NOTATION ALGORITHMIQUE

- Les éléments entre crochets **[** et **]** sont optionnels. Ex. : **nom_1[, nom_2]**
- Les éléments séparés par **|** s'excluent mutuellement. Ex. : **fromage | dessert**
- Les termes en gras et souligné sont des mots-clés. Ex. : **ecrire-ecran**
- Les mots-clés doivent être écrits sans modification ni abréviation !
- Les termes en italique désignent des éléments génériques. Ex : *valeur_initiale*

1 Forme générale d'un programme

CONSTANTES
Déclarations de constantes globales
TYPES GLOBAUX
Déclarations de types globaux
VARIABLES GLOBALES
Déclarations de variables globales → à éviter la plupart du temps
SOUS-PROGRAMMES
Déclarations de sous-programme
PROGRAMME PRINCIPAL
% But du programme commenté en français %
DÉCLARATIONS LOCALES
Déclarations de types, de constantes ou de variables
ACTIONS
Description des différentes actions
dans l'ordre où elles seront exécutées.
FIN ACTIONS
FIN PROGRAMME

2 Les déclarations de types, de constantes et de variables

Déclarations de constantes
type_constante : NOM_CONSTANTE = valeur
Déclarations de types TABLEAU et ENREGISTREMENT
nom_type_tab = TABLEAU [Nb_Max_Elt][Nb_max_elt2] de type_elements
nom_type_enr = ENREGISTREMENT
type : Nomchamp1
[type : Nomchamp2]
Fin_Enregistrement
Déclarations de variables
type_variable : nom_variable
ou **type_variable : nom_1 [, nom_2 [, ...]]** (plusieurs de variables de même type)

3 Les déclarations de sous-programmes

Déclaration de procédures

PROCEDURE *nom_procédure* (*liste_paramètres_formels*)
% But du sous-programme commenté en français %
DÉCLARATIONS LOCALES
Déclarations de types, de constantes ou de variables
ACTIONS
Description des différentes actions
dans l'ordre où elles seront exécutées
par la procédure.
FIN ACTIONS
FIN PROCEDURE

Déclaration de fonctions

FONCTION **type_resultat :** *nom_fonction* (*liste_paramètres_formels*)
% But du sous-programme commenté en français %
DÉCLARATIONS LOCALES
Déclarations de types, de constantes ou de variables
ACTIONS
Description des différentes actions
dans l'ordre où elles seront exécutées
par la fonction.
RETOURNE *valeur_résultat*
FIN ACTIONS
FIN FONCTION

Déclaration des listes de paramètres formels

(type : [param_1[, param_2[,...]] , type : [param_i[, param_j[, ...]] , [...]]])

Chacun des paramètres peut être :
encadré si le paramètre est en SORTIE
souligné si le paramètre est en ENTREE
encadré et souligné si le paramètre est en ENTREE/SORTIE

La liste des paramètres peut être vide et se réduit alors à une paire de parenthèses ( ).
Les paramètres de même type sont groupés et reliés par des virgules.
Tout paramètre (ou succession de paramètres entre virgules) doit être suivi d'un type !

4 Les différentes actions

Affectation d'une valeur à une variable

nom_variable ← *valeur ou expression*
nom_Enregistrement.champ1 ← *valeur ou expression*
nom_tab[3] ← *valeur ou expression*
Le contenu de la case n°3 du tableau nom_tab reçoit la valeur ou le résultat de l'expression

Échange d'informations avec l'utilisateur

lire-clavier variable1 [, variable2 [, ...]]
ecrire-ecran *valeur1 ou expression1* [, *valeur2 ou expression2* [, ...]] [,FdL]
NB. FdL signifie Fin De Ligne

Instruction de boucle TANT QUE ... FIN TANT QUE

TANT QUE (*condition*)
| *Suite des actions à faire ou refaire* ***après avoir vérifié que la condition est vraie***
FTQ

Instruction de boucle REPETER ... TANT-QUE

RÉPÉTER
| *Suite des actions à faire une première fois* ***avant de vérifier que la condition est vraie****. Ces actions sont ensuite répétées tant que la condition est vraie*
TQ (*condition*)

Instruction de boucle POUR ...QUAND...AVEC...FIN-POUR

POUR (*indice* ← *val_init*) **QUAND** (*condition sur l'indice*) **AVEC** (*progression de l'indice*)
| Suite des actions à répéter après que *indice* ait reçu sa *valeur initiale*, quand la *condition est vraie* et en effectuant l'instruction de *progression* après chaque tour de boucle
FPR

Instruction de choix conditionnel SI ... ALORS ... SINON ...FIN-SI

SI (*condition*) **ALORS**
| Action(s) à exécuter lorsque condition est vraie
SINON
| Action(s) à exécuter lorsque condition est fausse
| [Cette partie est optionnelle et peut ne pas être présente]
FSI

Instruction de choix à plusieurs possibilités d'actions SUIVANT ... FIN SUIVANT

SUIVANT (*valeur ou expression d'un type simple*)
| *valeur1* :
| Action(s) à exécuter lorsque la valeur ou l'expression est égale à valeur1
| *valeur2* :
| Action(s) à exécuter lorsque la valeur ou l'expression est égale à valeur2
| *valeur_n* :
| Action(s) à exécuter lorsque la valeur ou l'expression est égale à valeur_n
| **[autre-cas** :
| Action(s) à exécuter dans tous les autres cas. **]**
FIN SUIVANT

Les *valeurs* (appelées label) ne peuvent être que de **type simple** : entier, caractère, élément d'un ensemble énuméré. Si plusieurs valeurs nécessitent le même traitement leurs labels sont écrits l'un au-dessus de l'autre.

Instruction d'appel de sous-programme : procédure ou fonction

L'appel à une procédure est explicite. C'est le seul objectif de l'instruction.

APPEL *nom_de_la_procédure* (liste_de_paramètres_effectifs)

L'appel à une fonction est implicite. Le résultat de la fonction (calculé pour les valeurs effectives des paramètres) est en général intégré dans une expression ou une instruction plus complexe. Par exemple :

variable ← nom_fonction (liste_paramètres_effectifs)

TANT QUE (nom_fonction (liste_paramètres_effectifs) > *valeur*) ...

La valeur retournée par la fonction est disponible dans l'expression qui provoque l'appel.
Par exemple :
abscisse ← rayon * sin(theta)

Déclaration de liste de paramètres effectifs

Un paramètre d'**entrée PEUT** être **: expression | variable | constante**

Un paramètre de **sortie** ou **d'entrée/sortie DOIT** être **: variable**

5 Les commentaires

Deux symboles ' % ' délimitent un commentaire qui peut être inséré n'importe où. Par exemple : *% texte libre destiné à informer le lecteur du programme %*