PROGRAMME

Chapitre I : Notions de base

Chapitre II : Structures de contrôle ou Structures conditionnelles

Chapitre III : Structures itératives ou Structures répétitives ou Boucles

Chapitre IV ; Structures Composées ou Types composés

Chapitre V : Chaînes de caractères

Chapitre VI : Tableaux

Chapitre VII : Sous-programmes

Chapitre VIII : Fichiers

Chapitre IX : Pointeurs

Chapitre X : Listes chainées

Chapitre XI : Files d'attente

Chapitre XII : Piles

Chapitre XIII : Arbres

Chapitre I : *NOTIONS DE BASE*

I. DEFINITION

Un algorithme est une suite d'instructions ayant pour but de résoudre un problème donné et ses instructions doivent être exécutées de façon automatique par un ordinateur. Généralement, un algorithme reçoit puis manipule des données suivant un traitement bien défini pour fournir des données en sortie.

II. STRUCTURE D'UN PROGRAMME ALGORITHMIQUE

La structure d'un programme algorithmique est composée de 3 parties :

- ⇨ ***La partie entête*** *permet de nommer le programme. Elle est caractérisée par le mot clé PROGRAMME.*
- ⇨ ***La partie déclarative p****ermet de déclarer l'ensemble des données que nous utilisons dans le programme. Ces données peuvent être : des variables, des types et des prototypes ou des modules. Cette partie est caractérisée par des mots clés comme VAR, CONST, TYPE,…*
- ⇨ ***Le corps du programme : c****ette partie contient l'ensemble des instructions à suivre étape par étape, du début à la fin du programme destiné à être exécuter par un ordinateur. Cette partie commence par le mot clé DEBUT et se termine par le mot clé FIN.*

1. Les identificateurs

Les données que le programme manipule sont nommées. Ainsi donc, dans un même programme, il y'a risque de confusion si deux données ont les même noms. Pour éviter cette confusion, chaque donnée a son propre nom qui lui est unique (identificateur).

Un identificateur permet de reconnaitre (distinguer) chaque entité du programme de manière unique durant l'exécution du programme.

Pour créer un identificateur, il faudra respecter les règles suivantes :

- ✓ ***Règle 1 :***
  Le nom de l'identificateur doit être composé que de caractères alphanumériques.
- ✓ ***Règle 2 :***
  Le nom de l'identificateur ne doit pas commencer par un caractère numérique.
- ✓ ***Règle 3 :***
  Si le nom de l'identificateur est un nom composé, ces composants ne peuvent être séparés que par le tiret du 8 (_) et non par un espace ni le tiret du 6 (-)
- ✓ ***Règle 4 :***
  Le nom de l'identificateur ne doit pas se figurer dans la liste des mots réservés au langage algorithmique.
- ✓ ***Règle 5 (facultative) :***
  Le nom de l'identificateur doit être en rapport avec son contenu.

Exercice d'application :

Soient les identificateurs suivants, en se basant sur les règles, dites ceux qui sont corrects ou pas.

2. Notion de constante

Une constante, comme son nom l'indique est un identificateur dont la valeur ne change pas.
L'identificateur garde la même valeur durant toute l'exécution du programme.

Syntaxe :

Const identificateur = valeur *ou* ***Constante identificateur = valeur***

Exemple :

Const TVA = 0.18 (valeur réelle), *Const sexe = 'F' (valeur caractère).*

Const ecole = '' ISI '' (valeur chaine de caractères), *Const prix = 250 (valeur entière)*

Remarque :

Une constante ne peut être affectée que des valeurs de type entier, réel, chaine de caractères, caractère ou booléen.

3. Notion de variable

Une variable est un identificateur dont la valeur peut changer durant l'exécution du programme.

Syntaxe :

Var identificateur(s) : type *ou* ***Variable identificateur(s) : type***

Exemple :

Remarque :

Si plusieurs variables ont le même type, alors on peut les déclarer sur une même ligne en séparant les identificateurs par des virgules (Exemple : Var x, y, z : entier).

4. Notion de type

L'algorithme est un langage typé, c'est-à-dire que chaque donnée est rattachée à un type bien défini (sa nature).
Chaque variable doit avoir un type.

Les types sont regroupés en deux grandes familles :

- ✓ *les types simples*
- ✓ *les types composés.*

4.1. Les types Simples

On les appelle aussi types de base ou types primitifs et sont au nombre de cinq (5) :

- ***Entier** :*
  Il s'agit des variables destinées à contenir les nombres entiers
- ***Réel** :*
  Il s'agit des variables destinées à contenir les nombres décimaux
- ***Caractère** :*
  Ce sont des variables qui contiennent des valeurs alphabétiques ou des chiffres de 0 à 9 ou des caractères spéciaux
- ***Chaine de caractères** :*
  C'est une suite de caractères.
- ***Booléen** :*
  Ce sont des variables contenant les valeurs VRAI ou FAUX.

4.2. Les types composés

Ce sont les structures composées (voir chapitre 4).

III. SEQUENCE D'INSTRUCTIONS

Une instruction est une commande que l'ordinateur est capable d'exécuter. Elle permet de donner des directives qui sont destinées aux périphériques d'entrée ou de sortie.

1. Les instructions d'entrée

Elle permet d'entrer des données dans l'ordinateur en utilisant des périphériques d'entrée comme le clavier ou la souris.

Syntaxe :

Saisir (variable(s)) ***ou*** ***Lire (variable)***

L'instruction d'entrée se fait toujours avec des variables qui sont déjà déclarées dans la partie déclarative.

2. Les instructions de sortie

Elles permettent de donner des tâches à exécuter aux unités de sorite, essentiellement à l'écran et à l'imprimante.

Pour envoyer des flux de données vers l'écran, il faut utiliser les fonctions Afficher () ou Ecrire ().

2.1. A l'écran

Syntaxe :

Afficher (<Expression>) ***ou*** ***Ecrire (<Expression>)***

Remarque :

L'expression peut représenter :

1. *Une valeur,*
2. *Une variable,*
3. *Une Expression mathématique à évaluer,*
4. *La combinaison de (1), (2), ou (3)*

Exemple 1 *: Ecrire un programme qui affiche « Bonjours tout le monde » sur l'écran.*

Exemple 2 *: Ecrire un programme qui demande à l'utilisateur de saisir une expression chaine de caractères et d'afficher la valeur saisie.*

2.2. A l'imprimante

Pour envoyer des données vers une imprimante, il faut utiliser la fonction ***imprimer ().***

Syntaxe :

Imprimer (<Expression>)

IV. LES OPERATEURS LOGIQUES ET ARITHMETIQUES

Une opération est l'association d'opérandes (valeurs) et d'opérateurs (signes). Une opération est soit unaire (constituée d'un opérande et d'un opérateur), soit binaire (constituée de deux opérandes et d'un opérateur).

L'ordinateur dispose d'une unité appelée UAL (composante du processeur chargée d'exécuter toutes les opérations logiques de l'ordinateur)

1. Les opérateurs arithmétiques

Tous les opérateurs de base en mathématiques restent valables en informatique. Ces opérateurs sont définis dans le tableau ci-dessous.

Variables ou Valeurs		*Opérateurs*		*Opérations*		*Exemple*
		Signe	*Sens*	*Unaire*	*Binaire*	
A	*B*	*+*	*Addition*	*Oui*	*Oui*	*+A ; B+ ; A+B ; B+A*
		-	*Soustraction*	*Oui*	*Oui*	*-A ; -B ; A-B ; B-A*
		*	*multiplication*	*Non*	*Oui*	*A*B ; B*A*
		/	*Division réelle*	*Non*	*Oui*	*A/B ; B/A*
		div	*Division entière*	*Non*	*Oui*	A *div B* B *div A*
		mod	*Reste de la division entière*	*Non*	*Oui*	A *mod B* B *mod A*
		sqr	*Carré*	*Oui*	*Non*	*sqr(A) ; sqr(B)*
		sqrt	*Racine carrée*	*Oui*	*Non*	*sqrt(A) sqrt(B)*
		abs	*Valeur absolue*	*Oui*	*Non*	*abs(A) abs(B)*

2. Les opérateurs logiques

Ils sont utilisés au niveau des conditions. L'évaluation d'une condition est booléenne, c'est-à-dire qu'elle est soit vérifiée (vraie = 1) ou non vérifiée (fausse = 0).

Les opérateurs logiques de base sont la négation, le OU logique et le ET logique.

2.1. La négation

Soit X une condition ou variable logique.

X est vérifié c'est-à-dire X = VRAI ou X = 1 ; X n'est pas vérifié c'est-à-dire X = FAUX ou X = 0.

TABLE DE VERITE DE LA NEGATION	
X	***Négation de X = NON (X) = !X***
0	*1*
1	*0*

2.3. Le OU logique

*Si une condition globale est composée de plusieurs sous-conditions séparées par des **OU** logiques, alors la condition globale est Vérifiée si **au moins une des sous-conditions est vraie**.*

***Exemple** : Soient les sous-conditions A et B*

A	***B***	***A OU B***
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*1*

2.4. Le ET logique

*Si une condition globale est composée de plusieurs sous-conditions séparées par des **ET** logiques, alors la condition globale est Vérifiée si **toutes les sous-conditions est vraie**.*

***Exemple** : Soient les sous-conditions A et B*

A	***B***	***A ET B***
0	*0*	*0*
0	*1*	*0*
1	*0*	*0*
1	*1*	*1*

V. LES OPERATEURS DE COMPARAISON

Opérateurs		***Exemples***
Signe	***Sens***	
=	*Egalité*	*A =B*
<	*Inférieur*	*A<B*
>	*Supérieur*	*A>B*
<=	*Inferieur ou égale*	*A<=B*
>=	*Supérieur ou égale*	*A>=B*
!= ou <>	*différent*	*A != B*

Ces opérateurs de comparaison sont les mêmes qu'en mathématiques.

VI. OPERATEUR D'AFFECTATION

Syntaxe :

Variable ← <Expression>

Elle permet d'affecter une valeur à une variable. Elle se fait en utilisant le symbole : ←

NB :

L'expression peut être :

- *Une valeur,*
- *Une variable déjà initialisée,*
- *Une expression mathématique à calculer*