← 0          Répéter
          j ← 1              i ← i + 2
          k ← 2              j ← j + 2
          S2 ← 3/4           k ← k + 2
        signe ← 1            S1 ← S2
                             S2 ← S2 + signe * 1 / (i * j * k)
                             signe ← - signe
                         Jusqu'à (ABS (4*S2 - 4*S1) < E)
Fin Pi_APP
```

Ghanmi Aa Eddine

3- Calcul d'aire (Principe)



a) Méthode des rectangles



DEF FN **Rectangles** (N : Entier; a, b: Réel) : Réel

Résultat = **Rectangles**

Rectangles ← S * h

S = (X ← a ; S ← 0 ; h ← (b - a) / N)

Pour i de 1 à N Faire

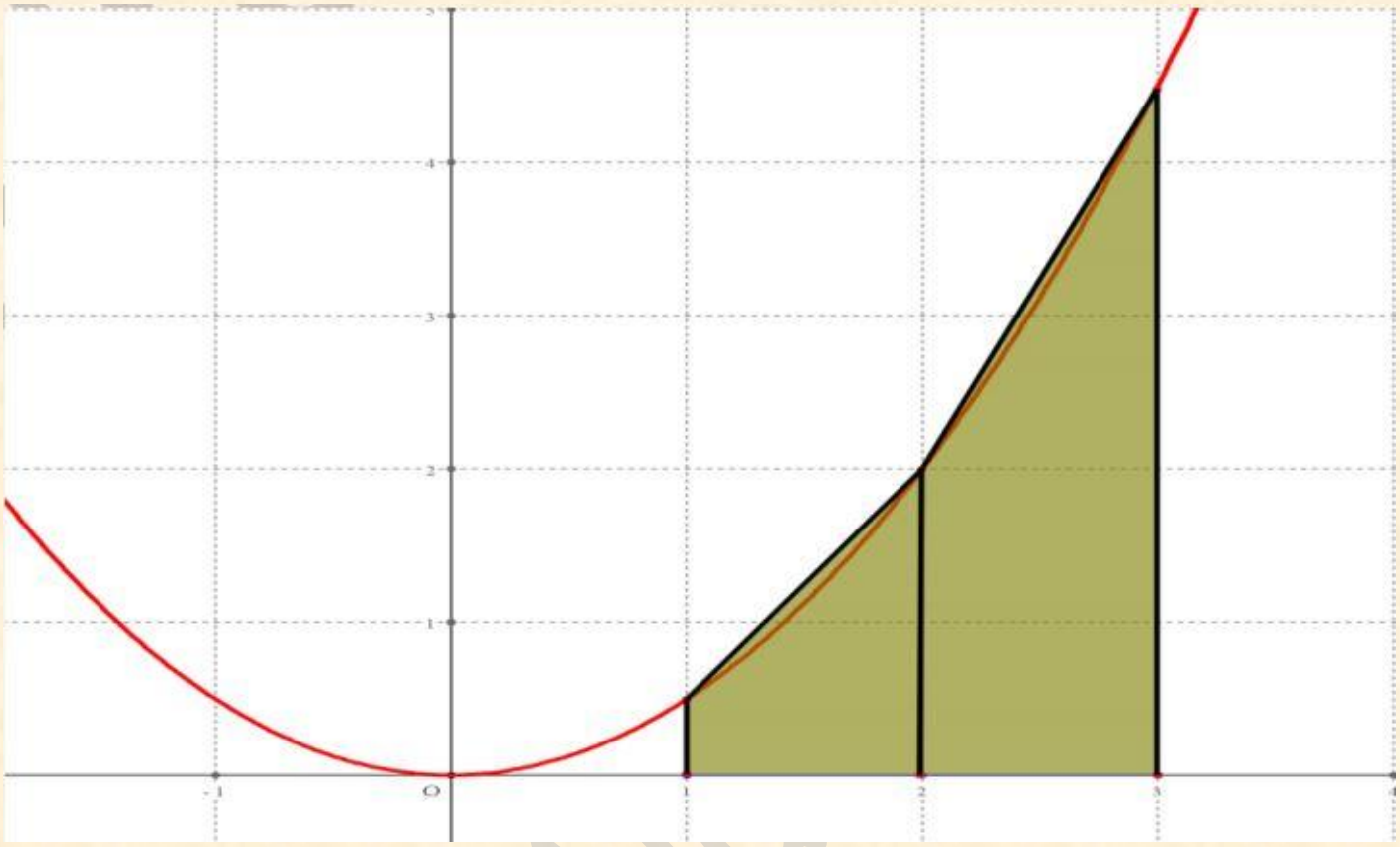
S ← S + **FN** f(x)

X ← X + h

Fin Pour

Fin **Rectangles**

b) Méthode des trapèzes



DEF FN **Trapezes** (N : Entier; a, b: Réel) : Réel

Résultat = **Trapezes**

Trapezes ← S * h / 2

S = (X ← a
S ← 0
h ← (b - a) / N)

Pour i de 1 à N Faire
S ← S + **FN** f(x) + **FN** f(x+h)
X ← X + h
Fin Pour

Fin **Trapezes**