Cours/TD n°1 : les variables

Les variables : affectation, saisie et affichage.

Qu'est ce qu'une variable ?

Pour qu'un programme ne donne pas toujours le même résultat, un programme a constamment besoin de stocker des valeurs. Ces valeurs peuvent être récupérées grâce au clavier, grâce à des fichiers, grâce au réseau… Bref, quel que soit l'origine de la valeur, le programme a besoin de les stocker pour pouvoir les utiliser. Le mot **variable** est donc utilisé pour représenter tout ce que l'utilisateur va « donner » à l'ordinateur comme informations. Pour employer une métaphore, on peut dire que déclarer une variable dans un programme revient à **réserver une boite** en mémoire. Par exemple, dans l'énoncé suivant :

`Écrire l'algorithme décrivant un programme qui demande à l'utilisateur de saisir deux valeurs entières et affiche le résultat de leur somme.`

On remarque que le programme va avoir besoin de deux valeurs que l'utilisateur donnera. Ces deux **valeurs**, que le programme devra stocker, vont donner naissance à deux **variables**.

Par ailleurs, il existe d'autres utilisations des variables. Pour reprendre l'exemple précédant, supposons que nous ayons besoin de cette somme pour d'autres traitements (par exemple, tester si la somme est supérieure à 10 puis de diviser cette somme par deux). Nous aurons alors la possibilité d'utiliser une variable pour stocker le **résultat temporaire**.

Pour conclure, nous aurons deux utilisations des variables :

- Une variable par information que l'utilisateur donnera à l'ordinateur
- Une variable pour chaque résultat temporaire.

Mise en pratique :

Identifier le nombre de variables qui seront nécessaires à la conception des programmes suivants

- `Écrire un programme qui demande à l'utilisateur de saisir son année de naissance, son mois de naissance puis qui affiche son signe astrologique.`
- `Écrire un programme qui demande à l'utilisateur de saisir son nom, son prénom et son âge et qui affiche s'il est majeur.`
- `Écrire un programme qui calcule la moyenne de 10 notes que l'utilisateur donnera, et qui affiche « bravo » si la moyenne est supérieure à 15, « bien » si la moyenne est entre 10 et 15 et « attention » si la moyenne est inferieur à 10.`
- `Écrire un programme qui demande à l'utilisateur de saisir les trois cotés d'un triangle et qui affiche si le triangle est rectangle.`
- `Écrire l'algorithme affichant le minimum d'une suite de 10 réels saisis au clavier.`

Utilisation des variables en algorithmique.

En algorithmique, la syntaxe est stricte (même si c'est moins stricte qu'en c++). Nous devons donc déclarer les variables en début de programme, avant le mot clé « DEBUT ». Cela aura pour conséquence de réserver des boites (vides) en mémoire. En reprenant l'exemple précédent, nous écrirons donc :

```
PROGRAMME Somme
VAR       val1,val2 : entier
DEBUT
```

Ainsi, nous avons créé deux boites : l'une s'appelle `val1`, l'autre `val2`. A ce propos, le nom des variables est très important et obéit à des règles strictes. La première règle interdit l'utilisation de signe de ponctuation dans le nom (pas de virgules, pas d'espaces, pas d'apostrophe…). La seconde règle est qu'un nom de variable ne doit pas commencer par un nombre. Mis à par ces deux règles, nous sommes libre d'utiliser ce que l'on veut, mais il est souhaitable d'utiliser un nom compréhensible.

Par ailleurs, nous avons contraint le contenu des boites avec le mot « `entier` » : c'est le type de la boite. Avec cette contrainte, on oblige à ce que tout ce qui rentrera dans la boite sera un chiffre entier (`1, 50, 1953, -21, 33,` …). Il est obligatoire de préciser dès le départ ce que la boite contiendra. Cela permet à l'ordinateur de prévoir la taille de la boite : un entier aura besoin d'une petite boite, tandis qu'une chaine de caractères aura besoin d'une grande boite. Il y a 5 types à connaitre :

- `Entier` : La boite ne contiendra que des chiffres entiers (1, 5, -9000, 1256, 98, -45)
- `Réel` : La boite ne contiendra que des chiffres entiers (1.5, 5.0, -90.125, 1.256, 9.8, -45.0)
- `Caractère` : La boite ne contiendra que des caractères (‘a’, ’b’, ’ ‘, ’7’, ’/’, ’^’, ‘R’, ‘.’)
- `Chaine` : La boite contiendra une suite de caractères (« bonjour », « Ha! Un vrai cours ! »)
- `Booléen` : La boite ne contiendra que VRAI ou FAUX (ou 1 et 0…) L'important à comprendre, c'est que ce type est très économique en place mémoire.

Attention !

Ne pas faire la confusion entre une variable `Chaine` contenant des nombres (par exemple « `185` ») et une variable `Entier` ! Sur une variable de type `Chaine`, il n'est pas possible de faire des calculs, il faut alors considérer le nombre comme un mot composé des caractères ‘1’, ‘8’, et ‘5’. Pour éviter de confondre un nombre d'une chaine, il faut **TOUJOURS** noter une chaine entre guillemets !

Mise en pratique

Reprendre les exemples de la page précédente et donner le nom et le type de chaque variable.

Manipuler les variables

Nous avons donc vu jusqu'ici comment faire pour réserver en mémoire une boite : déclarer le **nom** de la boite, et déclarer la **taille** de la boite (le **type**). Cependant, il manque encore une opération importante : mettre du contenu dans la boite… En effet, une boite vide ne sert à rien !

L'affectation

En algorithmique, on représente l'opération qui consiste à mettre une valeur dans une boite par une flèche : `<-`. Ainsi, l'opération suivante `titi <- 3` revient à mettre 3 dans la boite qui s'appelle `titi`. Ceci, soit dit en passant, sous-entend impérativement que `titi` soit une variable de type `Entier`. Si `titi` a été défini dans un autre type, il faut bien comprendre que cette instruction provoquera une erreur. C'est un peu comme si, en donnant un ordre à quelqu'un, on accolait un verbe et un complément incompatibles, du genre « Epluchez la casserole ». Même dotée de la meilleure volonté du monde, la ménagère lisant cette phrase ne pourrait qu'interrompre dubitativement sa tâche. Alors, un ordinateur, vous pensez bien…

Il est possible d'affecter d'autres valeurs à la boite `titi`, mais dans ce cas, la valeur sera à chaque fois remplacée par la nouvelle valeur. Il n'est possible de stocker **qu'une seule valeur par case** (sauf pour les tableaux que nous verrons plus tard). Par ailleurs, il y a de nombreuses manières de mettre une valeur dans une variable. Par exemple :

```
titi <- 4 + toto
```

signifie que la boite `titi` récupère le résultat de la somme de 4 avec la valeur de la boite `toto`. Important : une instruction d'affectation **ne modifie pas** ce qui est situé à droite de la flèche ! Il est aussi possible d'utiliser la même variable à droite et à gauche de la flèche. Cela permet notamment d'incrémenter la valeur :

```
titi <- titi + 1
```

Si `titi` valait 4 avant l'instruction, `titi` vaut 5 après.

Par ailleurs, il faut faire attention au nom de la variable (l'étiquette sur la boite) et la valeur (le contenu de la boite). Voilà un exemple de confusion fréquent :

Exemple 1 :

Début
Riri ← "Loulou"
Fifi ← "Riri"
Fin

Exemple 2 :

Début
Riri ← "Loulou"
Fifi ← Riri
Fin

La différence entre les deux exemples ne concerne que des guillemets. Mais le résultat du premier exemple n'a rien à voir avec le résultat du second exemple. En effet, dans le premier exemple, la variable `Fifi` contient la suite de caractères `Riri`, tandis que dans le second exemple, la variable `Fifi` contient la suite de caractères `Loulou` !

Mise en pratique

- Donner les valeurs des différentes variables à la fin des algorithmes (sauf si l'algorithme est faux).

Pour des raisons de place, je ne mets pas l'entête du programme, mais il est sous-entendu…

1
```
Var A, B : Entier
Début
A ← 1
B ← A + 3
A ← 3
Fin
```

2
```
Var A, B, C : Entier
Début
A ← 5
B ← 3
C ← A + B
A ← 2
C ← B – A
Fin
```

3
```
Var A, B : Entier
Début
A ← 5
B ← A + 4
A ← A + 1
B ← A – 4
Fin
```

4
```
Var A, B, C : Entier
Début
 A ← 3
B ← 10
C ← A + B
B ← A + B
A ← C
Fin
```

5
```
Var A, B : Entier
Début
A ← 5
B ← 2
A ← B
B ← A
Fin
```

6
```
Var A, B, C en Caractères
Début
A ← "423"
B ← "12"
C ← A + B
Fin
```

7
```
Var A, B, C en Caractères
Début
A ← "423"
B ← "12"
C ← A & B
Fin
```

Plus difficile :

- Ecrire un algorithme permettant d'échanger les valeurs de deux variables A et B de type entier, et ce quel que soit leur contenu préalable (on prendra `A<-5` et `B<-2`)
- Une variante du précédent : on dispose de trois variables de type réel : A, B et C. Ecrivez un algorithme transférant à B la valeur de A, à C la valeur de B et à A la valeur de C (on prendra `A<-5.6`, `B<-2.1` et `C<-15.43`).

La saisie au clavier et l'affichage.

Affecter des valeurs à des variables, c'est bien… Mais pour l'instant, on peut se demander à quoi ça peut servir. En effet. Imaginons que nous ayons fait un programme pour calculer le carré d'un nombre, mettons 12. Si on a fait au plus simple, on a écrit un truc du genre :

```
Programme carre
VAR  A : entier
DEBUT
A <- 12 * 12
FIN
```

D'une part, ce programme nous donne le carré de 12. C'est très gentil à lui. Mais si l'on veut le carré d'un autre nombre que 12, il faut réécrire le programme. Bof.

D'autre part, le résultat est indubitablement calculé par la machine. Mais elle le garde soigneusement pour elle, et le pauvre utilisateur qui fait exécuter ce programme, lui, ne saura jamais quel est le carré de 12. Re-bof.

Pour pallier à ces problèmes, nous avons à notre disposition deux instructions : **`Saisir`** et **`Afficher`**. **`Saisir`** permet d'envoyer les valeurs entrées au clavier dans une variable, tandis qu'**`afficher`** permet de récupérer des valeurs (dans des variables, des constantes ou des valeurs marquées directement dans le code).

Simplement, pour entrer la nouvelle valeur d'une variable, on mettra

```
Saisir titi
```

Dès que la touche Entrée a été frappée, l'exécution reprend, avec la valeur tapée au clavier qui se retrouve dans la variable `titi`. En mettant plusieurs variables séparées par des virgules, le programme attend que l'utilisateur ait entré une valeur pour chaque variable. Par exemple :

```
Saisir titi, tata, tutu
```

Le programme va attendre que l'utilisateur saisisse les 3 valeurs.

Pour afficher quelque chose à l'écran, il suffit de mettre

```
Afficher « Une belle phrase »
```

Cette instruction aura pour effet d'afficher sur l'écran de l'ordinateur « Une belle phrase ». Il est possible de combiner des variables, des opérations, et des valeurs :

```
Afficher « une variable »,A, « et une opération »,(A+4)*B
```

L'écran affichera alors « une variable » suivit de la valeur qui est dans la variable A, puis « et une opération » suivit du résultat du calcul.

Mise en pratique

- Ecrire l'algorithme demandant à l'utilisateur son nom et qui affiche « Bonjour Mr » suivit du nom de l'utilisateur
- Ecrire un programme qui lit le prix HT d'un article, le taux de la TVA, et qui fournit le prix total TTC correspondant. Faire en sorte que l'affichage soit de cette forme : 00 € TTC (avec les 00 remplacés par la valeur saisie par le client). Pour rappel : Prix TTC = (Prix HT * taux TVA)/100
- Ecrire l'algorithme demandant à l'utilisateur de saisir un nombre et d'afficher ce nombre à la puissance 2 suivant cette syntaxe :
  « 00 puissance 2 = 00 » (avec les 00 remplacés par les bonnes valeurs…)

Si vous en voulez encore, faites les exercices de la première page, et envoyez moi vos réponses à thibault.lelore@gmail.com

N'hésitez pas non plus à utiliser cette adresse pour me faire part de vos remarques sur mon cours, vos questions… Je suis ouvert au dialogue !