23 mars 2016

Paradigmes de programmation

Il existe des milliers de langages de programmation pour commander des machines (ordinateurs, téléphones, robots ...). Ils sont souvent classés par catégories, correspondant à des principes de fonctionnement différents : ces catégories sont appelées des *paradigmes de programmation*. Chaque paradigme suggère sa propre façon de concevoir les programmes. Pour illustrer cela, un exemple calculant le minimum d'une liste de nombres est donné ci-dessous dans les trois principaux paradigmes de programmation. Ces trois programmes reposent sur le même algorithme, qui consiste à parcourir la liste du début à la fin en mémorisant la plus petite valeur rencontrée au fur et à mesure.

Les langages impératifs : ils sont fondés sur une exécution étape par étape, on parle d'exécution « *séquentielle* ». Le programmeur décrit l'ordre selon lequel sont exécutées les instructions. Il a aussi la possibilité d'accéder directement à la mémoire contenant les données, ce qui permet d'écrire des programmes dont l'exécution est très rapide. Ce paradigme est le plus courant et historiquement le plus ancien, puisqu'il a été inventé conjointement à la création des premiers ordinateurs. Par exemple, les langages de programmation C, Pascal, Python, Fortran et COBOL font partie du paradigme impératif.

```
longueur=len(s)
if longueur == 0 : print ("La liste est vide")
else :
    mini = s[0]
    i = 1
    while i < longeuur :
        if s[i] < mini :  mini = s[i]
        i = i+1
    print ("Le minimum est : ", mini)
```

Exemple de programme Python qui calcule le minimum de la liste s.

Le programme Python commence par calculer dans la variable `longueur` le nombre d'éléments de la liste `s` grâce à la fonction `len(s)`. Ensuite le programme teste si la liste est vide en comparant sa longueur avec 0. Si c'est le cas, il affiche le message `La liste est vide`. Dans le cas contraire, il affecte à la variable `mini` la première valeur de la liste `s[0]` et initialise le compteur `i` à 1. Ensuite, tant que la valeur du compteur `i` est plus petite que la longueur de `s`, le programme compare la valeur de l'élément d'indice `i` de la liste `s` avec la valeur de `mini`. Si `s[i]` est plus petit que `mini` alors la variable `mini` prend pour nouvelle valeur `s[i]`. Ensuite le compteur `i` est augmenté de 1. Une fois la boucle terminée, le programme affiche la valeur de la variable `mini` qui est bien le minimum de la liste.

Les langages fonctionnels : ils sont fondés sur l'évaluation de fonctions, dont le résultat est à son tour utilisé pour évaluer d'autres fonctions. Leur proximité avec le formalisme mathématique est un atout majeur de ces langages. En effet, cette caractéristique permet de prouver plus facilement que dans le paradigme impératif que les programmes réalisent ce pour quoi ils ont été conçus. Cette propriété est très souhaitable pour simplifier le travail des programmeurs. De plus, des objets mathématiques comme les listes ou les fonctions sont omniprésents dans ces langages, ce qui permet d'écrire des programmes plus concis (environ dix fois plus courts qu'en C). Les langages de programmation tels que Haskell, Ocaml et Lisp font partie des langages de programmation les plus connus dans le paradigme fonctionnel. Ces langages sont de plus en plus utilisés car ils permettent d'écrire des programmes certifiés sans bug, puisque prouvables.

```
let rec minimum s = match s with
|[]   -> failwith "La liste est vide"
|[x]  -> x
|e::r -> let mini = minimum r in
         if mini > e then e
                     else mini
```

Exemple de programme OCaml qui calcule le minimum de la liste s.

La première ligne de ce programme OCaml définit une fonction récursive appellée `minimum` qui prend en paramètre une liste `s`. En fonction du contenu de cette liste trois situations se présentent :

- Si la liste est vide `[]` (seconde ligne) alors le programme affiche le message `La liste est vide`.
- Si la liste `s` vaut `[x]` (troisième ligne), elle contient un seul élément `x`. Dans se cas la fonction renvoie `x` qui est bien le minimum de la liste.

- Enfin si la liste `s` vaut `e::r` (quatrième ligne), elle est composée d'un élément `e` et de la liste du reste des éléments de `s` notée `r`. Dans ce cas, la fonction `minimum` est appelée récursivement sur la liste `r` afin de calculer le minimum de cette liste. Cette valeur est alors notée `mini`. Il suffit finalement de comparer la valeur de `e` avec `mini` pour trouver le minimum de la liste et donc de renvoyer soit `e` soit `mini` en fonction des cas.

La programmation logique : elle est fondée sur la logique formelle définie au début du XXème siècle pour traiter des problèmes posés par les fondements des mathématiques. Le langage emblématique de cette catégorie est Prolog. Le programmeur définit un ensemble de faits élémentaires et de règles de logique leur associant des conséquences. Il permet d'écrire des programmes très concis et élégants car proches de la description mathématique de l'algorithme à l'origine du programme. L'usage de ces langages reste toutefois confidentiel.

```
minimum([X], X).
minimum([E|Fin], Mini) :- minimum(Fin, MinFin),
                          Mini is min(E, MinFin)
```

Exemple de programme Prolog qui calcule le minimum d'une liste.

La première ligne de ce programme Prolog traite du cas où la liste contient un seul élément. Dans ce cas, le minimum de cette liste `[X]` est `X`. La suite du programme décrit le cas d'une liste contenant strictement plus d'un élément. Cette liste `[E|Fin]` est alors constituée d'un premier élément noté `E` et du reste de la liste noté `Fin`. Le programme calcule, de manière récursive, le minimum de la fin de la liste qui est stocké dans `MinFin`. Finalement, le minimum de la liste initiale, noté `Mini` est le minimum du premier élément `E` et de `MinFin`. Par exemple, l'exécution de la commande `minimum ([2,1,3],M)` calcule le minimum de la liste `[2,1,3]` à savoir `1` et stocke cet élément dans `M`.

Autres langages : Il existe d'autres catégories de langages que les trois paradigmes ci-dessus. Citons par exemple les langages événementiels : par opposition à la programmation séquentielle (impérative), dans ce cas, l'exécution est fondée sur la gestion d'événements (comme un clic de souris). La librairie swing en Java ou encore Node.js en JavaScript sont des langages événementiels. Il existe aussi des langages spécialisés, adaptés à des utilisations particulières : les langages orientés objet (Java), les langages pour pages Web (PHP, JavaScript), pour les bases de données (SQL) ou encore pour la pédagogie (LOGO, Scratch) et même des langages de programmation très exotiques [1].

L'objectif de ces exemples est uniquement d'illustrer quelques différences entre les trois paradigmes. Ainsi, on peut constater que le programme en Prolog est le plus concis, et que les syntaxes diffèrent radicalement entre les différents langages. Dans tous les cas, on note que la connaissance de chaque langage serait cruciale pour comprendre en détail le fonctionnement de ces différents programmes, pourtant issus du même algorithme.

1. Voir par exemple `http://en.wikipedia.org/wiki/BrainFuck`.