Chapitre 3

Les Itérations, Notions de modules, Suites récurrentes

I Itérations simples

I.1)Exemple

Soit le calcul de la paie.

Lorsque l'entreprise comprend plusieurs employés, on exécute l'algorithme de calcul autant de fois que nécessaire.

Hypothèse : Soit Nbe $\Rightarrow$ nombre d'employés dans l'entreprise.
On utilise un compteur qui comptabilise le nombre de fois que le traitement associé au calcul de la paie sera effectué.

Soit **c** l'identifiant du compteur.

Algo :

algo Paie_totale	**Ord**	**Lexique**
pour c ← 1 à Nbe faire	2	Nbe (**entier**) : nbre d'employés
∣ ecrire (nom,salnet,salbrut)		c (**entier**) : compte le nombre d'employés
∣ salnet ← …….		à traiter
∣ …….		nom (**chaîne**) : …
∣ lire (salhor,nbh,prime)		…..
fpour		……
lire (nbe)	1	

Calcul pour un salarié

Les valeurs successives du compteur : **c**=1, **c**=2, … , **c**= Nbe

c←1 signifie que la valeur initiale de **c** est 1.

c←1 à Nbe : c prend automatiquement ses valeurs entre 1 et Nbe.

I.1)Définition

Une itération simple permet de définir une suite d'opérations qui se répète un certain nombre de fois.

Le nombre de répétitions est défini par la variation d'un **indice** (ou *compteur* ou *itérateur*), de type **scalaire** (*entier*, *caractère*, *booléen*) qui prend ses valeurs dans un intervalle **donné**.

Caractéristiques d'une itération simple :

- ✓ un *indice* : variable
- ✓ un *intervalle de variation* : ensemble borné par une borne inférieure et une borne supérieure
- ✓ un *pas de variation* : +1 ou -1 (pas par *défaut* = +1)
- ✓ un *ensemble d'instructions*

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

a) Itération croissante

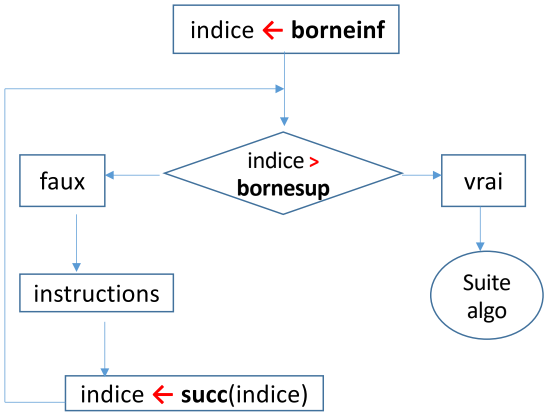
pour indice ← borneinf à bornesup faire

| Instructions

fpour

Remarque : borneinf <= bornesup

Fonctionnement :



Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

b) Itération décroissante

pour indice ← bornesup à borneinf [pas -1] faire

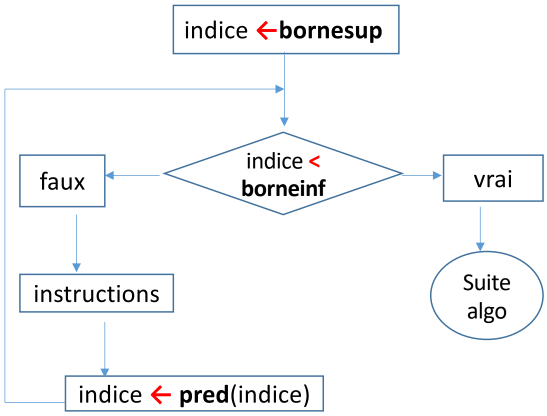
| Instructions

fpour

Remarques :

- **bornesup >= borneinf**
- **[pas -1] : optionnel**

Fonctionnement :



Exemples simples

algo ….	Ord	Lexique
pour i ← 1 à 10 faire		i (entier)
| ecrire (i)		
fpour		

algo ….	Ord	Lexique
pour a ← 2 à N faire	2	a (entier)
| ecrire (3*x+1)		N,x (entier)
fpour		
lire (N,x)	1	

algo ….	Ord	Lexique
pour c ← 'Z' à 'A' faire		c (caractère) :compteur
| ecrire(c)		
fpour		

II Itération Conditionnelle

II.1) Définition

- Une itération conditionnelle permet de définir une suite d'opérations qui se répète un certain nombre de fois. L'arrêt de l'itération est déterminé par la vérification d'une certaine condition.

- Il existe 2 formes d'itérations conditionnelles :

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

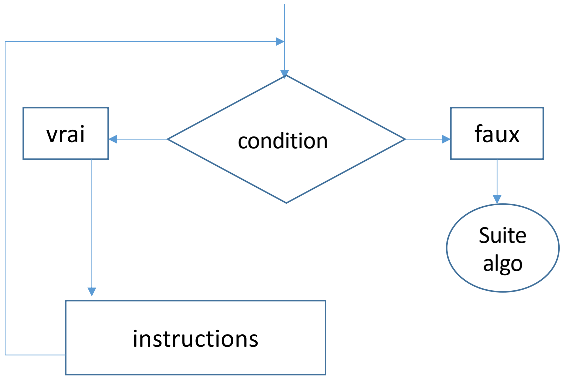
Forme 1

tantque condition=vrai faire
| instructions
fintantque

Ecriture simplifiée :

tq condition=vrai faire
| instructions
ftq

Fonctionnement



Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

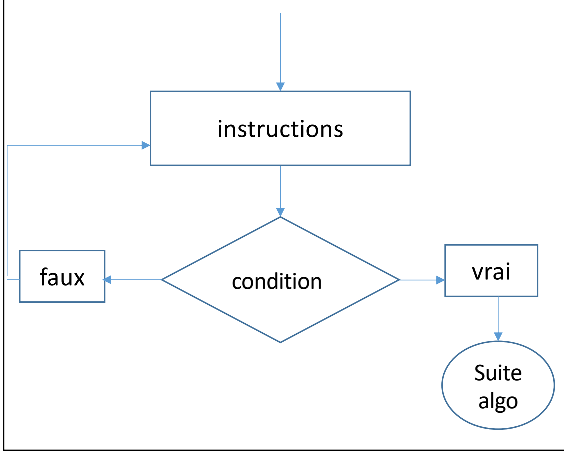
Forme 2

repeter
| instructions
jusqua condition=vrai

Ecriture simplifiée :

repeter
| instructions
jqa condition=vrai

Fonctionnement :



II.2) Différence entre tantque et répéter

- **Répéter** : La partie instructions est exécutée au moins une fois avant de vérifier la condition d'arrêt.
- **Tantque** : la partie instructions n'est exécutée que si la condition pour s'arrêter n'est pas validée (= la condition d'arrêt est fausse).

Remarque : une itération simple peut être exprimée à l'aide d'une itération conditionnelle, l'inverse n'est pas toujours possible.

Exemple :

algo …	Ord	Lexique
pour i ←1 à N faire	2	i (entier) : compteur
| ecrire (i)		N (entier) : nombre d'entiers à afficher
fpour		
lire(N)	1	

algo …	Ord	Lexique
tq i ≤ N faire	3	i, N (entier)
| ecrire (i)		
| i ← succ (i)		
ftq		
lire (N)	1	
i ←1	2	

III La notion de module

III.1) Exemple

Soit le calcul de la moyenne d'un étudiant à partir de 3 notes : maths, français, anglais.
Soit Nbe le nombre d'étudiants pour lesquels ce calcul est fait.

algo Calcul_moyenne	Ord	Lexique
pour i ← 1 à Nbe faire	2	i (entier) : compteur
| écrire (nom, prenom, Moy)	2.6	Nbe (entier) : nbre d'étudiants
| Moy ← somnot / somcoef	2.5	nom,prenom (chaine)
| sommnot ← noteM*coefM+noteA*coefA + noteF*coefF	2.3	Moy (reel) : moyenne
| somcoef ← coefM+coefF+coefA	2.4	noteM, noteF, , noteA (reel)
| lire (nom,prenom)	2.1	coefF(cste /reel=2.0)
| lire (noteA,noteM,noteF)	2.2	coefM(cste /reel=3.5)
		coefA(cste /reel=1.5)
fpour		
lire (Nbe)	1	

Ensemble d'instructions (+lexique correspondant) servant à calculer la moyenne d'un étudiant.

Le même traitement est utilisé pour tous les étudiants concernés.

Cet ensemble peut être écrit une et une seule fois et réutilisé à chaque fois que c'est nécessaire.

C'est la notion de sous-algorithme ou module

III.2) Définition

- Un module est un ensemble de définitions formant un algorithme identifié par un ***nom*** et ayant la même structure que l'algorithme d'origine (ou *algorithme principal*). Il peut être utilisé dans d'autres algorithmes et ceci autant de fois que nécessaire. Il possède son propre ***ordonnancement*** et son propre ***lexique***.

Appelons **Moyenne** le module de calcul de la moyenne d'un étudiant.

Ce qui donne, dans un premier temps, pour l'exemple de la moyenne :

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

Un Algorithme principal qui appelle un module

algo Calcul_Moyenne	**Ord**	**Lexique**
pour i ← 1 à Nbe faire	2	i (entier) : compteur
| Moyenne		Nbe (entier) : nbre étudiants
fpour		Moyenne (**module**) : calcule une moyenne
lire (Nbe)	1	

module Moyenne	**Ord**	**Lexique**
ecrire (nom,prenom,moy)	6	nom, prenom (chaine)
moy ← somnot / somcoef	5	Moy (reel)
somnot ← noteM*coefM + noteA*coefA + noteF*coefF	3	noteM, noteF ,noteA (réel)
somcoef ← coefM + coefF + coefA		coefF (cste /reel = 2)
lire (nom,prenom)	4	coefM (cste/réel=3.5)
lire (noteA,noteM,noteF)	1	coefA (cste/réel =1.5)
	2	somnot, somcoefA (reel)

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

III.3) Remontée des constantes

- Dans l'exemple précédent, les coefficients des matières sont identiques pour tous les étudiants. On peut les définir une fois pour toutes dans l'algorithme principal : on procède à la **remontée des constantes** dans l'algo principal.
- Les coefficients seront disponibles aussi bien pour le module **Moyenne** que pour tout autre sous-algo défini dans l'algo principal.

Plus généralement, tout objet défini dans l’algo principal est accessible pour tout module qui y est créé.

III.4) Utilisation des modules

Un module permet :

- ❖De décomposer le problème à résoudre en sous-problèmes et de donner un nom à chacun
- ❖De résoudre chaque sous-problème par la réalisation d’un module (ce qui implique une résolution du problème initial à l’aide d’une analyse par le résultat)

Remarques

- Un module peut :
  - utiliser des variables et des constantes définies dans l'algo principal, sans modifier les variables
  - utiliser les variables de l'algorithme principal en les modifiant
- Un algo **appelle** un module et un module **est appelé** par un algo
- Un appel de module est une **instruction** (au même titre qu'une affectation, une itération, …)

III.5) Module avec paramètres

Exemple : on veut calculer la moyenne de chaque étudiant d'une promo. De plus, on veut connaître la meilleure et la plus mauvaise note dans chaque matière.

Précédemment, dans le module ***Moyenne*** :

- on *saisit* : nom, prénom, note
- on *calcule* la moyenne
- on *affiche* le résultat

Pour répondre à la question, il faut

- redemander au niveau de l'algo les notes pour chaque étudiant
- conserver à chaque étudiant traité la plus grande et la plus petite note dans une matière.

Une meilleure solution

- **lire** une **seule** fois les notes d'un étudiant dans **l'algo** principal (*ainsi que son identité*)
- **appeler** le module **Moyenne** pour le calcul de la moyenne à partir des notes
- **effectuer** les opérations nécessaires dans **l'algo** pour déterminer la *meilleure* et la *plus mauvaise* des notes dans une matière
- faire le choix d'**afficher** les résultats non pas dans le module **Moyenne** mais au niveau de **l'algorithme principal** avec les changements que cela implique

<table>
<tr><th>algo Calcul_Moyenne</th><th>Ord</th><th>Lexique</th></tr>
<tr><td>pour i ← 1 à Nbe faire</td><td>9</td><td>bestA, bestM, bestF (réel) : meilleures notes</td></tr>
<tr><td>|ecrire(nom, prenom)</td><td>9.7</td><td></td></tr>
<tr><td>| ecrire(moy)</td><td>9.8</td><td>worstA, worstF, worstM (réel) : pires notes</td></tr>
<tr><td>| Moyenne (noteM[1], noteF[1], noteA[1], moy[2]) (*)</td><td>9.3</td><td></td></tr>
<tr><td>| lire (noteM, noteF, note A)</td><td>9.2</td><td>nom, prénom (chaine)</td></tr>
<tr><td>| lire (nom, prenom)</td><td>9.1</td><td></td></tr>
<tr><td>| si bestM < noteM alors</td><td>9.4</td><td>noteF, noteM, noteA (réel)</td></tr>
<tr><td>| | bestM ← noteM</td><td></td><td></td></tr>
<tr><td>| sinonsi worstM > noteM alors</td><td></td><td>Nbe(entier) : nombre d'étudiants</td></tr>
<tr><td>| | worstM ← noteM</td><td></td><td></td></tr>
<tr><td>| fsi</td><td></td><td>Moyenne (module) : calcule la moyenne d'un étudiant</td></tr>
<tr><td>| -- idem pour l'anglais</td><td>9.5</td><td></td></tr>
<tr><td>| -- idem pour le français –</td><td>9.6</td><td>coefM(cste/réel=...)</td></tr>
<tr><td>| fpour</td><td></td><td></td></tr>
<tr><td>bestM ← 0</td><td>1</td><td>coefF(cste/réel=...)</td></tr>
<tr><td>bestA ← 0</td><td>2</td><td></td></tr>
<tr><td>bestF ← 0   Valeurs initiales</td><td>3</td><td>coefA(cste/réel=...)</td></tr>
<tr><td>worstM ← 20</td><td>4</td><td></td></tr>
<tr><td>worstF ← 20</td><td>5</td><td>i(entier) : compteur</td></tr>
<tr><td>worstA ← 20</td><td>6</td><td></td></tr>
<tr><td>ecrire("Saisir le nombre d'étudiants : ")</td><td>7</td><td>moy(réel) : moyenne</td></tr>
<tr><td>lire(Nbe)</td><td>8</td><td></td></tr>
<tr><td>…….</td><td></td><td></td></tr>
<tr><td>…….</td><td></td><td></td></tr>
</table>

(*) : appel au module Moyenne avec paramètres
[1] : paramètres données utilisés par le module
[2] : paramètre résultat produit par le module moyenne
[1+2] : paramètres d'**appel** ou paramètres **effectifs**

Dans la définition du module :

module Moyenne (↓nM[1] : réel ; ↓nA[1] : réel ; ↓nF[1] : réel ; ↑m[2] : réel)	**Ord**	**Lexique**
m ← somnotes/somcoef	3	somnotes (réel)
somnotes ← nM*coefM + nA*coefA + nF*coefF	1	somcoef(réel)
somcoef ← coefM + coefF + coefA	2	

[1+2] : Paramètres formels

Définition :

- Un module avec paramètres est une suite de traitements à effectuer donnant zéro ou plusieurs résultats à partir d'un ensemble d'arguments ou paramètres (qui peut être vide).
- Un paramètre est caractérisé par
  - ✓ un *nom*
  - ✓ un *mode* ⇒ 3 modes possibles :
    - **Entrée :** représenté par ↓
    - **Sortie** : représenté par ↑ sortie
    - **Entrée/sortie** : représenté par ↕ ou ↓↑
  - ✓ un *type*

Le mode entrée : le paramètre est **non modifiable** (seulement accès en consultation, pas d'affectation)

Le mode sortie : le paramètre est **modifiable** (et utilisable après une première initialisation)

Le mode entrée/sortie : le paramètre est **consultable** et **modifiable**

Syntaxe :

Appel de module : Nom_Module(pe_1, pe_2, …, pe_n)

Définition du module :

module Nom_Module (* pf_1 : typf_1; * pf_2 : typf_2 ; ….* pf_n : typf_n)	**Ord**	**Lexique**
Liste d'instructions		…… …….

Avec : * ⇒ ↓, ↑, ↕ (ou ↓↑)

pf_ i ⇒ paramètre formel i; typf_i : type paramètre formel i

pe_i ⇒ paramètre effectif i; type_i : type paramètre effectif i

Remarque : on peut regrouper ensemble, dans la définition de l'entête du module, les paramètres formels ayant même type et même mode.

Exemple : ↓a,b,c : entier => trois paramètres d'entrée de type entier.

Correspondances entre paramètres :

Formels		Effectifs
Séparés par ;		Séparés par ,
Mode précisé : ↓, ↑, ↕ (ou ↓↑)		Mode non précisé à l'appel
Type précisé		Type non précisé à l'appel
Nombre de paramètres formels	=	Nombre de paramètres effectifs
Type des paramètres formels	=	Types des paramètres effectifs
L'ordre des paramètres formels	=	L'ordre des paramètres effectifs
Ne figurent pas dans le lexique du module		Si pe est un paramètre correspondant à un ↓pf ou ↕pf alors pe doit obligatoirement avoir une valeur à l'appel du module

Exemple :

Soit le module qui calcule la somme de 2 entiers, le résultat doit être utilisé dans l'algorithme principal.

algo Principal	**Ord**	**Lexique**
lire (n1, n2) Calculsomme (n1, n2, somme) écrire(somme) Calculsomme(10, somme, s) écrire(s) **si** s > somme **alors** |		n1, n2 (entier) : à sommer somme, s (entier) : résultats Calculsomme (**module**) : somme de 2 entiers

module Calculsomme (↓a, b : entier;↑s : entier)	**Ord**	**Lexique**
s ← a + b		

IV Les suites récurrentes : notion de variable récurrente

Soit l'exemple du calcul de la moyenne :

On veut calculer la moyenne générale de toute une promo et l'afficher dans l'algorithme principal.

Forme d'affichage du résultat voulue :

nom_etud1 prenom_etud1 moy_etud1

nom_etud2 prenom_etud2 moy_etud2

… … … … …

nom_etudNbe prenom_etudNbe moy_etudNbe

Moyenne générale : moygen

Réalisation :

algo Calcul_Moyenne	Ord	Lexique
ecrire(" Moyenne générale =", moygen)	4	nbe (entier) : nbre etudiants
pour i ← 1 à nbe faire	2	nom,prenom (chaine)
| ecrire (nom,prenom,moy)	2.4	moy (reel)
| Moyenne (noteM, noteF, noteA, moy)	2.3	noteM, noteF,, noteA (reel)
| lire (nom,prenom)	2.1	moygen (reel) : moyenne générale
| lire (noteA,noteM,noteF)	2.2	coefF (cste/reel = 2)
| (*) calcul de la moyenne générale ⇒ somme des moyennes	2.5	coefM (cste/reel = 3,5)
fpour		coefA (cste/reel = 1,5)
		somcoef (cste/reel = coefA+coefM+coefF)
		(*)(sommoy (reel) : somme des moyennes)
lire (nbe)	1	Moyenne (**module**)
(*)moygen ← sommoy/nbe	3	

module Moyenne (↓nM, nF, nA : réel; ↑m : réel)	Ord	Lexique
m ← somnot / somcoef	2	somnotes (reel)
sommnot← nM*coefM + nA*coefA + nF*coefF	1	

- En math : moygen = (somme des moyennes) / (nbre d'étudiants)

$$\text{moygen} = \left(\sum_{i=1}^{nbe} \text{moy}_i\right)/\text{nbe}$$

Où moy_i est la moyenne de l'étudiant i.

Actuellement, on ne mémorise que la moyenne de l'étudiant courant (1 seul à la fois).

$\text{sommoy} = \left(\sum_{i=1}^{nbe} \text{moy}_i\right) = \text{moy}_1 + \text{moy}_2 + \text{moy}_3 + \ldots\ldots + \text{moy}_{nbe}$

Etapes de calcul :

i = 1 : moy_1 : sommoy = $\text{sommoy}_1 = \text{moy}_1$

i = 2 : moy_2 : sommoy = $\text{sommoy}_2 = \text{moy}_1 + \text{moy}_2 = \text{sommoy}_1 + \text{moy}_2$

…

i = nbe : moy_{nbe} = sommoy = $\text{sommoy}_{nbe} = \text{moy}_1 + \ldots + \text{moy}_{nbe}$

$= \text{sommoy}_{nbe-1} + \text{moy}_{nbe}$

Conclusion : ce qui est calculé à l'étape i est ajouté à ce qui a été calculé à l'étape i-1 et le tout stocké dans la même variable résultat :

$\text{sommoy}_i = \text{sommoy}_{i-1} + \text{moy}_i$

- En algorithmique :

$\text{sommoy}_i \leftarrow \text{sommoy}_{i-1} + \text{moy}_i$

sommoy_i : somme des moyennes à l'étape i

sommoy_{i-1} : sommes des moyennes à l'étape précédente : i-1

moy_i : moyenne à l'étape i (étape courante)

sommoy est une variable récurrente : sa valeur à l'étape i dépend de son calcul à l'étape précédente :

(*) sommoy ← sommoy + moy

Ceci donnera pour la moyenne générale :

moygen ← sommoy / nbe

Dans l'algorithme :

* moygen ← sommoy / nbe | | sommoy (reel)

Algorithme complet :

algo Calcul_moyenne	**Ord**	**Lexique**
ecrire("moyenne générale = ", moygen)	6	i (entier) : compteur
moygen ← sommoy / nbe	5	
pour i ← 1 à nbe faire	4	
| ecrire (nom,prenom,moy)	4.5	nbe (entier) : nbre étudiants
| Moyenne (noteM, noteF, noteA, moy)	4.3	nom, prenom (chaine)
| lire (nom,prenom)	4.1	moy (reel) : moyenne
| lire (noteA,noteM,noteF)	4.2	noteM, noteF,, noteA (reel)
| sommoy ← sommoy + moy	4.4	
fpour		sommoy (reel) somme des moyennes
sommoy ← 0	3	coefF (cste/reel = 2)
ecrire(" Saisir le nombre d'étudiants : ")	1	coefM (cste/reel = 3,5)
		coefA (cste/reel = 1,5)
lire (nbe)	2	moygen (reel) : moyenne générale
		Moyenne (**module**) : calcule une moyenne

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

Définition :

Une suite récurrente est une suite définie comme suit :

- son premier terme est donné
- son terme général est défini en fonction du terme précédent

En math : $u_0=v$

$u_n=f(u_{n-1})$: le terme de rang *n* est fonction du terme de rang *n-1*

En algorithmique, pour la réaliser on utilise :

- une variable récurrente
- une itération simple

Exemple : Calcul de la somme des n premiers entiers (n étant un nombre lu).

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

❑En maths :

$S_0 = 0$

$S_n = S_{n-1} + n$

Si n =4, la somme des 4 premiers entiers = somme des 3 premiers entiers + 4

❑En Algo :

algo Somme_N	Ord	Lexique
ecrire(somme)	4	somme (entier) : résultat
pour i ← 1 à n faire	3	i, n (entier)
| somme ← somme + i		
fpour		
somme ← 0	2	
lire(n)	1	

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

V Notion de fonction

V.1) Exemple

En math :

Soit les fonctions :

Carre : $\mathbb{N} \rightarrow \mathbb{N}$

Carre(2) = 4

Carre(8) = 64

Carre(i) = i^2

Puissance : $\mathbb{N} * \mathbb{N} \rightarrow \mathbb{N}$

Puissance(2,3)=8

Puissance (i,j) = i^j

Plus généralement :

Soit une fonction F définie :

F : $D_1 * D_2 * \ldots * D_n \rightarrow D_A$

- ✓ domaine de départ : produit cartésien de plusieurs domaines D_i
- ✓ domaine d'arrivée : 1 seul domaine D_A

V.2) Définition

Une fonction est une suite d'opérations donnant un résultat à partir d'un ensemble d'arguments ou paramètres.

On distingue 2 types de fonctions :

- prédéfinies : exemple : carre, puissance, racine carrée, …
- écrites par nous même

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

La fonction calcule un et un seul résultat. Ce résultat est une donnée donc il a un type.
La fonction aura un type : le type du résultat calculé.

V.3) **Exemple :**

Pour le calcul de la moyenne d'un étudiant nous avions :

module Moyenne(↓m, f, a : réel; ↑avg : réel)

Dans l'algo on appelle le module avec les valeurs effectives de calcul :

Moyenne(nm, nf, na, moy)

On remarque que le module calcule un seul résultat ⇒ On peut le définir sous forme de fonction.

Moyenne : $\mathbb{R} * \mathbb{R} * \mathbb{R} \rightarrow \mathbb{R}$

(D_notemath ∗ D_notefr ∗ D_noteangl → D_moyenne)

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

Syntaxe de la définition de la fonction Moyenne :

fonction Moyenne(nM, nF, nA : réel) : réel	**Ord**	**Lexique**
m ← (coefF*nF + coefM*nM + coefA*nA)/(coefF+coefM+coefM)	1	m (réel) : moyenne d'un étudiant
Moyenne ⇐ m	2	

On associe le résultat calculé au nom de la fonction (différent de l'affectation à une variable)

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

Syntaxe pour l'appel à la fonction Moyenne :

<table>
<tr><th>algo Calcul_moyenne</th><th>Ord</th><th>Lexique</th></tr>
<tr><td>pour i ← 1 à nbe faire</td><td>3</td><td>i(entier) : compteur</td></tr>
<tr><td>| ecrire (nom,prenom,moy)</td><td>3.5</td><td>nbe (entier) : nb etudiants</td></tr>
<tr><td>| moy ← Moyenne(noteM, noteF, , noteA) ✹</td><td>3.3</td><td>nom, prenom (chaine)</td></tr>
<tr><td>| lire (nom,prenom)</td><td>3.1</td><td>moy(reel)</td></tr>
<tr><td>| lire (noteA,noteM,noteF)</td><td>3.2</td><td>noteM, noteF, noteA (reel)</td></tr>
<tr><td>| sommoy ← sommoy + moy</td><td>3.4</td><td>coefF (cste/reel = 2)</td></tr>
<tr><td>fpour</td><td></td><td>coefM (cste/reel =3,5)</td></tr>
<tr><td>sommoy ← 0</td><td>2</td><td>coefA (cste/reel= 1,5)</td></tr>
<tr><td>lire (nbe)</td><td>1</td><td>sommoy(reel) somme des</td></tr>
<tr><td>ecrire("moyenne générale = ", moygen)</td><td>5</td><td>moyennes de tous</td></tr>
<tr><td>moygen ← sommoy / nbe</td><td>4</td><td>Moyenne (fonction/reel) : calcule</td></tr>
<tr><td></td><td></td><td>une moyenne</td></tr>
</table>

✹ Ces deux instructions peuvent s'ecrire en une seule :

ecrire (nom, prenom, Moyenne(noteM, noteF, noteA))

Chapitre 3 Les Itérations, Notions de modules, Suites récurrentes

Remarques :

- un appel de fonction n'est pas une instruction (*contrairement à l'appel d'un module)*
- un nom de fonction n'est pas une variable. On ne peut pas l'utiliser comme variable récurrente.
- tous les paramètres d'une fonction sont des paramètres d'entrée.
- un appel de fonction est remplacé par la valeur calculée :

Exemples :

- si Moyenne (nm,nf,na) > 10 alors (*utilisation dans une comparaison*)

  ....
- moygen ← moygen + Moyenne (12, 4, note) (*utilisation dans une expression calculée*)
- ecrire (''Moyenne = '', Moyenne (n1, n2, n3)) ……

V.4) Syntaxe formelle

Définition :

fonction Nom_Fonction(pf_1 : typ_pf1; pf_2 : typ_pf2; …….) : typeresultat	**Ord**	**Lexique**
… Calcul …		…..
…….		
Nom_fonction ⇐ résultat		…..
: associe 1 valeur à la fonction pour le renvoi du résultat à l'appelant		

Utilisation dans l'algorithme :

Nom_Algo_Principal	**Ord**	**Lexique**
[…]		pe_1 (type1)
Nom_variable ← Nom_Fonction (pe_1, pe_2, ……)		pe_2 (type2)
OU ecrire(Nom_Fonction (pe_1, pe_2, ……)		…….
OU ……		Nom_fonction(**fonction/type résultat**) : commentaire