1 Programmation impérative

La présentation des algorithmes en cours suit souvent les principes de la programmation impérative :

- on définit un état de la machine au début du calcul, en particulier on initialise des variables ;
- puis on décrit les étapes élémentaires qui modifient le contenu des variables, dans un ordre bien défini (séquentiellement) ;
- enfin on récupère dans certaines variables le résultat voulu

La plupart des langages de programmation sont de type impératif essentiellement : C, Python, Perl, Php sont de ce type, même s'ils permettent de pratiquer d'autres styles de programmation.

Exemple typique : la somme des éléments d'un tableau.

```
t : tableau de n nombres
---
s <- 0
pour i variant de 0 à (n-1) faire
  s <- s + t[i]
finfaire
retourner s
```

Il revient au concepteur de l'algorithme de tout organiser. En particulier, il lui revient la charge de connaître dans quel état se trouve la machine à la fin d'un calcul, avant d'en relancer un autre, pour éviter les « effets de bord ».

En particulier, la programmation impérative modifie l'état interne de la machine.

2 Programmation fonctionnelle

Il existe d'autres styles de programmation (des « paradigmes »). L'un des premiers proposés au début de la réflexion sur le calculable fut le style fonctionnel. Malheureusement, il a été longtemps inutilisable en pratique à cause des capacités insuffisantes des machines, laissant le champ libre à la programmation impérative.

Avec le développement des technologies et surtout de profondes avancées dans le domaine de la recherche en informatique (gestion de la mémoire, ramasse-miettes, compilateurs efficaces, etc), la programmation fonctionnelle a resurgi dans les années 1980 et a pleinement été acceptée depuis les années 1990, notamment grâce aux travaux des chercheurs en langage ML. Des langages efficaces ont alors été développés : CAML, Haskell, etc.

Les principes de la programmation fonctionnelle sont les suivants :

- pas de variables modifiables, juste des alias : à un nom de symbole correspond une valeur qui ne pourra pas changer durant le calcul, on appelle alors « liaison » le fait de donner un nom symbolique à une valeur (ce n'est pas une affectation au sens de la programmation impérative) ;
- pas de boucles, qui par nature, servent à modifier des variables ;
- tout est calculé dans le cadre de fonctions : on définit des fonctions puis on les appelle avec des paramètres pour obtenir le résultat voulu ;
- tous les objets calculables le sont par des fonctions, en particulier les fonctions elles-mêmes peuvent être paramètres d'autres fonctions et être le résultat de calculs ;
- le principe fondamental de la programmation fonctionnelle est un principe sanitaire : quand le calcul est terminé, la machine a retrouvé son état initial.

Il paraît étonnant qu'on puisse se passer des variables et des boucles, mais c'est pourtant le cas. Ceci dit, il ne faut pas tomber dans le jusqu'au-boutisme : si un algorithme est naturellement plus clair en programmation impérative, on privilégie bien sûr ce style.

Retenons ces principes :

- les variables sont remplacées par les paramètres de fonctions ;
- les boucles sont remplacées par les fonctions récursives.

Le but, entre autres, de l'option informatique MPSI/MP est de vous amener à découvrir ce style de programmation, ses avantages et ses inconvénients.