

SOMMAIRE

1) Définitions.....	2
2) Organisation d'un programme.	2
2.1) Déclaration des constantes	2
2.2) Déclaration des variables	3
2.3) Définition du programme principal.....	3
2.4) Définition des fonctions et procédures.....	3
3) Affectation.....	4
4) Opérateurs - Conditions.....	4
4.1) Opérateurs.....	4
4.2) Conditions	4
5) Les Structures algorithmiques.	5
5.1) Séquencement linéaire	5
5.2) Structures de choix (ou conditionnelles)	5
5.2.1) Structure SI ... ALORS	5
5.2.2) Structure SI ... ALORS ...SINON	6
5.2.3) Structure de choix multiple	6
5.3) Structures itératives (ou répétitives)	7
5.3.1) Structure REPETER ... JUSQUA	7
5.3.2) Structure TANT QUE ... FAIRE	8
5.3.3) Structure POUR Indice ALLANT DE ... A FAIRE	8

1) DEFINITIONS.

Algorithme : Description en langage naturel de la suite des actions effectuées par un programme.

Algorigramme : Traduction graphique de l'algorithme. Parfois appelé Ordinogramme ou Organigramme.

Syntaxe : Règles d'écriture d'un langage donné.

Type de données :

Un programme peut être amené à manipuler différents types de données :

- **booléen** : valeur pouvant être soit Vraie, soit Fausse.
- **entiers** : valeur numériques entières pouvant être signées ou non signées (codées sur un ou plusieurs octets).
- **réels** : valeurs numériques codées avec une mantisse et un exposant.
- **caractère** : octet correspondant à un code ASCII.
- **chaîne de caractères** : ensemble de caractères.
- **tableau de données** : ensemble de données de même type (exemple : tableau d'entiers, tableau de réels).

Toutes ces données sont codées sous forme d'octets en mémoire.

Constante : donnée manipulée par un programme et ne pouvant être modifiée.
Exemple : Constante Pi = 3.141559

Variable : donnée manipulée par un programme et pouvant être modifiée.

Ce peut être :

- une donnée d'entrée ;
- le résultat final d'un calcul ;
- un résultat intermédiaire de calcul.

Identificateur : nom explicite d'une constante, d'une variable ou d'une fonction.
Exemples : Conversion_BCD, Resultat, Lettre...

Procédures et fonctions : Une procédure ou une fonction effectue une suite d'actions élémentaires constituant un tout.

Une fonction se différencie d'une procédure par le fait qu'elle fournit un résultat.

2) ORGANISATION D'UN PROGRAMME.

L'algorithme d'un programme est organisé en plusieurs parties :

- déclaration des constantes
- déclaration des variables
- définition du programme principal
- définition des fonctions et procédures

2.1) Déclaration des constantes

Syntaxe : **Constante** NomConstante : [Type] = Valeur

Exemples : **Constante** Pi : Reel = 3.141559
Constante NombreLettres : Entier = 10

2.2) Déclaration des variables

Syntaxe : **Variable** NomVariable : [Type]

Exemples : **Variable** Rayon : Reel
Variable Compteur : Entier
Variable Lettre : Caractere

2.3) Définition du programme principal

Le programme principal consiste en une suite d'opérations élémentaires faisant souvent appel à des fonctions ou procédures. Ces différentes opérations sont mentionnées en utilisant les structures algorithmiques décrites au paragraphe 5.

Le programme principal est délimité par les mots clefs **Début** et **Fin**

2.4) Définition des fonctions et procédures

Les procédures et fonctions peuvent nécessiter éventuellement un ou plusieurs paramètres d'entrée ou de sortie.

Un paramètre d'entrée est la référence à une variable manipulée par la procédure ou la fonction.

Un paramètre de sortie est une valeur renvoyée par une fonction.

Une fonction ou une procédure peut elle-même appeler une ou plusieurs fonctions et procédures.

Syntaxe de la déclaration d'une procédure :

Procédure NomProcédure (NomEntrée1 : [Type], NomEntrée2 : [Type],...)

Constante ~ déclaration des constantes locales ~

Variable ~ déclaration des variables locales ~

Début

~ description des actions effectuées par la procédure ~

Fin

Syntaxe de l'appel d'une procédure :

NomProcédure (NomEntrée1, NomEntrée2...)

Syntaxe de la déclaration d'une fonction :

Fonction NomFonction (NomEntrée1 : [Type], NomEntrée2 : [Type],...) : [TypeDuRésultat]

Constante ~ déclaration des constantes locales ~

Variable ~ déclaration des variables locales ~

Début

~ description des actions effectuées par la fonction ~

Fin

Syntaxe de l'appel d'une fonction :

Variable ← NomFonction (NomEntrée1, NomEntrée2...)

Exemples d'appels de fonctions et procédures :

Procédure sans paramètre : Ex : Effacer_Ecran

Procédure avec un paramètre d'entrée : Ex : Afficher ('Bonjour')

Fonction avec paramètres d'entrée et de sortie : Ex : Resultat ← Racine (69)

Exemple de déclaration de fonction :

Fonction Moyenne (Note1 : Reel, Note2 : Reel) : Reel

Variable Intermediaire : Reel

Début

Intermediaire ← Note1 + Note2

Intermediaire ← Intermediaire / 2

Moyenne ← Intermediaire

Fin

Exemples d'utilisation de la fonction :

Afficher (Moyenne(10,5,15)) ou NouvelleNote ← Moyenne (10,5,5)

5) LES STRUCTURES ALGORITHMIQUES.

Les structures algorithmiques sont réparties en 3 catégories :

- succession linéaire d'opérations;
- structures conditionnelles ou de choix : en fonction d'une condition, le programme exécute des opérations différentes;
- structures itératives ou répétitives: sous contrôle d'une condition, une séquence d'opérations est exécutée répétitivement.

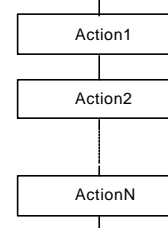
5.1) Séquencement linéaire

Les actions successives sont mentionnées les unes après les autres.

Syntaxe

Action1
Action2
...
ActionN

Algorithme



Remarque : dans la suite, la notation « Actions » ou « ActionsN » représentera une succession d'actions comme ci-dessus.

Exemple : Calcul d'un produit de 2 nombres

Variable

a,b : réel ~ opérandes ~
p : réel ~ résultat du produit ~

Début

Afficher ('Saisir le nombre a ')
Saisir (a)
Afficher ('Saisir le nombre b ')
Saisir (b)
 $p \leftarrow a * b$
afficher (p)

Fin

5.2) Structures de choix (ou conditionnelles)

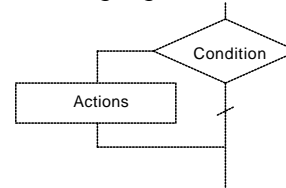
5.2.1) Structure SI ... ALORS ...

Une condition est testée pour déterminer si l'action ou le groupe d'actions suivant doit être exécuté.

Syntaxe

Si Condition
Alors Actions
FinSi

Algorithme



ALGORITHMIQUE

Exemple : Calcul d'une racine carrée

Variable

x: réel ~ opérande ~
r: réel ~ résultat de la racine carrée ~

Début

Afficher ('Saisir le nombre x')

Saisir (x)

Si $x > 0$

Alors

$r \leftarrow$ racine (x)

afficher (r)

FinSi

Fin

5.2.2) Structure SI ... ALORS ...SINON ...

Une condition est testée pour déterminer quelle action ou quel groupe d'actions doit être exécuté.

Syntaxe

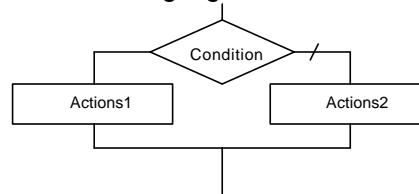
Si Condition

Alors Actions1

Sinon Actions2

FinSi

Algorithme



Exemple : Calcul d'une racine carrée

Variables :

x: réel ~ opérande ~
r: réel ~ résultat de la racine carrée ~

Début

Afficher ('Saisir le nombre x')

Saisir (x)

Si $x < 0$

Alors

afficher ('x est négatif')

Sinon

$r \leftarrow$ racine (x)

afficher (r)

FinSi

Fin

5.2.3) Structure de choix multiple

Une donnée est comparée successivement à des valeurs constantes :

Syntaxe

Cas Ou Donnée **Vaut**

Valeur1 : Actions1

Valeur2 : Actions2

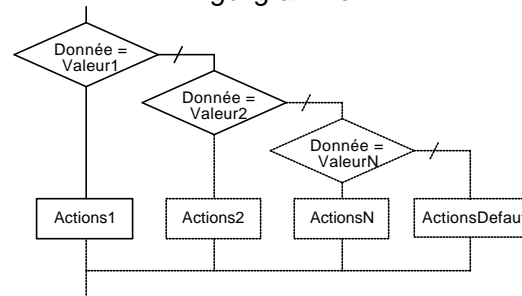
...

ValeurN : ActionsN

Autre : ActionsDéfaut

FinCas

Algorithme



Remarques : la partie « ActionsDéfaut » peut ne pas exister.

Plusieurs valeurs différentes peuvent être regroupées sur une même ligne si les actions correspondantes sont identiques.

Exemple : Affichage de la nature d'un caractère

Variable

c: caractère ~ caractère saisi au clavier ~

Début

Afficher ('Taper un caractère')

Saisir (c)

Cas Ou c Vaut

'A'..'Z' : afficher ('Lettre majuscule')

'a'..'z' : afficher ('Lettre minuscule')

'0'..'9' : afficher ('Chiffre')

autre : afficher ('Ni Lettre Ni Chiffre')

FinCas

Fin

5.3) Structures itératives (ou répétitives)

5.3.1) Structure REPETER ... JUSQUA ...

Une action ou un groupe d'actions est exécuté répétitivement jusqu'à ce qu'une condition soit vérifiée.

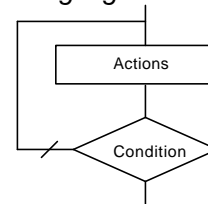
Syntaxe

Répéter

Actions

Jusqu'à Condition

Algorithme



Remarque : la vérification de la condition s'effectue **après** les actions. Celles-ci sont donc exécutées au moins une fois.

Exemple : exécution répétitive d'un programme

Variables :

a,b : réel ~ opérandes ~

p : réel ~ résultat du produit ~

c : caractère ~ réponse de l'utilisateur ~

Début

Répéter

Afficher ('Saisir le nombre a')

Saisir (a)

Afficher ('Saisir le nombre b')

Saisir (b)

$p \leftarrow a * b$

afficher (p)

afficher ('encore un calcul ? Non touche N ; Oui autre touche')

saisir (c)

Jusqu'à c = 'N'

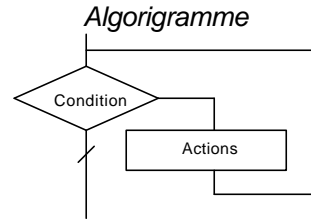
Fin

5.3.2) Structure TANT QUE ... FAIRE ...

Une action ou un groupe d'actions est exécuté répétitivement tout le temps où une condition est vraie.

Syntaxe

TantQue Condition
Faire Actions
FinFaire



Remarque : la vérification de la condition s'effectue **avant** les actions. Celles-ci peuvent donc ne jamais être exécutées.

Exemple : Exécution répétitive d'une action

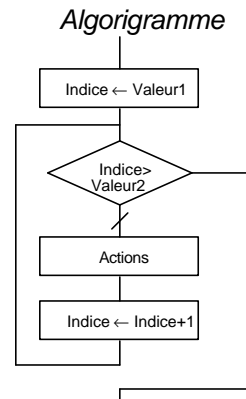
Début
Tant Que Non (ToucheAppuyée)
Faire Afficher ('Attente')
FinFaire
Fin

5.3.3) Structure POUR Indice ALLANT DE ... A FAIRE ...

Une action ou un groupe d'actions est exécuté répétitivement un certain nombre de fois : le nombre dépend des valeurs initiale et finale données à la variable « Indice ».

Syntaxe

Pour Indice **Allant De** Valeur1 **A** Valeur2
Faire Actions
FinFaire



Remarque : les valeurs initiale (Valeur1) et finale (Valeur2) sont comprises. Il est éventuellement possible de spécifier un autre pas d'incréméntation (+2,+10,-1....)

Exemple : Affichage d'une ligne d'étoiles
Variable
i : entier ~ compteur de boucles ~
Début
Pour i **allant de** 1 **à** 80
Faire Afficher ('*')
FinFaire
Fin

ALGORITHMIQUE

Remarque : cette structure algorithmique peut en fait être remplacée par une structure TANT QUE ... FAIRE ...

Variable

i : entier ~ compteur de boucles ~

Début

i ← 1

Tant Que $i \leq 80$

Faire

Afficher (ⁱ*)

i ← i + 1

FinFaire

Fin