COMPILATION

STRUCTURE D'ALGORITHME

Réalise par :

EZ-ZAKKAR Mohammed

ELMOUMNY Azddine

AALLOUL Outmane

AMAACH Amine

Encadre par :

Mr QAZDAR AIMAD

SMI-S6

2019/2020

Langage algorithmique

1. Structure générale d'un algorithme

La présentation d'un algorithme sous forme de bloc est très proche du langage de programmation.

Sa structure est la suivante :

```
Algorithme <identificateur_nom> ; {En-tête}

VAR
                <identificateur> : <Type> ; {Partie déclarations}

Début
                <partie actions> ; {Corps de l'algorithme}

Fin.
```

Notons que chaque partie de l'algorithme possède des mots clés spécifiant la nature ainsi que l'étape de description.

1.1En-tête

L'en-tête d'un algorithme est de la forme suivante :

```
Algorithme <identificateur_nom>;
```

où *Algorithme* est un mot clé indiquant le début d'un algorithme et *identificateur_nom* c'est le nom donné par le programmeur. Généralement, on choisit un nom évoquant le rôle de l'algorithme.

1.2 Partie déclaration

Elle contient la déclaration de tous les objets manipulés (constantes et variables) par un algorithme. Elle associe à chaque objet un nom (identificateur), un type et éventuellement une valeur (pour les constantes)

Identificateurs :

C'est un nom que l'on attribue à toute entité manipulée dans un programme. Les identificateurs sont choisis librement, par l'utilisateur, toutefois ils obéissent à certaines règles :

1. Un nom doit commencer par une lettre et non par un chiffre. **Exemple 1** : d1 ou D1 et non 1D;

2. Doit être constitué uniquement de lettres, de chiffres et du soulignement (éviter les caractères de ponctuation et les espaces),

Exemple 2 : SM2013, USTHB et non SM 2013, U S T H B.

3. Doit être différent des mots clés réservés au langage (par exemple en Pascal: *Var*, *begin*, *sqrt*, *write* . . .).

Types de base (simples) :

1. **Le type entier :** un objet de type entier prend ses valeurs dans l'ensemble des entiers relatifs Z. En effet, un entier s'écrit comme suit : [+/−] *< Chiffre >.*

Exemple : +20, −10, 2013 sont des valeurs entières valides.

2. **Le type réel :** un objet de type réel prend ses valeurs dans l'ensemble R. En effet, un réel s'écrit comme suit : [+/−] <partie entière>. <Partie fractionnaire> *E* <entier> ; *E* désigne la puissance de 10.

Exemple : +12.05, −0.05 ≡−5*E* − 2, −2.5 × 10^5 ≡−2.5*E* + 5.

4. **Le type caractère :** un objet de type caractère prend ses valeurs dans l'ensemble des caractères alphabétiques minuscules et majuscules, numériques et caractères spéciaux ∗, +, /, ?, <, > *etc*. Les opérations qu'on peut appliquer sur le type caractère sont les opérations de comparaison (>, <, =, <=, etc.).

5. **Le type chaîne de caractères :** un objet de type chaîne de caractères est composé d'un ensemble d'objets de type caractère.
   Exemple: ''USTHB'' , ''Section'', ''SM2013''.

Déclaration des objets (constantes et variables) :

1. **La déclaration des constantes :** une constante est une variable dont la valeur ne change pas au cours de l'exécution du programme. Elle peut être un nombre, un caractère, ou une chaîne de caractères. Syntaxe :

```
CONST  <identificateur> = <valeur de la constante> ;
```

Exemple :

```
Constantes
```

Pi = 3.14 : Réel ;

Ln2 = 2.93 : Réel ;

2. **La déclaration des variables :** Syntaxe:

```
VAR <identificateur> : <type de la variable>;
```

Exemple 10 :

```
VAR
Prix : Réel ;
Nombre_etudiant : Entier ;
Nombre_groupe : Entier ;
Nom_section : CHAR;
Nom_etudiant[30] : char;
```


1.3 Partie instructions

Appelée aussi partie d'actions ; elle commence par le mot clé ***Début*** et se termine par ***Fin***. Ces deux bornes constituent deux parenthèses, l'une ouvrante et l'autre sortante, délimitant un bloc appelé corps de l'algorithme. Ce bloc regroupe toutes les instructions nécessaires pour traiter un problème donné.

Expression :

Une expression (située à droite de la flèche) peut être une valeur, une variable ou une opération constituée de variables reliées par des opérateurs. Il existent trois types d'expressions:

1. **L'expression de base :** appelée aussi élémentaire, représente les valeurs que peut prendre une constante ou une variable.

   Exemple : −10, 1, +50, *Vrai* et *Faux*.

2. **L'expression arithmétique :** elle est formée par des combinaisons d'objets numériques (entier et réel) et des opérateurs arithmétiques. Une expression arithmétique donne un résultat numérique dont le type est entier ou réel.

   Exemple : $a * 2$, $(a + 3) * b/c$, $k * q_1 * q_2/r * r$.

3. **L'expression logique :** elle est formée d'objets de type booléen et d'opérateurs logiques. Une expression booléenne donne un résultat booléen (vrai ou faux).

Exemple : $(X \leq Y)$ et B est une expression logique formée de deux variables booléennes.

Opérateur :

C'est un signe qui relie deux valeurs, pour produire un résultat. Il existe plusieurs types d'opérateurs.

1. **Les opérateurs arithmétiques :** Ils permettent le calcul de la valeur d'une expression arithmétique. Le tableau suivant montre quelques types d'opérateurs arithmétiques.

Symboles	Opérations
+ , -	Addition et soustraction
* , /	Multiplication et division
%	Modulo (reste de la division entière)

2. **Les opérateurs rationnels :** ils permettent de comparer deux valeurs de même type (numérique ou caractère) en fournissant un résultat booléen (vrai ou faux).

Symboles	Relation
>	Supérieur
<	Inférieur
>=	Supérieur ou égal
<=	Inférieur ou égal
=	Égal
<>	Différent

Exemple 14 : *Comp* ← *A* > *B*. *Comp* doit être de type booléen. *A* et *B* sont de même type.

---

3. **Les opérateurs logiques :** ces opérateurs sont appliqués à des données de type booléen et renvoient un résultat de type booléen.

Opérateurs logiques	Fonction
Et	Réalise une conjonction entre deux valeurs booléennes
Ou	Réalise une disjonction entre deux valeurs booléennes

Instructions de base (simples) :

1. **L'affectation:** elle consiste à attribuer une valeur à une variable (c'est-à-dire remplir le contenu d'une zone d'une mémoire). L'affectation est réalisée au moyen de l'opérateur =.

   Exemple : *X* = 5, signifie attribuer la valeur 5 à la variable *X*.

2. **L'opération d'entrée (lecture):** elle permet d'entrer des données à partir d'un périphérique d'entrée (clavier). Syntaxe:

```
Lire (Identificateur_variable)
```

Exemple :

5

```
Variables A, B : <type>
Lire (A) , Lire (B)  ou bien Lire (A,B) .
```

3. **L'opération de sortie (écriture):** Syntaxe :

```
Écrire (expression)
```

Exemple :

```
Ecrire (a)
Ecrire ('Section SM')
```

Structures conditionnelles et itératives

2.1 Structures conditionnelles

Ce sont des instructions qui permettent d'exécuter une ou plusieurs actions en fonction d'une condition.

Types d'instructions conditionnelles :

1. **L'instruction alternative simple (si . . . alors .. .)**

Syntaxe:

```
Si < Cond > Alors
    < Bloc d'instructions >
FinSi
```

Exemple :

```
Si (y<0) Alors
        Ecrire ('y est négatif')
FinSi
```

2. **L'instruction alternative multiple (si . .. alors .. . sinon)**

Cette instruction offre un choix entre deux possibilités, selon le résultat du test de la condition.
Syntaxe:

```
Si < Cond > Alors < Bloc d'instructions 1 >
                    Sinon < Bloc d'instructions 2 >
FinSi
```

Exemple : Afficher le signe d'un nombre entier x.

```
Si (x>= 0) Alors
      Ecrire(x, ' est positif')
Sinon
      Ecrire(x, ' est négatif')
FinSi
```

3. **L'instruction** ***Cas. . . Vaut. . .***

Elle permet de faire un choix parmi plusieurs cas possibles, suivant la valeur d'une expression. Syntaxe :

```
Cas <expression> vaut
    <valeur 1> : <Action 1>; Break
    <valeur 2> : <Action 2>;Break    .          .
                 .                    .
    <valeur n> : <Action n>Break
    Sinon : <Action>; Break
Fin cas ;
```

Exemple : L'affichage du libellé de la saison en connaissant le numéro du mois.

```
Cas numéro vaut
   01, 02, 12 : écrire ('hiver'); Break
   03, 04, 05 : écrire ('printemps'); Break
   06, 07, 08 : écrire ('été'); Break
   09, 10, 11 : écrire ('automne') Break
   Sinon : écrire ('numéro du mois incorrect')Break
Fincas;
```

2.2 Structures itératives

1. **La boucle tant que**

Syntaxe :

```
TantQue <Cond> Faire
            < Bloc d'instructions >
 FinTantQue
```

2. **La boucle répéter**

Syntaxe :

```
REP
        < Bloc d'instructions >
JUS  <COND>
```

3. **La boucle Pour** Syntaxe:

```
Pour i   valeur initiale '_ valeur finale Faire
                < Bloc d'actions >
FinPour
```

Tableaux, fonctions et procédures

3.1 Tableaux

3.1.1 Tableaux à une dimension (vecteurs)

Déclaration d'un vecteur :

La déclaration d'une variable de type tableau à une dimension (vecteur) se fait par :

```
VAR <identificateur_tab>[<size table>] :<Type> ;
```

Exemple :

```
VAR T : Tableau [100] : CHAR ;
```

3.1.2 Tableaux à deux dimensions (matrices)

Déclaration d'une matrice :

La déclaration d'une variable de type tableau à deux dimensions (matrice) se fait par l'une des deux syntaxes suivantes:

```
VAR <iden_mat> tableau [M][N] : <Type> ;
```

3.2 Fonctions et procédures

3.2.1 Notion de fonction

Déclaration d'une fonction :

La déclaration des fonctions se fait à la fin de la partie déclaration de l'algorithme principal et suit le format suivant :

```
<Fonction> <iden_fonction> : (arguments : type) : <Type du résultat renvoyé> Début
                                                    <Corps de la fonction>
                                                        Renvoyer =<expression>

                                                Fin ;
```

Exemple : Fonction qui calcule la somme de deux nombres.

Fonction Som (*A*, *B* : REEL) : REEL

Début

Som = *A* + *B*

Renvoyer Som ;

Fin.

3.2.2 Notion de procédure

Déclaration d'une procédure :

Syntaxe :

```
<Procédure> <iden_procédure> :(arguments : type) Début
                                    <Corps de la procédure>
                            Fin ;
```

Exemple 42 : Déclaration ou définition d'une procédure qui calcule le minimum de deux réels.

Procédure Minimum (*a*, *b* : Reel)

VAR min : ENTIER

Début

Si ($a \leq b$) alors

min = *a* sinon

min = *b*

Ecrire ('le minimum est ', min)

Fin.