Gilles Dowek

Les principes des langages de programmation

L'auteur tient à remercier François Pottier, Philippe Baptiste, Julien Cervelle, Albert Cohen, Olivier Delande, Olivier Hermant, Ian Mackie, François Morain, Jean-Marc Steyaert et Paul Zimmermann pour leurs remarques sur un premier brouillon de ce document.

Introduction

Nous connaissons des algorithmes depuis des millénaires, mais cela fait à peine quelques décennies que nous écrivons des programmes. Une grande différence entre l'époque d'Euclide ou d'Érathostène et la nôtre est que, depuis le milieu du XX^{e} siècle, nous exprimons les algorithmes que nous concevons dans des langages formels : les langages de programmation.

Les informaticiens ne sont pas les seuls à utiliser des langages formels. Les ophtalmologistes, par exemple, prescrivent des lunettes en utilisant des expressions très rigoureuses, comme « OD : -1,25 (-0,50)180° OG : -1,00 (-0,25)180° », dans laquelle les parenthèses sont essentielles. De nombreux langages formels de ce type ont été développés au cours de l'histoire : la notation musicale, le langage algébrique, ... En particulier, de tels langages ont été utilisés depuis longtemps pour commander des machines, comme les métiers à tisser ou les carillons des cathédrales.

Cependant, jusqu'à l'apparition des langages de programmation, ces langages n'ont eu qu'une importance secondaire : ils étaient limités à des usages très particuliers, ils n'avaient qu'un petit nombre de spécialistes et les textes écrits dans ces langages restaient relativement courts. Cette situation a changé avec l'apparition des langages de programmation, qui ont un spectre d'applications beaucoup plus étendu que la prescription de lunettes ou la commande d'un métier à tisser, qui sont utilisés par de vastes communautés et qui ont permis d'exprimer des programmes de plusieurs centaines de milliers de lignes.

L'apparition des langages de programmation a donc permis la conception d'objets artificiels, les programmes, d'une complexité sans commune mesure avec les objets conçus auparavant, comme les locomotives à vapeur ou les postes de radio. Ces programmes ont, en retour, permis la conception d'autres objets très complexes, comme des circuits intégrés constitués de centaines de millions de transistors ou des démonstrations mathématiques formées de centaines de

milliers de pages. Il est tout à fait surprenant que l'on ait réussi à écrire des programmes aussi complexes dans des langages organisés autour d'un si petit nombre de constructions — affectation, boucle, ... —, c'est-à-dire dans des langages à peine plus riches que le langage de prescription des lunettes.

Les textes écrits dans les langages de programmation présentent enfin la nouveauté d'être compréhensibles à la fois par les hommes, ce qui les rapproche des partitions utilisées par les organistes, et utilisables par des machines, ce qui les rapproche des cartes perforées utilisées par les orgues de Barbarie.

L'apparition des langages de programmation a donc profondément changé notre rapport au langage, à la complexité et aux machines.

Ce livre présente une introduction aux principes des langages de programmation, qui utilise le langage Java comme support. Il s'adresse à des étudiants qui ont déjà une certaine expérience de la programmation. Il fait l'hypothèse ceux-ci ont appris la programmation empiriquement, dans un langage de programmation unique, différent de Java.

Le premier objectif de ce livre sera donc naturellement l'apprentissage des rudiments de Java. Cependant, connaître un langage de programmation unique ne suffit pas pour savoir programmer. Il faut, pour cela, non seulement connaître plusieurs langages, mais surtout être capable d'en apprendre rapidement un nouveau. Cela demande de savoir distinguer les concepts universels, comme celui de *fonction* ou de *cellule*, qui reviennent sous une forme ou une autre dans tous les langages de programmation, de la manière particulière dont ils sont utilisés en Java. Cela ne peut se faire qu'en comparant le langage que l'on apprend à d'autres. Dans ce livre, deux langages de comparaison ont été choisis : Caml et C. Le but n'est donc pas que les étudiants apprennent simultanément à programmer dans les trois langages, mais que la comparaison avec Caml et C leur permette de comprendre quels sont les principes autour desquels les langages de programmation sont organisés. Cette compréhension leur permettra d'acquérir par la suite, s'ils le souhaitent, une réelle compétence en Caml ou en C, ou dans n'importe quel autre langage de programmation.

Un autre objectif de ce livre est que les étudiants commencent à acquérir les outils qui permettent de décrire précisément la signification d'un programme. Cette précision est, en effet, le seul moyen de comprendre clairement ce qui se passe quand on exécute un programme et d'être capable de raisonner dans des situations trop complexes pour que l'intuition suffise. L'idée est ici de décrire la signification d'une instruction par une fonction opérant sur un ensemble d'états. Les objectifs sur ce point restent cependant modestes : les étudiants voulant poursuivre dans cette direction devront le faire ailleurs.

Le dernier objectif de ce livre est l'apprentissage des algorithmes de base sur les listes et les arbres. Ici aussi, les objectifs restent modestes : les étudiants voulant poursuivre dans cette direction devront également le faire ailleurs.