

Notation algorithmique de base

1. Structure générale d'un algorithme

Algorithme Nom de l'algorithme

```
{  
    // Bloc de déclaration des variables  
    // Programme principal  
}
```

2. Déclaration des variables

```
variable  
nom_var_1 : type_1  
nom_var_2, nom_var_3 : type_2  
...  
constante type_1 nom_const_1 ← valeur_constante
```

3. Types

entier, réel, booléen, caractère et chaîne de caractères.

Rq : Les caractères sont encadrés par des simples quotes : 'o'

Les chaînes sont délimitées par des doubles quotes : "chaîne"

4. Commentaires

// Exemple de commentaire

5. Séquence

```
{  
    Action_1  
    Action_2  
    ...  
}
```

6. Action de lecture/écriture

Saisir(var_1, var_2, ..., var_n)

Afficher(expression_1, expression_2, ..., expression_n)

Pour le retour à la ligne utiliser "\n" à la place de expression_i dans la ligne Afficher(...)

7. Affectation

variable ← expression

8. Action conditionnelle

Si condition Alors

 action_si

sinon

 action_sinon

9. Choix multiple

Selon variable

```
{  
    valeur_1 : action_1  
    valeur_2 : action_2  
    ...  
    sinon : action_sinon  
}
```

10. Actions itératives

Pour variable de indice_début à indice_fin

 action

Tant que condition

 action

Répéter

 action

Jusqu'à condition

Rq : si plusieurs actions apparaissent dans une boucle elles doivent être spécifiées sous forme de séquence (sauf dans le cas d'une boucle Répéter).

11. Sous-programmes

Syntaxe d'une procédure :

```
procédure nom_procédure()           procédure nom_procédure(paramètres)
{                                     {
    // Déclaration des variables ou // Déclaration des variables
    // Actions                       // Actions
}
```

Appel d'une procédure :

nom_procédure() ou nom_procédure(variables ou constantes)

Les paramètres sont séparés par des virgules et spécifiés sous la forme :

E nom_paramètre: <type> ou S nom_paramètre: <type> ou E/S nom_paramètre: <type>
avec E pour un paramètre d'entrée, S pour un paramètre de sortie et E/S pour un paramètre d'entrée/sortie

Syntaxe d'une fonction (les paramètres fonctionnent de manière similaire) :

```
fonction nom_fonction(): type_retour      fonction nom_fonction(paramètres): type_retour
{                                          {
    // Déclaration des variables          ou // Déclaration des variables
    // Actions                           // Actions
}
```

Pour « retourner » une valeur, on place dans le corps de la fonction une ou plusieurs actions de type :

retourner(valeur)

Appel d'une fonction :

variable ← nom_fonction() ou variable ← nom_fonction(variables ou constantes)

12. Fichiers

Manipulation par des variables de type Fichier

Création d'un nouveau fichier: Créer(nom_fichier, fichier)

Désignation d'un fichier existant : Designer(nom_fichier, fichier)

Ouverture : Ouvrir(fichier, lecture) ou Ouvrir(fichier, écriture) ou
Ouvrir(fichier, lecture/écriture)

Fermeture : Fermer(fichier)

Lecture : Lire(fichier, variable)

Écriture : Ecrire(fichier, données)

Décalage : Decaler(fichier, Début_fichier, decalage) ou

Decaler(fichier, Fin_fichier, decalage) ou

Decaler(fichier, Position_courante, decalage)

avec *decalage* un entier représentant le décalage en nombre d'éléments par rapport à la position initiale dans le fichier.

13. Structures

structure nom_structure

{

...

}

Accès à un élément d'une variable de type structure : variable.élément

14. Vecteurs et tableaux

Déclaration de variables :

nom_vecteur : vecteur de nombres_éléments type_éléments ou

nom_tableau : Tableau de dim_1 X dim_2 X ... X dim_n type_éléments

Accès à une case à une position donnée :

nom_vecteur[position] ou nom_tableau[pos_1][pos_2][...]

15. Opérations diverses

Modulo: entier_1 mod entier_2

Division entière: entier_1 div entier_2

Nombre aléatoire entre deux entiers a et b inclus: alea(a, b)

16. Pointeurs

Déclaration: nom_pointeur : pointeur sur type_pointé

Allocation mémoire d'un pointeur: nom_pointeur ← Allouer type_pointé

Opérateur Adresse: &nom_variable

Accès à l'élément pointé: *nom_pointeur

Pour accéder à un champ donné d'un pointeur sur enregistrement, la notation nom_pointeur->champ est équivalente à (*nom_pointeur).champ