Initiation à l'algorithmique

I- Structure générale d'un algorithme

1.Schéma général d'un algorithme

Un algorithme comporte généralement deux parties :

* Partie déclarative : elle contient l'entête, la déclaration des constantes et celle des variables.

* Partie corps de l'algorithme : elle consiste en une séquence d'actions faisant appel à des opérations de base de l'ordinateur.

Syntaxe :

Algorithme « nom de l'algorithme »

Const

« Liste des constantes avec leurs valeurs » **Partie déclarative**

Var

« Liste des variables suivies par leurs types »

Début

« Séquence d'actions » **Fin**

Une action peut être :

- Action d'affectation ou
- Action d'entrée- sortie ou

*Partie corps de l'algorithme

Action de contrôle conditionnelle simple ou à choix multiple ou Action de répétition.

2.Définition d'une variable

Une variable est un emplacement mémoire capable de contenir des valeurs de type défini au préalable. Elle peut être définie comme une boite qui admet un nom, une taille, un contenu et une adresse.

Le nom de la variable s'appelle identificateur de la variable.

La taille dépend du type de la variable (exemple : 2 octets pour un entier, 1 octet pour un caractère, 4 octets pour un réel…)

L'adresse désigne le numéro du 1er octet occupé par cette variable en mémoire centrale
Dans un algorithme, les variables sont déclarées comme suit :

Var

Liste des variables suivies par des virgules : type 1

Liste des variables suivies par des virgules : type 2

. .

Exemple :

Var

X, Y : entier

A : réel

3. Définition d'une constante

La définition d'une constante est la même que celle d'une variable à la différence que la valeur d'une constante reste inchangée tout au long de l'algorithme.

Syntaxe :

Const

Nom const 1 = val 1

Nom const i = val i

Exemple:

Const

Min = 10

Max = 200

4. Les types de base

A toute variable est attribué un type qui définit :

- L'ensemble des valeurs que peut prendre la variable
- L'ensemble des opérations qu'on peut appliquer sur la variable

Il existe des types simples qui sont prédéfinis tels que les types : entier, réel, caractère ou booléen.

a) Type entier

- Il représente l'ensemble des entiers relatifs tel que : 8, -10, 3……
- Les opérations permises sont : +, -, *, div (division entière) et mod (reste de la division entière)

b) Type réel

- Il représente l'ensemble IR
- Deux formes de représentation : La forme usuelle « a.b » exemple : -4.6, 13.9 ….. ou la forme scientifique a E b exemple : 345 = 3.45 E2 = 0.345 E3
- Les opérations permises sont : +, -, *, /

c) Type caractère

- Il peut être une lettre, un chiffre ou caractère spécial exemple : 'a', 'b', '3' ….
- Les opérations permises : =, ?, <, <=, >, >=.

d) Type booléen

- Il représente les deux valeurs ‘Vrai’ et ‘Faux’
- Les opérations : NON, ET, OU

Remarque : Il existe des types composés définis à partir des types de base comme les tableaux, les chaînes de caractère….

II-Les traitements séquentiels

1- Instruction d’affectation

Cette action permet de ranger une nouvelle valeur dans une variable

Syntaxe :

Identificateur var ? <expression>

- Expression peut être :

- Une constante
- Une expression arithmétique
- Une expression logique

Remarque

- Une constante ne peut jamais figurer à gauche d’une affectation.
- Après une affectation, l’ancien contenu est perdu pour être substitué par le nouveau contenu.
- Une action d’affectation doit se faire entre deux types compatibles.

1. 1.Les expressions arithmétiques

<exp-arith> op_arith <exp-arith>

- Op_arith peut être ‘+’, ‘-‘, ‘/’ ou ‘*’ Exemple : (Y/2) + x*3
- L’ordre de priorité des opérateurs arithmétiques :
- signe négatif

( ) parenthèses

^ puissance

* et / multiplication et division

+ et – addition et soustraction

1. 2.Les expressions logiques

- Les expressions logiques admettent Vrai ou Faux comme résultat.
- Elles peuvent utiliser des opérateurs relationnels (= , ?, <,<=, >, >=) ou des opérateurs logiques (NON, ET, OU)
- L’ordre de priorité est :

NON
ET
OU

2- Instruction de lecture ou d'entrée

- Elle permet d'affecter, à une variable, une donnée introduite) partir d'une périphérique d'entrée (clavier).
- Syntaxe :

Lire (nom_var1, nom_var2,……)

- Exemple :

Lire(A) : lit une valeur à partir du périphérique d'entrée et la range dans la case mémoire associée à A.

Lire(X,Y) : lit deux valeurs la première pour X et la deuxième pour Y.

3- Instruction d'écriture ou de sortie

- Elle permet d'afficher des résultats sur un périphérique de sortie (écran). Ce résultat peut être :

- Une chaîne de caractères délimitée par des '' ''
- La valeur d'une variable dont le nom est spécifié

III- Les structures conditionnelles

Introduction

En programmation, on est souvent confronté à des situations ou on a besoin de choisir entre 2 ou plusieurs traitements selon la réalisation ou non d'une certaine condition d'ou la notion de traitement conditionnel. On distingue deux structures de traitement conditionnel à savoir :

- La structure conditionnelle simple qui consiste à évaluer une condition (expression logique à valeur vrai ou faux) et d'effectuer le traitement relatif à la valeur de vérité trouvée.
- La structure conditionnelle à choix multiple qui consiste à évaluer une expression qui n'est pas nécessairement à valeur booléenne (elle peut avoir plus de deux valeurs) et selon la valeur trouvée, effectue un traitement.

1- Structure conditionnelle simple

1.1.Forme simple : (Si …. Alors ….. Finsi)

Syntaxe :

Si condition
Alors
action(s)
Fin si

Dans cette forme, la condition est évaluée. Si elle vaut vrai alors c'est la séquence d'actions qui est exécutée sinon c'est l'action qui suit l'action conditionnelle dans l'algorithme qui est exécutée.

1.2.Forme composée : (Si … alors…….sinon)

Syntaxe :

Si condition **Alors**
Action(s)1
Sinon
Action(s)2
Fin si

Dans cette forme, la condition est évaluée. Si elle vaut vrai alors c'est la séquence d'actions 1 qui sera exécutée sinon c'est la séquence d'actions 2 qui sera exécutée. L'exécution de cette instruction se déroule selon l'organigramme suivant :

1.3.Forme imbriquée

Syntaxe :

Si condition 1 **Alors**
Action(s)1
Sinon
Si condition 2 **Alors**
Action(s)2
Sinon
Si condition N-1 **Alors**
Action(s)N-1
Sinon
Action(s)N
Fin si

Si la condition est vraie, alors la séquence d'actions 1 sera exécutée sinon on évalue la condition 2 si elle est vraie la séquence d'actions 2 sera exécutée. Enfin, si aucune des N-1 conditions est vraie alors on exécute la séquence d'actions N.

2- Structure conditionnelle à choix multiple

Syntaxe :

Selon <sélecteur> faire

<liste de valeurs1> : <traitement 1>

<liste de valeurs2> : <traitement 2> …..

…..

<liste de valeursN> : <traitement N>

Sinon

<traitement N+1>

Fin selon

- Le sélecteur est un identificateur
- <traitement i> est une séquence d'actions.

<liste de valeurs i> peut être une constante ou un intervalle de constantes de même type que sélecteur.

IV- Les structures itératives

Introduction

On peut exécuter une action ou un ensemble d'actions non pas infiniment mais un certain nombre de fois : c'est la notion de boucles.

1- La structure « Pour ….. faire…….Finpour »

Syntaxe

Pour vc de vi à vf faire Traitement

Finpour

* Vc : compteur de type entier
* Vi et vf : valeur initiale et valeur finale de vc
* Traitement : action ou séquence d'actions à répéter (vf-vi +1) fois.
* La boucle Pour est utilisée lorsque le nombre d'itération est connu à l'avance.

Vc reçoit une valeur initiale vi pour la première fois, il ne doit pas être modifié par une action de traitement à l'intérieur de la boucle.

Vc est incrémenté automatiquement par 1 à chaque exécution du corps de la boucle Pour.

Cette valeur d'incrémentation est appelée le pas de la boucle.

L 'exécution de la boucle finit lorsque vc atteind vf.

Schéma d'exécution d'une boucle « Pour »

Exemple :

Pour i de 1 à 5 faire

Ecrire (i * 100)

Fin pour

Exécution : i = 1 2 3 4 5 6

Résultat 100 200 300 400 500

Remarques :

Une boucle peut être exécutée une ou plusieurs fois.

Si le pas est différent de 1, il faut ajouter l'option (pas = constante)

2- La structure « Répéter …. Jusqu'à »

Syntaxe :

Répéter

Traitement

Jusqu'à (condition)

- Condition : condition d'arrêt et de sortie de la boucle
- Traitement : action ou ensemble d'actions à exécuter tant que la condition n'est pas vérifiée, dés qu'elle soit vérifiée, l'exécution du traitement s'arrête.
- Le nombre de répétition n'est pas connu à l'avance.
- Le traitement est exécuté au moins une fois quelque soit le résultat de la condition.
- La condition doit être initialisée avant le début de la boucle et doit être modifiée à l'intérieur de la boucle.

3- La structure « Tantque……. Faire…….. Fintantque »

Syntaxe :

Tantque (condition) **faire** Traitement

Fintantque

- Condition : condition de maintien de la boucle.
- Traitement : Action ou ensemble d'actions à exécuter tant que la condition est vérifiée.
- Le traitement est exécuté tant que la condition est vérifiée sinon on sort de la boucle.
- Si la condition n'est pas vraie dés la première exécution, la boucle ne sera jamais exécutée (0 fois).
- Le nombre de répétition n 'est pas connu à l'avance.