Introduction à la programmation

David Roche, lycée G Fichet Bonneville (Haute-Savoie)

Introduction

Programmer un ordinateur, c'est quoi ?
Programmer, c'est créer des programmes (suite d'ordres donnés à l'ordinateur) ! Un ordinateur sans programme ne sait rien faire. Il existe différents langages qui permettent de programmer un ordinateur, mais le seul directement utilisable par le processeur est le langage machine (suite de 1 et de 0), aussi appelé binaire. Aujourd'hui (presque) plus personne ne programme en binaire (trop compliqué).
Les informaticiens utilisent des instructions (mots souvent en anglais) en lieu et place de la suite de 0 et de 1. Ces instructions, une fois écrites par le programmeur, sont « traduites » en langage machine. Un programme spécialisé assure cette traduction. Ce système de traduction s'appellera interpréteur ou bien compilateur, suivant la méthode utilisée pour effectuer la traduction.

Il existe 2 grandes familles de langages de programmation :
les langages de bas niveau sont très complexes à utiliser, car très éloignés du langage naturel, on dit que ce sont des langages « proches de la machine », en contrepartie ils permettent de faire des programmes très rapides à l'exécution. L'assembleur est le langage de bas niveau. Certains "morceaux" de programmes sont écrits en assembleur encore aujourd'hui.

Les langages de haut niveau sont eux plus « faciles » à utiliser, car plus proches du langage naturel (exemple : si a=3 alors b=c)
Exemples de langages de haut niveau : C, C++ , java, Qbasic, ...
NB : Aujourd'hui, le langage roi reste le C++, tout programmeur sérieux (professionnel) doit connaître le C++, mais il est assez difficile à apprendre (certains le disent trop près de la machine malgré son statut de langage de haut niveau).

Les variables

Un programme « passe son temps » à traiter des données. Pour pouvoir traiter ces données, l'ordinateur doit les ranger dans sa mémoire (RAM, voir partie sur le matériel). La RAM se compose de cases dans lesquelles nous allons ranger ces données (une donnée dans une case).
Chaque case a une adresse (ce qui permet au processeur de savoir où sont rangées les données).
Alors, qu'est-ce qu'une variable ?
Eh bien c'est une petite information (une donnée) temporaire que l'on stocke dans une case de la RAM. On dit qu'elle est "variable" car c'est une valeur qui peut changer pendant le déroulement du programme.
Une variable est constituée de 2 choses :
– Elle a une valeur : c'est la donnée qu'elle stocke (par exemple le nombre 5 ou la suite de caractères (chaîne de caractères) « bonjour »)
– Elle a un nom : c'est ce qui permet de la reconnaître. Nous n'aurons pas à retenir l'adresse de mémoire, nous allons juste indiquer des noms de variables à la place.
Exemple :
Dans un programme écrire « a = 5 » veut dire : « réserve une case mémoire (à l'adresse xxxxx, mais cela, le programmeur s'en moque un peu !) que l'on va appeler a, et range la valeur 5 dans cette case mémoire » (on peut aussi ranger la chaîne de caractères « bonjour » dans une autre case mémoire appelée b) on aura le programme « b='bonjour' »
Le nom donné à une variable peut être composé de plusieurs lettres (exemple : « nbre_de_vie = 5 »)
Les noms de variables doivent obéir à quelques règles simples :
Un nom de variable est une séquence de lettres et de chiffres, qui doit toujours commencer par une lettre. Seules les lettres ordinaires sont autorisées. Les lettres accentuées, les cédilles, les espaces, les caractères spéciaux tels que $, #, @, etc. sont interdits, à l'exception du caractère _ (souligné).

La casse est significative (les caractères majuscules et minuscules sont distingués).
Attention : Variable, variable et VARIABLE sont donc des variables différentes.
Il faut prendre l'habitude d'écrire l'essentiel des noms de variables en caractères minuscules (y compris la première lettre). Il s'agit d'une simple convention, mais elle est largement respectée.
Nous pouvons utiliser les majuscules à l'intérieur même du nom, pour en augmenter éventuellement la lisibilité, comme dans tableDesMatieres, par exemple.

Comme déjà dit ci-dessus il existe différents "types" de variables :

- Les variables qui permettent de stocker des nombres entiers
- Les variables qui permettent de stocker des nombres à virgule flottante
- Les variables qui permettent de stocker des booléens (vrai ou faux)
- Les variables qui permettent de stocker des caractères
- Les variables qui permettent de stocker des tableaux
- Les variables qui permettent de stocker des chaînes de caractères

Dans certains langages, le programmeur doit définir le type de variable au moment de la déclaration (création) de cette dernière, on parle de langages "fortement typés" (exemples : C, C++, Java,.....)
En effet les différents types de variables n'occupent pas la même "place" en mémoire. En définissant le type de la variable, le programmeur "réserve" la quantité de mémoire nécessaire pour "accueillir" sa variable.

Dans d'autres langages (JavaScript, python,....) cela n'est pas nécessaire, on parle de langages "peu typés". Les variables ont ici aussi un "type", mais le programmeur n'a pas besoin de le préciser au moment de la déclaration.

Exemples de déclaration d'une variable en Java :
public int i = 1 ;
Ne vous préoccupez pas du "public" et du ";" en revanche le "int" signifie que la variable i est de type **int**eger (nombre entier)
public String monMot = "Hello World" ;
Ici la variable "monMot" est de type String (chaîne de caractères)

exemples de déclaration d'une variable en JavaScript :
var i = 1 ;
i est de type "nombre entier" mais cela n'est pas précisé lors de la déclaration
var monMot = "Hello World" ;
"monMot" est de type chaîne de caractères

Les fonctions

L'un des concepts les plus importants en programmation est celui de fonction.
Les fonctions permettent en effet de décomposer un programme complexe en une série de sous-programmes plus simples, lesquels peuvent à leur tour être décomposés eux-mêmes en fragments plus petits, et ainsi de suite. D'autre part, les fonctions sont réutilisables : si nous disposons d'une fonction capable de calculer une racine carrée, par exemple, nous pouvons l'utiliser un peu partout dans nos programmes sans avoir à la réécrire à chaque fois.
La notion de fonction en informatique est comparable à la notion de fonction en mathématiques.
Si nous avons y = 3x+2 , pour une valeur donnée de x, on aura une valeur de y.
exemple : x=4 donc y= 14 (y = 3x4+2=14, attention ici x correspond au signe "multiplié")

La fonction en informatique est basée sur la même idée :
valeur de départ (appelée paramètre) => fonction => valeur retournée par la fonction (grâce au mot clé return)

Voici la syntaxe employée par le langage Python (quelque soit le langage, le principe est le même) la définition d'une fonction est la suivante :

```
def nomDeLaFonction (liste des paramètres):
.....
instruction qui compose la fonction
.....
return y
```

La fonction retournera la valeur contenue dans la variable y.

Codons notre exemple (y=3x+2) en créant une fonction "maFonction" toujours en Python :

```
def  maFonction (x):
        y=3*x+2
        return y
```

Pour "utiliser" la fonction "maFonction", il suffit d'écrire :
maFonction (4) (dans ce cas précis, notre fonction renverra le nombre 14)
Il faut savoir qu'au moment de l'exécution de votre programme le code "maFonction(4)" sera systématiquement remplacé par la valeur retournée par la fonction (toujours dans notre exemple le "maFonction(4) sera remplacé par le nombre 14). En Python un "print maFonction(4)" affichera à l'écran "14".

En Java, cette même fonction s'écrira :

```
public static int maFonction (int x) {
        int y = 3*x+2 ;
        return y ;
}
```

Ici aussi nous ne nous attarderons pas sur le mot clé "public" et le mot clé "static". En revanche, vous pouvez remarquer le mot clé "int" devant x et devant y, ceci est bien sûr dû à l'aspect fortement typé du Java. En Java, toute définition d'une fonction est encadrée par des accolades. Sinon, le principe est le même qu'en Python.
Le paramètre et la valeur retournée ne sont pas forcément des nombres entiers, cela peut-être des chaînes de caractères, des tableaux,.....
Reprenons un exemple en Java où le paramètre et la valeur retournée sont de type String

```
public static String maFonction (String nom) {
        String phrase = "Bonjour " + nom ;
        return phrase ;
}
```

Dans ce cas, le + est ici le signe qui permet la concaténation (d'après Wikipédia : Le terme **concaténation** (substantif féminin), du latin *cum* («avec») et *catena* («chaîne, liaison»), désigne l'action de mettre bout à bout au moins deux chaînes.)
Le code Java "System.out.println (maFonction ("Toto"))" permettra d'afficher à l'écran

"Bonjour Toto" (le System.out.println est l'équivalent en Java du print en Python).
NB1 : Attention tous les codes donnés ici ne sont pas directement utilisables.
NB2 : Malgré ce que cet exemple pourrait laisser penser, le paramètre et la valeur retournée ne sont pas forcément du même type

Il est possible de faire passer plusieurs paramètres à une fonction.

Voici un exemple en Python :

```
def  MaFonction (x,b):
        y=3*x+b
        return y
```

Un "print MaFonction (4,5)" affichera à l'écran "17"

Les paramètres peuvent être de type différent.

Il faut aussi savoir que la fonction ne retourne pas forcément de valeur (le mot clé return n'est pas obligatoire). Mais si une fonction ne retourne pas de valeur, que fait-elle ? Elle peut faire plein de choses, par exemple une fonction peut tout simplement afficher un texte. Voici un exemple en Java :

```
public static void afficheNomAge (String nom, int age) {
        System.out.println ("Bonjour "+nom+", vous avez "+age+" ans");
}
```

Vous avez sans doute remarqué l'absence du mot clé return. Cette fonction ne renvoie rien, en revanche, elle affiche une phrase à l'écran.
Un simple "afficheNomAge ("Toto",14) ;" affichera à l'écran : "Bonjour Toto, vous avez 14 ans"
En Java le nom d'une fonction qui ne renvoie rien devra être précédé par le mot clé void.

Les paramètres ne sont pas non plus obligatoires, un exemple en Python :

```
def  MaFonction ():
        print "Hello World"
```

Notion de variable locale

Dans la plupart des langages, une variable définie dans une fonction est utilisable uniquement dans cette fonction. En dehors de cette fonction, la variable n'existe pas !
Une variable définie dans une fonction est appelée variable locale.
Un exemple en Java :

```
public static maFonction () {
        int i=1;
}
```

En dehors de cette fonction un "System.out.println (i)" entraînera une erreur, i n'existe pas en dehors de la fonction "maFonction".