Algorithmique

CHAPITRE 2 : LES VARIABLES

Sommaire

1. Définition et utilité des variables

- Dans un programme informatique, on va avoir besoin de stocker généralement des valeurs, comme des nombres fournis par l'utilisateur, par exemple dans le cas d'un programme qui effectue l'addition entre deux nombres.
- **La variable** est donc cette boite qui va nous permettre de stocker chacune de ces valeurs au cours du programme. Ces valeurs peuvent être de plusieurs types comme des nombres, du texte ...

1. Définition et utilité des variables

- Lorsqu'on crée une variable dans un programme, on la désigne par une étiquette et pour avoir accès au contenu de la variable, il suffit de la désigner par son étiquette.
- Dès que l'on a besoin de stocker une information au cours d'un programme, il faut toujours penser à utiliser une variable.

<u>Exemple</u> : Pour écrire le programme qui va calculer la somme entre deux nombres, on va avoir besoin de 3 variables :

- **Une variable pour contenir le nombre 1.**
- **Une autre pour contenir le nombre 2.**
- **Et la dernière pour contenir le résultat de l'opération nombre 1 + nombre 2**

A vous de jouer : De combien de variables aurons nous besoin pour réaliser le calcul suivant : $(1234 + 456)\ x\ (908 + 567)$?

2. Déclaration des variables

- La déclaration d'une variable se fait selon deux conventions et peut varier d'un langage à l'autre. Les principaux groupes de langages sont:
  - **Le groupe des langages typés :** dans ce groupe de langages lorsqu'on déclare une variable on doit désigner le type des informations qu'elle va contenir. Par exemple quand on crée une variable pour contenir des nombres, on ne pourra pas en cours de programme y mettre du texte...

```
int boiteNombre = 1;
```

  - **Le groupe des langages non typés :** ici contrairement au précédent lorsqu'on déclare une variable on est pas obligé de spécifier le type d'informations qu'elle contiendra donc une variable peut contenir des nombres, des images .....

```
var boiteNombre = 1;
```

- La majorité des langages déclarent les variables selon l'une des deux conventions soit typée ou non typée mais parfois on peut rencontrer une manière un peu différente de déclaration des variables.

- Pour le pseudo-code, nous allons déclarer nos variables par un nom et un type d'informations qu'elle contiendra appelé généralement **le type de la variable**.

3. Nom des variables

- En pseudo-code, nous avons besoin de deux informations pour déclarer une variable, son nom et son type.
- Le choix du nom de la variable est laissé au programmeur, mais ce nom ne se choisit pas n'importe comment.
- Un nom de variable peut importe le langage peut comporter des lettres et des chiffres mais il faut exclure la plupart des signes de ponctuations et des espaces.
- Le nom de la variable doit être intuitif, pour pouvoir se retrouver dans son programme. Il faut choisir un nom qui donne des indications sur son contenant.

Exemple : Pour le programme de calcul de la somme entre deux nombres, on va préférer les noms de variable pour le nombre 1 comme : **nombre1, nbre1, ...** pour désigner la variable qui va contenir le nombre 1 et éviter les noms comme **a, b, c, variable_qui_contient_le_nombre1 ....**

- Une variable peut avoir un nom avec des minuscules et des majuscules mais en un seul mot pas avec des espaces mais intuitif et court.

4. Types numériques des variables

- Le type numérique est le type qui va nous permettre de désigner le stockage des nombres.

- Les principaux types que nous utiliserons :
  - **Byte :** pour désigner une variable qui va contenir un nombre entre 0 et 255.
  - **Entier :** pour désigner une variable qui va contenir un nombre sans virgule. On dispose de deux familles d'entiers: les entiers simples (nombres entre −32 768 et 32 767) et les entiers longs (−2 147 483 648 et 2 147 483 647).
  - **Réel :** pour désigner une variable qui va contenir un nombre à virgule. On en dénombre deux types : les réels simples pour les nombres à virgule de petite taille et les réels doubles identique au réels simples mais avec des nombres à virgule de grande taille
  - En pseudo-code on déclarera nos variables sans trop spécifier les sous classes. On aura des exemples comme:

```
Variable nombre1 en Entier

Variables nombre2, resultat en Réel
```

4. Types numériques des variables

- On peut retrouver d'autres types numériques comme :
  - Le type **monétaire** pour la représentation des monnaies (tout ce qui est argent).
  - Le type **date** pour représenter les dates.
  - Le type **décimal** identique au réel mais ici on peut représenter des nombres encore plus grands.
- La manière de représenter chacun de ces types varie d'un langage à l'autre.
- Lorsqu'on arrive en programmation, lors de la traduction de l'algorithme en langage informatique, il faudra veiller pour les **langages typés** à bien spécifier le type dont on a vraiment besoin.
- Quand on est sûr que notre variable ne dépassera pas des valeurs au delà du byte, il faut préférer l'usage du byte lors de la déclaration que d'un type encore plus grand.
- Utiliser les types nécessaires permettra de limiter le gaspillage de ressources machine et d'utiliser les ressources dont on a vraiment besoin.

5. Type alphanumérique

- Un programme informatique ne manipule pas seulement des nombres mais aussi du texte, des caractères…
- Pour prendre en compte ce type d'informations comme du texte, un type est utilisé il s'agit de l'**alphanumérique** également appelé **type caractère, type chaîne** ou en anglais **type string**.
- L'alphanumérique permet de contenir une suite de caractères qui peut être soit des chiffres, des lettres, des caractères de ponctuation.
- Dans ce cours, nous allons utiliser la notion de **Chaîne** pour désigner une suite de caractères et la notion de **Caractère** pour désigner un seul caractère.
- Avec ces deux types, on pourra déclarer des variables de type **Chaîne** qui ne pourront contenir plus de deux caractères et déclarer des variables de type **Caractère** qui ne pourront contenir qu'un seul caractère.
- Ces deux types sont pris en compte aujourd'hui dans la quasi-totalité des langages de programmation.

5. Type alphanumérique

- Lorsqu'on veut donner une valeur à une variable de type chaine de caractères ou caractère on doit mettre la valeur entre guillemets "valeur *variable*" ou *'valeur variable'*.

- Pour déclarer une variable chaine on va utiliser cette notation en pseudo-code:

```
Variable nom en Chaîne
```

- La variable **nom** peut contenir des valeurs comme *"Maurice"* ou *'Oscar'*...

- Pour déclarer une variable caractère on va utiliser cette notation en pseudo-code:

```
Variable reponse en Caractère
```

- La variable **reponse** peut contenir des valeurs comme *"O"* ou *'N'*...

Une variable chaîne peut contenir des valeurs d'une variable caractère.

6. Type booléen

- Le type **booléen** est le type qui permet de stocker uniquement les valeurs logiques **VRAI ou FAUX,** en anglais **TRUE** ou **FALSE**.
- Ce type permet de tester des assertions par exemple en affectant à une variable booléenne la vérification de $3 < 4$, la variable booléenne prendra la valeur de VRAIE car 3 est inférieur à 4.
- C'est très abstrait comme concept, mais dans un programme informatique, les booléens nous permettent de faire des tests sans savoir précisément le contenu des variables.
- Certains langages comme le langage C implémentent le booléen par l'utilisation d'un bit avec 0 pour Faux et 1 pour Vrai.
- En pseudo-code nous allons utiliser cette notation pour déclarer une variable booléenne :

```
Variable test en Booléen
```

7. L'instruction d'affectation

- Une variable est comme une boite permettant de contenir des valeurs qui seront utilisées au cours du programme. **Mais comment mettre une valeur dans une variable?**

- Pour réaliser cette action on va effectuer une opération d'affectation qui va permettre d'assigner ou modifier une valeur dans une variable.

- En pseudo-code nous utiliserons $<-$ pour assigner une valeur à une variable.

```
nombre <- 4
```

- Avec l'instruction ci-dessus, on assigne à la variable ***nombre*** la valeur 4.

- On peut aussi assigner à une variable, le contenu d'une autre variable. Par exemple pour affecter à une variable ***nombre1*** le contenu de la variable ***nombre***, on procédera comme suit :

```
nombre <- 4

nombre1 <- nombre
```

8. Ordre des instructions

- Pour écrire nos algorithmes nous allons utiliser pour ce cours l'ordre suivant:
  - Tout algorithme doit commencer par une instruction **Début (ou DEBUT)** et se terminer par une instruction **Fin (ou FIN)**.
  - Après l'instruction **Début,** on procède à la déclaration des variables grâce à l'instruction **Variable/VARIABLE** ou utiliser **Variables/VARIABLES** pour plusieurs variables de même type.
  - Après la déclaration des variables, les autres opérations peuvent être exécutées : l'affectation, lecture/écriture.
  - Terminer son algorithme par l'instruction **Fin** ou **FIN**.

```
Début
    Variable nombre1 en Entier
    nombre1 <- 1
Fin
```

```
DEBUT
    VARIABLES nombre1, nombre2 En Entier
    nombre1 <- 1
FIN
```

Cette notation et méthodologie est juste un choix personnel d'écriture de pseudo-code. Contrairement aux langages de programmation, la présentation du pseudo-code est au choix du développeur.

9. Exercices

Exercice 1

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Début
    Variables A, B en Entier
    A <- 1
    B <- A + 3
    A <- 3
Fin
```

Exercice 2

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Début
    Variables A, B, C en Entier
    A <- 5
    B <- 3
    C <- A + B
    A <- 2
    C <- B - A
Fin
```

9. Exercices

Exercice 3

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Début
    Variables A, B en Entier
    A <- 5
    B <- A + 4
    A <- A + 1
    B <- A - 4
Fin
```

Exercice 4

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Début
   Variables A, B, C en Entier
   A <- 3
   B <- 10
   C <- A + B
   B <- A + B
   A <- C
Fin
```

9. Exercices

Exercice 5

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Début
    Variables A, B en Entier
    A <- 5
    B <- 2
    A <- B
    B <- A
Fin
```

Les deux dernières instructions permettent-elles d'échanger les deux valeurs de B et A ? Si l'on inverse les deux dernières instructions, cela change-t-il quelque chose ?

Exercice 6

Ecrire un algorithme permettant d'échanger les valeurs de deux variables A et B, et ce quelque soit leur contenu préalable.

9. Exercices

Exercice 7

Soit trois variables A, B et C de valeurs respectives 12, 13, 3.

Ecrire un algorithme qui transfère à B la valeur de A, à C la valeur de B et à A la valeur de C.

Tester l'algorithme avec différentes valeurs de A, B, et C.

10. Corrections

Exercice 1

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Début
    Variables A, B en Entier
    A <- 1
    B <- A + 3
    A <- 3
Fin
```

```
On exécute cet algorithme comme suit:
A <- 1 implique A = 1
B <- A + 3 implique A = 1 et B = 4
A <- 3 ici on écrase l'ancienne valeur de A donc A = 3 et B = 4

Les variables A et B auront donc: A = 3 et B = 4.
```

Exercice 2

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Début
    Variables A, B, C en Entier
    A <- 5
    B <- 3
    C <- A + B
    A <- 2
    C <- B - A
Fin
```

```
On exécute cet algorithme comme suit:
A <- 5 implique A = 5
B <- 3 implique A = 5 et B = 3
C <- A + B implique A = 5, B = 3 et C = 8
A <- 2 implique A = 2, B = 3 et C = 8
C <- B - A implique A = 2, B = 3 et C = 1

Les variables A, B et C auront: A = 2, B = 3 et C = 1.
```

10. Corrections

Exercice 3

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Début
    Variables A, B en Entier
    A <- 5
    B <- A + 4
    A <- A + 1
    B <- A - 4
Fin
```

```
On exécute cet algorithme comme suit:
A <- 5 implique  A = 5
B <- A + 4 implique A = 5 et B = 9
A <- A + 1 implique A = 6 et B = 9
B <- A - 4 implique A = 6 et B = 2

Les variables A et B auront: A = 6 et B = 2
```

Exercice 4

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Début
    Variables A, B, C en Entier
    A <- 3
    B <- 10
    C <- A + B
    B <- A + B
    A <- C
Fin
```

```
On exécute cet algorithme comme suit:
A <- 3 implique A = 3
B <- 10 implique A = 3 et B = 10
C <- A + B implique A = 3, B = 10 et C = 13
B <- A + B implique A = 3, B = 13 et C =13
A <- C implique A = 13, B = 13 et C = 13

Les variables A, B et C auront: A = 13, B = 13 et C = 13
```

10. Corrections

Exercice 5

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Début
   Variables A, B en Entier
   A <- 5
   B <- 2
   A <- B
   B <- A
Fin
```

```
On exécute cet algorithme comme suit:
A <- 5 implique A = 5
B <- 2 implique A = 5 et B = 2
A <- B implique A = 2 et B = 2
B <- A implique A = 2 et B = 2

Les variables A et B auront: A = 2 et B = 2.
Non cet algorithme ne permet pas d'échanger les valeurs de A et B.
En inversant les deux dernières instructions on aura A = 5 et B = 5.
```

Exercice 6

Ecrire un algorithme permettant d'échanger les valeurs de deux variables A et B, et ce quelque soit leur contenu préalable.

```
Algo: Echanger les valeurs de deux variables
Début
   Variables A, B, tmp en Entier
   tmp <- A
   A <- B
   B <- tmp
Fin
```

10. Corrections

Exercice 7

Soit trois variables A, B et C de valeurs respectives 12, 13, 3.

Ecrire un algorithme qui transfère à B la valeur de A, à C la valeur de B et à A la valeur de C.

Tester l'algorithme avec différentes valeurs de A, B, et C.

```
Algo: Echanger les valeurs de trois variables
Début
    Variables A, B, C, tmp en Entier
    A <- 12
    B <- 13
    C <- 3
    tmp <- B
    B <- A
    A <- C
    C <- tmp
Fin
```

Citations

« Compter en octal, c'est comme compter en décimal, si on n'utilise pas ses pouces. » Tom Lehrer

« A l'origine de toute erreur attribuée à l'ordinateur, vous trouverez au moins deux erreurs humaines, dont celle consistant à attribuer l'erreur à l'ordinateur. » Anonyme

« Un langage de programmation est une convention pour donner des ordres à un ordinateur. Ce n'est pas censé être obscur, bizarre et plein de pièges subtils. Çà, se sont les caractéristiques de la magie. » Dave Small