Support de cours

Chapitre 2 : Algorithme séquentiel simple

1) Notion de langage et le langage algorithmique

L'ordinateur est une machine électronique programmable mais ne peut exécuter que des instructions au format binaire (0/1). Les langages de programmation servent d'interprètes entre la machine et l'être humain. Les programmes sont écrits avec un langage respectant des règles bien précises avec un vocabulaire bien défini qui sont traduits ensuite par un compilateur en langage binaire pour être exécuter par l'ordinateur.

a) **La compilation :** permet de traduire un programme écrit en langage de programmation dit '***langage évolué***' en une suite d'instructions exécutables par la machine. un langage évolué est très proche du langage humain.

b) **Le langage Algorithmique** : comme les langages de programmation, le langage algorithmique utilise un vocabulaire et respecte des règles d'écriture universelles. Contrairement au langage de programmation, l'algorithmique est très proche des langages humains. Un algorithme est la formalisation de la solution d'un problème par l'être humain et il n'est destiné pour être interpréter par la machine. Alors l'algorithmique n'impose pas le respect stricte de sa syntaxe.

2) Structure d'un algorithme

Un algorithme est une représentation formelle de la solution d'un problème en utilisant le vocabulaire du langage '*algorithmique'.* Les mots du vocabulaire algorithmique sont appelés '**mots clés**'. Un algorithme est composé de trois parties :

a) **L'entête** : permet de donner un nom à l'algorithme à travers un identificateur

b) **La partie déclaration** : pour définir tous les objets nécessaires dont la machine aura besoin pour résoudre le problème.

c) **La partie actions** : contient les instructions à exécuter pour solutionner le problème. La partie actions est délimitée par les mots clés '**Debut'** et '**Fin'**

```
Algorithme nom_algorithme ;        *Entête*

Déclarations                       contient les données nécessaires à la résolution du problème.
Debut
   Instruction 1 ;
   Instruction 2 ;                 contient exclusivement des instructions.
   …
   …
Fin.
```

Support de cours

NB : chaque déclaration et chaque instruction doit se terminer par un ' **;**'

3) Les données : variables et constantes (la partie déclaration)

a) Variables : déclarer une variable dans un algorithme consiste à réserver un espace au sein de la mémoire centrale (RAM) pour être utilisé dans l'algorithme. Une variable déclarée dans un algorithme prend immédiatement une valeur par défaut. La valeur d'une variable est modifiable dans la partie action de l'algorithme.

b) Constantes : les constantes sont utilisées pour simplifier la programmation. Une constante reçoit sa valeur au moment de sa déclaration et elle n'est pas modifiable. Après compilation d'un programme, les constantes sont remplacées par leurs valeurs.

Déclaration :

```
VAR identificateur : type ;
CONST identificateur = valeur ;
```

4) Identificateurs : Un identificateur est un nom utilisé pour nommer (identifier) un algorithme, une variable, une constante et le type. Un identificateur est une chaine de caractères composé uniquement des caractères alphanumériques (alphabétiques non accentuées (A..Z, a..z) et numériques (0-9)) et du caractère tiret 8 (_). Il doit respecter les règles d'écriture ci-dessous :

- Ne doit pas commencer par un chiffre.
- Ne doit pas être un mot clés.
- Ne contient pas les caractères spéciaux, ni le caractère espace.

Exemples :

a) Les types élémentaires : il est nécessaire de préciser le type d'une variable pour savoir quelle est le nombre d'octets à attribuer à la variable. On utilise quatre types de base prédéfinis en algorithmique :

- Le type ***entier*** sera utilisé pour stocker des valeurs entières, positives ou négatives.
- Le type ***réel*** sera utilisé pour stocker les nombres à virgule.
- Le type ***caractère*** sera utilisé pour stocker les caractères. Les caractères sont un ensemble de symboles comprenant les 26 lettres de l'alphabet latin en versions

minuscules et majuscules, les 10 chiffres de 0 à 9, les différents symboles de ponctuation, l'espace, et bien d'autres symboles encore. Ils sont au nombre de 256 au total.

- Le type ***booléen*** sera utilisé pour stocker les valeurs de type vrai/faux.

En plus de ces quatre types simples, nous avons un autre type **composé** :

- Le type ***chaîne de caractères*** sera utilisé pour stocker des séquences finies de caractères. Le nombre de caractères composant une chaîne de caractères est la longueur de cette chaîne.

Type	**Mot clef**	**Exemple**
Entier	entier	36, -25
Réel	Réel	6.5
Booléen	booleen	Vrai, faux
Caractère	carctere	'b', '7'
Chaîne de caractères	Chaine[nombre de caractères]	'omar'

Exemples :

NB : il n'est pas nécessaire de répéter les mots clés 'VAR' et 'CONST' pour chaque déclaration.

b) Définition de nouveau type : il désormais possible de définir des nouveaux type inexistant en algorithmique avec le mot clés '**Type'**.

i) **Le type intervalle** : Un intervalle est défini avec sa borne inferieur et sa borne supérieur. Le type des valeurs contenues dans l'intervalle est détecté automatiquement à partir de ses bornes.

Syntaxe :

```
Type id_type_inter = borne_inf .. borne_sup ;
```

Exemples:

Support de cours

ii) **Les types énumérés :** parfois il est utile de déclarer une variable qui prend ses valeurs dans un ensemble fini.

Syntaxe :

```
Type nom_type = (valeur1, valeur2, . . . , valeurn) ;
```

5) Opérations de base (la partie action)

a) Instruction d'entrées : permet de récupérer des valeurs saisies au clavier et de les sauvegarder dans des variables.

Syntaxe :

```
Lire (variable1, variable2,…, variablen) ;
```

L'instruction peut lire une ou plusieurs variables à la fois.

Exemple : `lire(N) ;`

A l'exécution de l'instruction lire(N), le programme se met en attente jusqu'à ce que l'utilisateur saisie une valeur au clavier et valide par la touche entré. La valeur saisie est sauvegardée ensuite dans la case mémoire réservée pour la variable N.

b) Instruction de sortie : permet la communication avec l'utilisateur en affichant à l'écran les résultats de l'algorithme qui peuvent être le contenue des variables ou de simple message.

syntaxe :

```
ecrire(variable);
ecrire(constante);
```

l'instruction `ecrire` peut afficher plusieurs variables et plusieurs constantes en même temps séparées par des virgules;

exemple :

`ecrire(N) ;` affiche la valeur de la variable N.

`ecrire(pi) ;` affiche la valeur de la constante pi.

`ecrire(x,y,z) ;` affiche les valeurs des variables x, y, et z.

`ecrire('Bonjour') ;` affiche le message *Bonjour*

Support de cours

c) L'affectation :

l'instruction d'affection permet de modifier le contenu d'une variable. Elle symbolisée par le caractère ←

syntaxe :

variable ← expression ;

- Expression peut être une constante, une variable ou une expression qui donne un résultat de type compatible avec la partie gauche de l'affectation.
- La partie gauche de l'instruction d'affectation doit être une variable.

exemples :

6) Les opérateurs : en algorithmique les traitements arithmétique et logique sont effectués en utilisant des opérateurs. On distingue deux types d'opérateurs :

a) Les opérateurs arithmétiques : sont applicables sur des donnée de type nombre (entier, réel).

+ addition

- soustraction

* multiplication

/ division réelle

`div` division entière, **ex :** `20 div 3 = 6;`

`mod` modulo, il fournit le reste de la division, **ex :** `20 mod 3 = 2 ;`

b) Les opérateurs logique : sont applicables sur des données de type bouleen. On distingue : NON, ET, OU.

A	B	NON(A)	A ET B	A OU B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Support de cours

c) **Les opérateurs relationnels :** ils sont utilisés pour comparer les valeurs de deux variables. Ils produisent un résultat booleen (vrai/faux).

< inferieur, < = inférieur ou égal, > supérieur, >= supérieur ou égal, = égalité, < > différent.

7) **Evaluation des expressions** : une expression arithmétique ou logique est évaluée en respectant l'ordre de priorité des opérateurs. La machine effectue un seul calcul à la fois, de ce fait lorsqu'une expression contient plusieurs opérateurs de même priorité, l'opérateur le plus à gauche est évalué en premier.

l'ordre de priorité des opérateurs :

1- ( ). s'il y a plusieurs parenthèses, les plus internes sont prioritaires.

2- NON

3- ET, *, /, div, mod

4- OU, +, -

5- <, <=, >, >=, =, < >

Exemple :

E = 4 + 3 – 6 / 2 * 5

E = 4 + 3 – 3 * 5

E = 4 + 3 – 15

E = 7 – 15

E = – 8

8) **Exercice** : écrire un algorithme permettant de calculer la moyenne d'un étudiant dans le module algorithmique.