

# **ALGORITHMIQUE ET PROGRAMMATION INFORMATIQUE**

## Chapitre 2 : Les éléments de Base d'un langage algorithmique

### I. Notion d'objet

Un algorithme manipule des informations (données) pour produire d'autres informations (résultats).

Une information contenue dans un algorithme est appelée **objet**. L'ensemble des objets manipulés par un algorithme est appelé **environnement** de l'algorithme.

Généralement à chaque objet correspond un emplacement en mémoire. Un objet est caractérisé son adresse, son identificateur, sa valeur, son type.

#### 1. adresse

C'est l'adresse de l'emplacement correspondant à l'objet.

#### 2. identificateur d'objet

L'identificateur est une suite de caractères alphanumériques non accentués, sans espace, commençant par une lettre. C'est nom donné à l'objet par le programmeur, permettant de sa manipulation. Le nom doit être unique et en rapport avec le contenu de l'objet.

#### 3. Valeur d'un objet

C'est le contenu de l'objet (c'est à dire une valeur de l'information) à un moment donné du déroulement de l'algorithme. Un objet ne peut prendre qu'une valeur à un instant donné.

Au cours de l'exécution de l'algorithme, certains objets peuvent changer de valeur: ce sont des **variables** ; d'autres objets au contraire ne seront pas modifiables. Ce sont des **constantes**.

La manipulation d'un objet concerne sa valeur.

#### 4. Type d'un objet

Le type d'un objet est la nature, la catégorie de l'objet. Le type d'un objet détermine:

- L'ensemble des valeurs que peut prendre l'objet.
- La taille de l'objet c'est à dire de l'emplacement correspondant.
- Les opérations applicables à l'objet

On distingue généralement deux catégories de types :

- Les types élémentaires (ou simples ou de base): un type est dit élémentaire si les actions qui manipulent un objet de ce type ne peuvent l'accéder que dans sa totalité.

- Les types structurés (ou complexes) qui sont définis à partir d'autres types et dont les objets peuvent être accédés en partie.

## II. Types de données de base

### 1. Type entier

Le type entier représente l'ensemble des entiers relatifs  $\mathbb{Z}$ .

Une constante littérale entière c'est à dire un élément du type entier est notée de manière classique: suite chiffres en base 10, précédée ou non des signes + ou - .

Les opérations applicables:

Opérations arithmétiques: -, +, -, \*, div, mod, ^

Opérations de comparaison: <, >, =, ≠, ≥, ≤

### 2. Type réel

Le type réel représente l'ensemble  $\mathbb{R}$  des réels. Les constantes littérales réelles s'écrivent:

- En notation décimale avec le point décimal : 5, 3.5, -10.67,...
- En notation exponentielle : -5.698e -12, 25e7,...

Les opérations applicables:

- Opérations arithmétiques: -, +, -, \*, /, ^
- Opérations de comparaison: <, >, =, ≠, ≥, ≤

### 3. Type booléen

Le type booléen représente l'ensemble composé des deux constantes **.vrai.** et **.faux.**

Les opérateurs applicables:

Opérations logiques: non, et, ou.

Opérations de comparaison : <, >, =, ≠, ≥, ≤

### 1. Type caractère

Le type caractère est l'ensemble formé des lettres (minuscules, majuscules), des chiffres et des symboles spéciaux (ponctuation, espace, ....).

Les constants caractères se notent entre simple quote (').

Chaque caractère est lié à une valeur entière qui le représente en mémoire.

Les opérateurs sur les caractères sont les opérateurs de comparaison

### III. Les Expressions

#### 1. Définition

Une expression est une suite bien formée d'opérateurs portant sur des objets (variables ou constantes), des constantes littérales, des appels de fonctions, et qui a une valeur.

Une expression représente en fait un calcul que l'ordinateur doit mener pour obtenir une valeur.

La valeur d'une expression est généralement utilisée: affichée, affectée, utilisée pour appeler un sous-programme. On peut utiliser des parenthèses dans une expression pour former l'ordre d'évaluation dans l'expression.

#### 2. Type d'une expression

Le type d'une expression est le type de sa valeur. Ainsi on distingue des expressions:

- entières (type entier) ;
- réelles (type réel) ;
- numériques (type réel ou entier) ;
- booléennes ou logiques (type booléen) ;

### IV. Structure de base

En matière de programmation, il n'y a pas grande chose d'obligatoire mais beaucoup de choses recommandées. En particulier, un programme a à peu près toujours la même organisation générale

1: Nom du programme
2: Déclaration de variables
3: Déclaration de fonctions
4: <b>Début</b>
5: ...
6: Liste des instructions
7: ...
8: <b>Fin</b>

### V. Instruction élémentaires

Une instruction est l'indication d'une action dont est capable l'ordinateur; il s'agit d'une instruction d'action c'est à dire d'ordre. Donc l'algorithme est en fait un ensemble d'instructions d'action sur des objets.

Chaque instruction donnera lieu à une action de la part de l'ordinateur.

## 1. Déclaration de variables

Définition de variable : Une variable est un espace mémoire nommée, de taille fixée prenant au cours du déroulement de l'algorithme un nombre indéfini de valeurs différentes. Ce changement de valeur se fait par l'opération d'**affectation** (notée dans notre langage algorithmique). La variable diffère de la notion de **constante** qui, comme son nom l'indique, ne prend qu'une unique valeur au cours de l'exécution de l'algorithme.

**Utilité de la déclaration de variable :** La partie *déclaration de variable* permet de spécifier quelle seront les variables utilisées au cours de l'algorithme ainsi que le type de valeur qu'elles doivent respectivement prendre. Il est bien évident que l'on ne peut mémoriser une chaîne de caractères dans une variable de type ``Entier''. Le type des variables est utile à l'algorithmicien pour lui permettre de structurer ses idées. Il est très utile au programmeur car il permet de détecter des erreurs potentielles. Il est indispensable au compilateur car les différents types ne sont pas tous représentés de la même façon. La représentation d'un même type peut même varier d'un processeur à l'autre.

### a) Syntaxe en LDA

#### i. Variable simple

##### Syntaxe

Variable <nom de donnée>: type

##### Exemple

- val, unNombre: entiers
- nom, prénom : chaînes de caractères

##### Fonctionnement

Instruction permettant de réserver de l'espace mémoire pour stocker des données de type <entier> et <chaîne de caractère> dont les noms sont respectivement <val > et <unNombre>

#### ii. Variable constante

##### Syntaxe

Constante <nom de donnée>: type ← valeur ou Expression

##### Exemple

- MAX : entier ← 10

- DEUXFOISMAX : entier  $\leftarrow$  MAX x 2

### Fonctionnement

Instruction permettant de réserver de l'espace mémoire pour stocker une constante dont la valeur ne varie plus. Ici les noms sont Max et DEUXFOISMAX avec les valeurs respectives 10 et 20.

### Remarque:

Tout objet ou variable avant d'être manipulé doit être déclaré.

## 2. Affectation

L'affectation consiste à placer une valeur dans une variable. Le contenu précédent de la variable est définitivement perdu.

### a) Syntaxe en LDA

<nom\_var>  $\leftarrow$  <expression>;

1. {vExemple d'affectation }

2. A : Entier

3. DEBUT

4. A  $\leftarrow$  1

5. FIN

{ Cette séquence d'instructions donne la valeur « 1 » à la variable A. }

{ Remarque : la valeur donnée est toujours du même type que la variable }

### b) Fonctionnement

L'instruction d'affectation s'exécute en deux étapes:

- Evaluation de l'expression <expression>
- Assignment du résultat dans la variable <nom\_var>

L'expression et la variable doivent être de types compatibles.

## 3. LECTURE ÉCRITURE DE DONNÉES

### a) Ecriture de données

L'écriture de données consiste à communiquer sur l'écran des résultats sous une forme directement compréhensible par l'utilisateur. Elle consiste en l'affichage d'un texte ou de la valeur d'une variable.

#### i) Syntaxe en LDA

Ecrire (<expression>);

#### ii) Fonctionnement

L'instruction va évaluer l'expression <expression> puis afficher le résultat à l'écran.

#### Exemple:

```
6. {Exemple d'utilisation de Ecrire }
7. A : Entier ← 5
8. DEBUT
9.     Ecrire (A+10)
10.    Ecrire ("La Valeur de A est :", A)
11.    Ecrire ("La Valeur de A+1 est :", A+1)
12. FIN
```

Cette séquence d'instructions affiche les phrases suivantes à l'écran:

```
15
La Valeur de A est 5
La Valeur de A+1 est 6.
```

#### Remarque:

On peut généraliser la syntaxe afin de permettre l'écriture de plusieurs expressions:

Ecrire(<exp1>, ..., <exp n>);

#### b) Lecture de données

La lecture consiste à lire (à récupérer) une information saisie depuis le clavier de l'ordinateur, à la convertir en binaire et à la ranger en mémoire centrale dans une variable.

L'instruction de lecture de données permet donc de communiquer des données à l'algorithme.

La saisie se fait par l'intermédiaire de la commande *Lire*.

#### i) Syntaxe en LDA

Lire(<nom\_var>);

#### ii) Fonctionnement

L'instruction de lecture suspend le déroulement de l'algorithme jusqu'à ce que l'utilisateur ait fini de taper au clavier une valeur et assigne la valeur entrée à la variable <nom\_var>. Le type de la variable détermine ce que l'on entre au clavier (entier, réel, ...).

**Exemple:**

1. {Exemple d'utilisation de *Lire*}
2. A : **Entier**
3. **DEBUT**
4.     **Lire** (A)
5.     **Ecrire** ('La Valeur de 5\*A est :', 5 x A)
6. **FIN**

Si l'utilisateur saisit la valeur 10, nous aurons alors à l'écran:

La Valeur de 5\*A est : 50

**4. Commentaire**

Le commentaire est l'indication d'un texte à ignorer lors de la traduction de l'algorithme.

**Syntaxe en LDA**

(\* <texte à ignorer>\*)



## TD1

**Exercice 1** Que fait la liste d'instructions suivantes ?

```
1:  $A \leftarrow 2$   
2:  $A \leftarrow A + 2$   
3:  $B \leftarrow A \times 2 + A$   
4:  $C \leftarrow 4$   
5:  $C \leftarrow B - C$   
6:  $C \leftarrow C + A - B$   
7:  $A \leftarrow B - C \times A$   
8:  $A \leftarrow (B - A) \times C$   
9:  $B \leftarrow (A + C) \times B$ 
```

**Exercice 2** Que fait la liste d'instructions suivantes ?

```
1:  $X \leftarrow -5$   
2:  $X \leftarrow X^2$   
3:  $Y \leftarrow -X - 3$   
4:  $Z \leftarrow (-X - Y)^2$   
5:  $X \leftarrow -(X + Y)^2 + Z$   
6:  $Y \leftarrow Z^X \times Y$   
7:  $Y \leftarrow -(Z + Y)$   
8:  $X \leftarrow X + Y - Z$   
9:  $Y \leftarrow X + Z$   
10:  $X \leftarrow (Y - Z)^2$   
11:  $Y \leftarrow X - Y$   
12:  $X \leftarrow (Y + Z)/(X/10)$   
13:  $Y \leftarrow ((X \times Z)/Y) \times 9$ 
```

## Comprendre les principes de l'affectation

**Exercice 3** Comment inverser le contenu de deux variables ?

```
1: {Inversion de deux variables}  
2: Entier  $A, B, X$   
3: Début  
4:   Lire  $A$   
5:   Lire  $B$   
6:   ...  
7:   ...  
8:   ...  
9:   Écrire ('La valeur de A est ',  $A$ )  
10:  Écrire ('La valeur de B est ',  $B$ )  
11: Fin
```

**Exercice 4** Comment faire sans utiliser une variable supplémentaire?

```

1: {Inversion de deux variables}
2: Entier A, B, X
3: Début
4: Lire A
5: Lire B
6: ...
7: ...
8: ...
9: Écrire ('La valeur de A est ', A)
10: Écrire ('La valeur de B est ', B)
11: Fin

```

**Exercice 5** étant donnée X, comment calculer le plus rapidement possible  $X^{16}$  ?

```

1: {Exponentiation}
2: Entier X
3: Début
4: Lire X
5: ...
6: ...
7: ...
8: ...
9: Écrire ('La valeur de  $X^{16}$  est ', X)
10: Fin

```

### Résolution de problèmes simples

**Exercice 6** : Ecrire un algorithme saisissant le prix "TTC" d'une marchandise et affichant le prix "Hors Taxe" sachant que cet article a une T.V.A. de 18,6%.

**Exercice 7** : Ecrire un algorithme saisissant 2 variables entières qui calcule et affiche leur moyenne.

**Exercice 8** : Ecrire un algorithme saisissant un temps en seconde que l'on transcrira en jours, heure, minutes, secondes.

**Exercice 9** : En se basant sur l'exercice précédent, Ecrire un algorithme permettant de faire la différence entre deux horaires saisis en heure, minutes, secondes.

**Exercice 10** : Ecrire un algorithme qui demande à l'utilisateur d'entrer la largeur et la longueur et afficher la surface d'un rectangle

**Exercice 11** : Créer un algorithme pour calculer la moyenne de 3 notes.

**Exercice 12** : Ecrire un algorithme qui demande à l'utilisateur d'entrer la largeur et la longueur et afficher la surface d'un rectangle

**Exercice 13** : Ecrire un algorithme qui demande à l'utilisateur le rayon et calculer la surface de disque.

**Exercice 14 :** Ecrire un algorithme pour lire la partie réelle et la partie imaginaire d'un nombre complexe  $z$  ( $Re+iIm$ ) et d'afficher le module.

**Exercice 15 :** Ecrire un algorithme qui demande à l'utilisateur de saisir les composantes de deux vecteurs et calculer le produit scalaire

**Exercice 16 :** Ecrire un algorithme pour lire le PHT, TVA et afficher le PTTC

Remarque :  $PTTC = PHT * (1 + TVA)$ .

**Exercice 17 :** Ecrire un algorithme pour calculer le quotient de deux nombres :

$a/b$