Chapitre 4

Les proverbes du programmeur

Cette partie a été empruntée au livre de Raymond Séroul :
Informatique pour les mathématiciens

4.1 Surtout, mais surtout pas d'astuces !

Ce proverbe - *peut-être le plus important de tous* - choque toujours le mathématicien qui n'a jamais programmé ; il choque aussi les **programmeurs débutants qui ont toujours tendance à *bidouiller*.** N'en déduisez surtout pas que les informaticiens sont des imbéciles ! Vous comprendrez la profondeur de ce conseil le jour où vous commencerez à écrire des programmes importants.
Ce qui comptera alors, ce sera un programme clair, qui fonctionne tout de suite (ou presque...) et **facile à maintenir**.
Un informaticien passe beaucoup *(trop)* de temps à maintenir des programmes, ce qui veut dire modifier un programme qui a été écrit -*en général*- par une personne qui n'est plus là.
Un programme professionnel est très long : plusieurs centaines de pages. Lire un programme est un exercice très difficile *(c'est aussi atroce qu'essayer de comprendre une démonstration réduite à de simples calculs et dépourvue d'explications : **Commenter un programme,** c'est comme rédiger un devoir de maths et réciproquement).*
En outre, le temps est compté ; le programmeur qui modifie un programme ne peut pas se permettre de passer des heures à se demander ce que signifie le code qu'il a sous les yeux. *Plus le code est bête* **c'est-à-dire limpide, sans détours**, plus il est sûr et plus il est facile à modifier.
Le gros reproche que l'on peut faire aux astuces est qu'elles sont la plupart du temps inutiles.
On estime qu'en général un programmeur passe plus de 80 % de son temps dans moins de 20 % du code. Il en résulte qu'une astuce a de très fortes chances d'intervenir dans une partie du code où la machine ne fait qu'attendre *(quand on entre des données au clavier par exemple)*. Il est stupide et suicidaire de fragiliser un programme en utilisant une astuce pour faire gagner quelques millisecondes à un programme qui tourne mille fois ou cent mille fois moins vite parce qu'il vous attend
Faites-vous violence : **choisissez toujours le code le plus bête,** même si cela exige quelques lignes supplémentaires.
La modestie paye toujours à la longue. Ceci dit, le recours à une astuce est parfois indispensable, par exemple dans une boucle sollicitée en permanence dans un programme trop lent. Si c'est le cas **documentez !** Avertissez votre lecteur, expliquez-lui en détail ce que vous faites et pourquoi vous le faites *(la machine est trop lente, elle n'a pas assez de mémoire, etc.)*. N'oubliez pas que ce lecteur, ce sera peut-être vous dans quelques mois avec une machine et un point de vue différents...
Que préférez-vous ? :

- Un programme facile à mettre au point et que vous aurez plaisir à optimiser ensuite,
- Ou un programme illisible, plein de fautes et qui ne marchera qu'après des heures et des heures de déverminage ?

4.2 On ne mâche pas son chewing gum en montant un escalier

Ce proverbe -*tiré d'une campagne électorale américaine*- exprime une idée de bon sens :

on ne fait bien qu'une seule chose à la fois.

C'est pour cela qu'on

découpe un programme en procédures ou fonctions et en modules indépendants.

4.3 Nommez ce que vous ne connaissez pas encore

Ce proverbe s'applique chaque fois que vous rencontrez un sous-problème à l'intérieur d'un problème. Refusez de résoudre le sous-problème *(proverbe précédent)*, laissez-le de côté et avancez. Comment ? En donnant un nom à ce que vous ne connaissez pas encore *(du code, une fonction)*.
Cela vous permettra de terminer le travail *(i.e. le problème)* en cours. Pour le sous-problème, voyez le proverbe suivant.

4.4 Demain sera mieux, après demain, encore mieux !

Non, ce n'est pas un éloge de la paresse ! Il s'agit au contraire d'une technique extraordinaire pour être efficace.
Appliquez ce proverbe chaque fois que vous constatez un blocage provoqué par l'apparition d'un nouveau problème à l'intérieur du problème que vous essayez de résoudre : appliquez le proverbe précédent en remettant la solution du nouveau problème à demain en lui donnant un nom sous la forme d'une procédure ou d'une fonction dont vous écrirez le code plus tard.
Séparez toujours ce qui est urgent de ce qui ne l'est pas ;, apprenez à distinguer l'essentiel de l'accessoire ; ne vous noyez pas prématurément dans les détails. Les détails, vous vous en occuperez demain, ce qui signifie quelques minutes ou quelques heures en général.
Il ne s'agit pas de reprendre le travail 24 heures plus tard comme aimeraient le croire certains paresseux ! Cette technique permet d'avancer petit à petit. Vous l'avez certainement pratiquée en mathématique :

Je vais d'abord prouver mon théorème en admettant provisoirement les lemmes 1, 2 et 3.

Ce proverbe nous met aussi en garde contre un défaut de débutant : vouloir agir tout de suite en écrivant prématurément un code très technique. La bonne attitude est à l'opposé : il faut cultiver une certaine nonchalance en donnant des ordres aujourd'hui ; le reste se traitera demain. « Hâtez-vous lentement ! » dit un autre proverbe.

4.5 On n'exécute jamais un ordre avant de le donner

Un débutant est toujours trop pressé : il écrit, il écrit... Le résultat est un code trop riche, trop technique, trop long et donc forcément incompréhensible. Allez trouver ensuite la faute dans ce fatras !
Ce défaut est très facile à mettre en évidence. Si pour comprendre un code vous devez entourer certaines parties d'un rectangle auquel vous attachez une explication, vous pouvez être certain qu'il manque une procédure *(l'ordre, si vous préférez)* à cet endroit.
Remplacez cette partie de code par un appel de procédure. Vous « exécuterez » cet ordre plus tard, lorsque vous écrirez le code de la procédure. Sachez donc vous retenir...

4.6 Documentez aujourd'hui pour ne pas pleurer demain

Imaginez une démonstration réduite à des calculs et quelques symboles de logique : elle est illisible, donc inutile. [1] Quand vous programmez, pensez à expliquer très précisément ce que vous faites.

- Tout d'abord cela vous oblige à comprendre ce que vous voulez entreprendre :

1. Grand défaut habituel des étudiants

ce qui se conçoit bien s'énonce clairement

dit la sagesse populaire. Si vous n'arrivez pas à expliquer votre code à un camarade, vous pouvez être certain que vos idées sont approximatives et que votre programme est vraisemblablement incorrect.

Pour prendre conscience, pour éclaircir vos idées, dialoguez avec vous-même. Le meilleur moyen d'y parvenir est de vous contraindre à écrire les commentaires au fur et à mesure de votre progression ; n'attendez pas que votre programme soit terminé, ce serait trop tard.

Les mathématiciens ont compris ce mécanisme de prise de conscience depuis longtemps : ils rédigent soigneusement leurs démonstrations avant d'y croire vraiment.

- Si votre programme est faux, ou si vous devez le reprendre six mois plus tard pour le transformer ou recycler une procédure, vous serez bien content de trouver les explications qui vous indiqueront comment le programme a été conçu.

4.7 Le discours de la méthode de Descartes

Le Discours de la méthode de Descartes a été publié en 1637. C'est un texte fascinant,important, car les méthodes de programmation moderne s'en inspirent !

Comme la multitude des lois fournit souvent des excuses aux vices, en sorte qu'un état est bien mieux réglé lorsque, n'en ayant que fort peu, elles y sont fort étroitement observées, ainsi, au lieu de ce grand nombre de préceptes dont la logique est composée, je crus que j'aurais assez des quatre principes suivants, pourvu que je prisse une ferme et constante résolution de ne manquer pas une seule fois à les observer.

Le premier était de ne recevoir jamais aucune chose pour vraie que je ne la connusse évidemment être telle ; c'est-à-dire **d'éviter soigneusement la précipitation et la prévention,** et de ne comprendre rien de plus en mes jugements que ce qui se présenterait si clairement et si distinctement à mon esprit que je n'eusse aucune occasion de le mettre en doute.

Le second, de diviser chacune des difficultés que j'examinerais en autant de parcelles qu'il se pourrait et qu'il serait requis pour les mieux résoudre.

Le troisième, de conduire par ordre mes pensées, en commençant par les objets les plus simples et les plus aisés à connaître, pour monter peu à peu comme par degrés jusques à la connaissance des plus composés, et supposant même de l'ordre entre ceux qui ne se précèdent point naturellement les uns les autres.

Et le dernier, de faire partout des dénombrements si entiers et des revues si générales, que je fusse assuré de ne rien omettre.