

# Manipulation des variables

## 1. Nommage des variables

Lors de l'analyse d'un problème, on est souvent amené à le décomposer en sous-problèmes qui doivent être résolus dans un ordre séquentiel bien précis. Chaque sous-problème produit des résultats qui peuvent être utilisés dans les sous-problèmes qui suivent. Il convient donc, pour pouvoir manipuler ces résultats, de les nommer. Ce nommage se fait par l'intermédiaire d'un **identificateur**. On parle aussi très souvent de **variable mémoire**.

À chaque variable mémoire, on associe les caractéristiques suivantes :

- sa **dénomination**, qui doit être judicieuse (deux résultats différents sont désignés par un identificateur différent),
- son **type** (ensemble auquel appartient la valeur qu'il désigne).

Ces caractéristiques sont précisées lors de la déclaration de la variable mémoire.


Syntaxe de la déclaration :


**Type** IDENTIFICATEUR

 **Type** est le type et IDENTIFICATEUR la dénomination de la variable mémoire.

Exemples :

```
Réel PI
Car SEPARATEUR
Bool TEST
Ent NBMOIS, NBJOURS
```


 Une fois une variable mémoire définie, on ne peut plus changer son type.

 Toute variable mémoire utilisée dans un algorithme doit avoir été déclarée auparavant.

Dans la suite de ce support de cours, il a été choisi de **nommer les variables mémoire en majuscules pour une meilleure lisibilité**. Sachez que de nombreux langages de programmation sont sensibles à la "casse" des variables mémoire (une variable MONTANT est différente d'une variable Montant !!!).

Évidemment en programmation ensuite (JavaScript dans notre cas), des conventions de nommages différentes pourront vous être proposées.

En JavaScript pour les variables mémoire (et les fonctions) la convention communément acceptée est le "camelCaps", c'est-à-dire que les noms doivent commencer systématiquement par une minuscule et ensuite des majuscules viendront s'insérer en début de chaque mot constituant le nom de la variable. Par exemple une variable comme CUMUL\_ANNUEL (la convention dans cette présentation de l'algorithmique) sera codée cumulAnnuel en JavaScript.

 Vous pourrez noter dans la suite du livre que dans de nombreux scripts JavaScript proposés la convention camelCaps n'est pas utilisée. Le choix ayant été fait de ne pas modifier les noms des variables mémoire des

## 2. Affectation

L'affectation consiste à stocker ou encore à imputer une expression à une variable mémoire. La syntaxe retenue dans cette présentation de l'algorithmique est :

IDENTIFICATEUR <- expression

L'affectation est notée par le symbole "<-" (et non pas le "=" qui sert à faire des comparaisons d'égalité).

L'expression doit être du type des valeurs désignées par l'identificateur. Cette expression devient la nouvelle valeur désignée par IDENTIFICATEUR.



Les conventions retenues ici peuvent être différentes de celles de certains langages de programmation. À titre indicatif en JavaScript l'affectation est notée "=" alors que la comparaison en égalité se code "==". Vous l'avez compris au travers de ce petit exemple, il faudra être très vigilant lors du codage en programmation. Nous aurons l'occasion de revenir sur cette difficulté à multiples reprises dans ce livre.

Pour illustrer les différentes notions décrites dans la suite de ce livre, des exemples vous seront proposés. Cela vous permettra de découvrir les mécanismes de l'algorithmique en situation. Vous pourrez aussi ensuite, c'est un peu également l'objectif de cette présentation, rédiger vos propres algorithmes pour des traitements que vous coderez finalement en JavaScript.

## 3. Exercice n°1 : Inversion du contenu de deux variables mémoire

### Sujet

Il vous est demandé d'inverser le contenu de deux variables mémoire de type entier. La première variable sera par exemple nommée A, la seconde B. La valeur initiale de A sera par exemple 5, celle de B sera 3. Il vous faut par algorithme inverser les contenus afin qu'au final A contienne 3 et B contienne 5.

#### **Début**

**Ent** A, B, C

A <- 5

B <- 3

C <- A

A <- B

B <- C

#### **Fin**

À titre indicatif le contenu des variables mémoire au fil du déroulement de l'algorithme sera :

- A <- 5 (A contient 5, B est vide)
- B <- 3 (A contient 5, B contient 3)
- C <- A (A contient 5, B contient 3, C contient 5)
- A <- B (A contient 3, B contient 3, C contient 5)
- B <- C (A contient 3, B contient 5, C contient 5)

Terminons par quelques règles générales de mise en forme et de syntaxe dans les corrigés :

- Vous pouvez prévoir une ligne blanche (interligne) entre les sections principales de votre algorithme (déclarations, initialisations, calculs, affichages des résultats...) et même pourquoi pas tout de suite après **Début** et avant **Fin**.
- Prévoyez un trait vertical pour relier le mot clé **Début** au mot clé **Fin**.
- Attention les mots clés apparaissent en **Gras/Italique** dans les corrigés (initiale uniquement en majuscule).
- Les noms des variables seront entièrement en MAJUSCULES (contraire aux conventions habituelles en JavaScript).

## 4. Affichage des résultats

Dans l'exercice précédent, l'inversion du contenu des deux variables A et B s'est faite sans difficulté. Par contre le contenu de ces deux variables n'a pas été présenté à l'écran (ou dans l'afficheur). Il est utile d'afficher le résultat des traitements, tout du moins le résultat final, à l'écran. Cet affichage (ou écriture) des résultats doit être demandé explicitement.

Au niveau algorithmique, on va considérer que l'écriture se fait sur une suite de lignes, sans indication particulière sur la mise en forme de l'affichage (mise en gras, italique, police de caractères spécifique...).

L'ordre d'écriture est réalisé par l'instruction **Ecrire**(e1, e2, ..., ei, ..., en) où les "ei" peuvent représenter :

- une expression,
- une chaîne de caractères notée entre guillemets,
- l'ordre **Aligne** qui fait commencer l'impression sur la ligne suivante.

Exemple n°1 :

```
Ecrire("Le carré de 4 est ", 4 * 4)  
provoque l'impression de :  
Le carré de 4 est 16
```

Exemple n°2 :

```
Ecrire("La note en français est ", 10 + 4, Aligne, "La note en informatique est ", (18 + 12) / 2)  
provoque l'impression de :  
La note en français est 14  
La note en informatique est 15
```

Exemple n°3 :

```
Ecrire("La note en français est ", 10 + 4)  
Ecrire("La note en informatique est ", (18 + 12) / 2)  
provoque l'impression de :  
La note en français est 14  
La note en informatique est 15
```

Exemple n°4 :

```

Réel MATH, INFO
MATH <- 15
INFO <- 13
Ecrire("La moyenne est ", (MATH + INFO) / 2)
provoque l'impression de :
La moyenne est 14

```

Exemple n°5 :

```

Réel MATH, INFO, MOY
MATH <- 15
INFO <- 13
MOY <- (MATH + INFO) / 2
Ecrire("La moyenne est ", MOY)
provoque l'impression de :
La moyenne est 14

```



Dans la grande majorité des langages de programmation, l'instruction d'affichage de messages à l'écran ne permet pas la présentation simultanée d'une information alphanumérique et d'une information numérique. Il conviendrait donc également en algorithmique d'intégrer cette contrainte en utilisant la fonction **Numérique\_vers\_chaine**. Dans la plupart des corrigés d'exercices proposés dans la suite de ce chapitre, cette fonction n'a pas été utilisée.

Exemple n°6 :

```

Réel MATH, INFO
MATH <- 15
INFO <- 13
Ecrire("La moyenne est ", Numérique_vers_chaine((MATH + INFO) / 2))
provoque l'impression de :
La moyenne est 14

```

## 5. Exercice n°2 : Surfaces de cercles

### Sujet

Calculer (et afficher à l'écran) la surface de deux cercles de rayons prédéterminés (5.5 mètres et 3.5 mètres par exemple) ainsi que la différence entre ces deux surfaces

### Corrigé

#### **Début**

```

Réel RAYON1, RAYON2, PI, SURFACE1, SURFACE2, DIFFERENCE

RAYON1 <- 5.5
RAYON2 <- 3.5
PI <- 3.14

SURFACE1 <- PI * RAYON1 * RAYON1
SURFACE2 <- PI * RAYON2 * RAYON2
DIFFERENCE <- SURFACE1 - SURFACE2

Ecrire("Surface 1 = ", SURFACE1, Alaligne, "Surface 2 = ",
SURFACE2, Alaligne, "Différence = ", DIFFERENCE)

```

## 6. Saisie au clavier

Les algorithmes ne présentent pas un intérêt général si, au cours de chaque exécution, ils fournissent toujours le même résultat.

Pour accéder à des données, saisies au clavier par l'opérateur (ou lues dans un fichier), on disposera au niveau algorithmique d'une instruction dédiée. La saisie de données sur le périphérique d'entrée se fera par l'intermédiaire de l'instruction **Lire**.

La saisie par **Lire** sera affectée à une variable mémoire (déclarée auparavant). Un message explicatif (**Ecrire**("Votre saisie : ") par exemple) devra précéder impérativement l'instruction **Lire** pour préciser à l'opérateur la saisie attendue.

Exemple :

**Début**

```

Co Déclarations Fco
Car NOM, PRENOM

Co Saisie du nom au clavier Fco
Ecrire("Nom : ")
NOM <- Lire

Co Saisie du prénom au clavier Fco
Ecrire("Prénom : ")
PRENOM <- Lire

Co Affichage écran pour contrôle Fco
Ecrire("L'identité est ", NOM, " ", PRENOM)

```

**Fin**

NB :

**Co** signifie début de commentaire

**Fco** signifie fin de commentaire



Les commentaires placés dans un algorithme (et dans un programme informatique) ne jouent pas de rôle effectif. Dans le cas d'un langage de programmation comme JavaScript, ils seront ignorés par la machine lors de l'exécution du programme. Par contre ils sont fortement requis car ils permettent au programmeur (ou aux tiers qui auraient à intervenir ultérieurement sur l'algorithme ou programme) de comprendre plus facilement le raisonnement codé. Les commentaires sont placés a minima au début de chaque portion de code spécifique.

## 7. Exercice n°3 : Surface et volume d'une sphère

Sujet

Calculer et afficher la surface et le volume d'une sphère dont la valeur du rayon sera saisie au clavier

Corrigé

## **Début**

**Co** Déclarations **Fco**

**Réel** PI, RAYON, SURFACE, VOLUME

**Co** Saisie du rayon au clavier **Fco**

**Ecrire**("Rayon : ")

RAYON <- **Lire**

**Co** Calculs **Fco**

PI <- 3.14

SURFACE <- 4 \* PI \* RAYON \* RAYON

VOLUME <- SURFACE \* RAYON / 3

**Co** Affichage des résultats **Fco**

**Ecrire**("Rayon de la sphère = ", RAYON, **Alaligne**, "Surface = ", SURFACE, **Alaligne**, "Volume = ", VOLUME)

## **Fin**