Humanbooster

Algorithmie

02 – Variables et fonctions

DWWM PE6 LYON

Barthélémy DELUY –

Introduction

Écrire des instructions détaillées, c’est bien, mais c’est rapidement limité…

Dans ce support, nous allons découvrir 2 nouveaux concepts fondamentaux de la programmation : les variables, et les fonctions.

Sommaire

0. Les conventions de nommage 1/5

Les variables et les fonctions sont des manières de nommer des éléments dans notre algorithme.

Il existe quelques règles à suivre pour écrire ces noms :

- Le snake_case
- Le PascalCase
- Le camelCase
- Le SCREAMING_SNAKE_CASE

1. Les variables 1/5

Une variable est un nom qu'on donne à une zone mémoire, dans laquelle on va pouvoir stocker une valeur, pour la réutiliser plus tard.

Une variable est donc un contenant, un récipient, dans lequel on va enregistrer une valeur. Les noms des variables sont écrites soit en snake_case, soit en camelCase.

En algo, pour déclarer une variable, on va utiliser :

    Déclarer ma_variable

ATTENTION : une variable n'existe que pour la routine qui la déclare !

Une variable cesse donc d'exister au mot-clé FIN de sa routine, et elle n'est connue que par cette routine. On parle de « variable locale ».

1. Les variables 2/5

Pour enregistrer une valeur dans une variable, on parle « d'affecter » la valeur à la variable. Il est possible d'affecter la variable directement lors de sa déclaration :

En algo : Déclarer ma_variable ← 4

En JS/TS : let ma_variable = 4;

En PHP : $ma_variable = 4;

1. Les variables 3/5

Pour réutiliser la valeur, il suffit d'indiquer la variable dans laquelle est stockée :

resultat ← 4+2
Afficher resultat

1. Les variables 4/5

Pour changer la valeur enregistrée dans une variable, il suffit de la réaffecter :

resultat ← 4+2
resultat ← 3+1

1. Les variables 5/5

ATTENTION : il existe 2 types de langages de programmation vis-à-vis des variables :

- Les langages faiblement typés (ou « à typage faible ») : on peut stocker ce qu'on veut dans une variable. JS et PHP sont des langages faiblement typés.
- Les langages fortement typés (ou « à typage fort ») : quand on déclare une variable, on doit indiquer quel type de valeurs on veut y stocker : des lettres, des chaînes de caractères, des entiers, des nombres à virgule, etc (les types disponibles sont spécifiques à chaque langage). TypeScript et PHP sont des langages fortement typés.

1. Les fonctions 1/10

Une routine est un nom que l'on donne à un ensemble d'instructions (un « bloc de code ») pour pouvoir l'exécuter quand on le souhaite. Une routine peut être soit une procédure, soit une fonction.

Le nom des routines est écrit généralement en camelCase.

On les utilise :

- Pour un morceau de code que l'on veut exécuter à plusieurs endroits du programme (par exemple une fonction de validation)
- Pour découper son code pour le rendre plus lisible

FLASH BONNE PRATIQUE :

Une bonne procédure/fonction respecte les règles suivantes :

- Moins de 25 lignes
- Moins de 4 niveaux d’indentation

1. Les fonctions 3/10

Syntaxe en Algo :

FONCTION/PROCÉDURE nomDeLaProcedure()
DÉBUT
    <instructions>
FIN

Les procédures et les fonctions doivent être déclarées AVANT de pouvoir être utilisées. On les place donc systématiquement avant le programme principal.

1. Les fonctions 4/10

Pour utiliser une routine, il suffit de l'appeler par son nom suivi des parenthèses au moment où on souhaite l'exécuter :

<instructions>
nomDeLaProcedure()
<instructions>

1. Les fonctions 5/10

Les fonctions ont également un rôle particulier : elles renvoient (retournent) une valeur à leur routine appelante. Une procédure est une fonction qui ne renvoie rien. Une fonction renvoie systématiquement une valeur, qu'il faudra récupérer :)

Par exemple :

```
FONCTION checkConnection()
DÉBUT
    SI Utilisateur connecté ALORS
        Renvoyer VRAI
    FIN SI
    Renvoyer FAUX
FIN

var is_connected ← checkConnection()
```

1. Les fonctions 6/10

Si on fait de l'algo pour un langage fortement typé, on précisera le type de retour (le type de la valeur retournée) au moment de déclarer la fonction :

FONCTION checkConnection() : booléen

…

1. Les fonctions 7/10

Il est possible de transmettre des valeurs à une routine, pour qu'elle puisse les utiliser. Dans ce cas, on parle de paramètres, ou d'arguments, de la routine. On les indique dans les parenthèses :

```
FONCTION additionner(var i, var j)
DÉBUT
    Retourner i+j
FIN

var resultat ← additionner( 1, 2)
```

1. Les fonctions 8/10

L’expression « additionner(var i, var j) » est appelée signature, ou prototype, de la routine.

Les paramètres sont donc des variables, dont les noms ne sont connus que par cette routine.

ATTENTION : le passage de paramètres est généralement fait « par copie », c’est-à-dire que la routine reçoit une copie de la valeur. Si elle la modifie, ça ne change pas la valeur de la variable de la routine qui l’a appelée.

```
FONCTION additionner( entier i, entier j) : entier
DÉBUT
    Retourner i+j
FIN

PROGRAMME calculette
DÉBUT
    entier n1 ← 2, n2 ← 4
    Afficher additionner( n1, n2)
FIN
```

1. Les fonctions 10/10

Il est tout à fait possible qu'une routine appelle une autre routine :

```
FONCTION checkHeureValide( var horaire)
….

FONCTION addHeures( var heure1, heure2)
DÉBUT
    SI checkHeureValide( heure1) ET checkHeureValide( heure2) ALORS
        Retourner heure1 + heure2
    FINSI
```

L'enchaînement des appels de routine s'appelle la « pile d'appel ».

Conclusion

Nous venons de découvrir deux nouveaux concepts fondamentaux de la programmation, les variables et les fonctions.

Reprenons les algos de ce matin pour les rendre plus lisibles :-)