Benjamin Wack
Sylvain Conchon, Judicaël Courant, Marc de Falco, Gilles Dowek,
Jean-Christophe Filliâtre et Stéphane Gonnord

Informatique pour tous en classes préparatoires aux grandes écoles

Manuel d'algorithmique et programmation structurée avec Python

Nouveaux programmes 2013
Voies MP, PC, PSI, PT, TPC et TSI

 ISBN : 978-2-212-13700-2

EYROLLES

Table des matières

Avant-propos

Proposer un enseignement spécifique d'informatique à tous les élèves de classes préparatoires aux grandes écoles scientifiques était une nécessité.

L'informatique est omniprésente dans le monde actuel. Chacun en a sa représentation personnelle, enthousiaste ou méfiante, superficielle ou pointue. Pour comprendre en profondeur ce qu'on entend par *informatique*, il faut commencer par clarifier ce qu'elle n'est pas.

Les anglophones la nomment souvent *computer science*, ce qui est un double contresens. Premièrement, parce que l'ordinateur n'est pas qu'une machine à calculer (*to compute* en anglais), en tout cas pas au sens où on l'entend dans le langage courant. Certes, les premières machines comme l'ENIAC ou l'EDVAC étaient utilisées exclusivement pour le calcul de tables balistiques. Cependant, il n'y a qu'à regarder fonctionner quelques minutes un ordinateur de bureau pour voir la diversité des tâches qu'il peut réaliser. Même si elle n'était pas bien exploitée par manque de ressources, cette polyvalence était déjà présente dans les tout premiers ordinateurs. Comme leur nom français l'indique, ce sont plutôt des machines à *ordonner* l'information, capables de stocker, de manipuler et de transmettre efficacement n'importe quelles données pourvu qu'elles leur soient fournies dans un format adéquat. Plutôt que de parler de polyvalence, on parlera donc d'*universalité* : un ordinateur peut traiter les données de toutes les façons raisonnables que l'on peut imaginer.

Ensuite, parce que l'informatique n'est pas uniquement la science des ordinateurs : elle est avant tout la science de l'*information* et de son traitement automatique. Elle démontre que tout objet du monde réel, et de certains mondes abstraits comme les mathématiques, peut se traduire par une représentation numérique, certes souvent imparfaite, mais que la recherche ne cesse d'améliorer au fil des années. La question du codage de l'information est bien antérieure à l'invention des ordinateurs, du dénombrement des moutons à l'aide de petits cailloux jusqu'au code morse, en passant par les différents systèmes d'écriture et de numération ou la programmation de motifs complexes dans les métiers à tisser.

Les ordinateurs n'ont fait que confirmer *a posteriori* la validité et l'utilité de ces représentations. Grâce à leur extraordinaire puissance de calcul, ils ont également donné en moins de cent ans un essor inédit à des modes de raisonnement patiemment élaborés pendant plusieurs millénaires. Cette pensée *algorithmique*, qui consiste à établir une méthode systématique pour résoudre un problème, était déjà connue des Mésopotamiens. Elle a trouvé une application directe lorsqu'il a fallu exprimer des façons de réaliser des processus complexes pour des machines très rapides mais dénuées d'initiative et de compréhension.

À peine née, la science informatique s'est confrontée à l'un de ses plus grands défis, qui l'occupe encore aujourd'hui : mettre au point des *langages* de communication communs à l'homme et à la machine. Comme les langues naturelles, ceux-ci doivent permettre une compréhension mutuelle entre deux entités ayant chacune leur propre représentation du monde, mais à un degré bien plus important puisqu'aucune place ne peut être laissée à l'ambiguïté.

L'informatique est donc bien plus qu'une technologie de pointe, c'est la science qui réunit ces quatre concepts de machine, d'information, d'algorithme et de langage et qui, les faisant travailler ensemble, leur a donné la place qu'ils occupent aujourd'hui dans nos sociétés.

L'étudiant qui se destine à une carrière d'ingénieur, d'enseignant ou de chercheur ne peut se dispenser d'une formation dans ce domaine tant il est incontournable dans presque toute activité professionnelle, en particulier dans les activités à caractère scientifique. En outre, cette formation ne peut se réduire à l'utilisation technique de logiciels : son contenu serait à la fois peu pérenne à cause de l'évolution rapide des outils, et d'une utilité très incertaine en fonction du parcours professionnel que l'on poursuivra.

Bien apprendre l'informatique demande surtout d'en saisir les concepts sous-jacents, qui restent valables malgré l'évolution des technologies. Pour ne donner que quelques exemples, l'architecture globale des ordinateurs n'a pas changé depuis plus de 60 ans ; les premiers ordinateurs représentaient déjà l'information en binaire ; les algorithmes de calcul numérique présentés dans cet ouvrage sont tous connus depuis le XVIII^e siècle au moins ; les langages de programmation reposent presque tous sur les mêmes cinq instructions fondamentales. Il importe donc d'acquérir une culture informatique solide plutôt qu'un vernis technologique ; cela ne dispense pas de mettre très régulièrement en pratique les notions que l'on découvre, l'informatique ne pouvant s'apprendre qu'accompagnée d'une expérience régulière de programmation.

Enfin, même en dehors de tout cadre professionnel, le jeune citoyen qu'est l'élève de classe préparatoire sera fréquemment amené à rencontrer l'informatique dans sa vie quotidienne que ce soit par le biais des réseaux de communication, de ses loisirs, de ses achats, de ses interactions avec les administrations, etc. Comprendre comment fonctionnent ces systèmes et, à plus forte raison, en avoir soi-même programmé, même à un niveau modeste, est une clé indispensable pour en profiter en tant qu'acteur et pas seulement en tant que consommateur.

Structure de l'ouvrage

Le contenu de ce manuel se veut fidèle au programme officiel des deux années de classes préparatoires aux grandes écoles scientifiques. Il aborde les différentes notions qui sont pertinentes dans une formation scientifique, toujours avec la préoccupation de les replacer dans le contexte plus général de la science informatique. L'ensemble du contenu a vocation à être réutilisé pour le développement des Travaux d'Initiative Personnelle Encadrés.

- Dans la première partie **Architecture matérielle et logicielle**, on aborde les mécanismes internes d'un ordinateur. On présente les modèles théoriques qui régissent son fonctionnement, le système d'exploitation qui en permet l'usage quotidien, et les grands principes d'un environnement de programmation (**chapitre 1**). On donne ensuite un premier aperçu de la traduction numérique de l'information via la représentation des nombres en machine (**chapitre 2**), qui aura des conséquences importantes lorsqu'on voudra effectuer du calcul numérique.
- Dans la deuxième partie **Algorithmique et programmation**, on présente les notions clés de l'algorithmique (**chapitres 3 et 4**) en s'attachant systématiquement à démontrer que les algorithmes que l'on écrit produisent le résultat attendu. On aborde également la traduction de ces algorithmes sous forme de programmes. On présente ensuite la notion de fonction (**chapitre 5**) qui permet d'organiser les programmes et leur développement. Ce chapitre présente également les fonctions récursives, qui font partie du programme de deuxième année. On montre enfin comment évaluer l'efficacité d'un algorithme, et on présente une première structure de données : les tableaux (**chapitre 6**).
- Dans la troisième partie **Ingénierie numérique et simulation**, on étudie la traduction dans un langage de programmation d'algorithmes numériques abordés en cours de mathématiques : le pivot de Gauss pour la résolution de systèmes linéaires (**chapitre 7**), les méthodes de dichotomie et de Newton pour la résolution d'équations sur les réels (**chapitre 8**) et la méthode d'Euler pour la résolution d'équations différentielles (**chapitre 9**). Ces méthodes numériques mettent en lumière les limitations introduites par le passage sur machine. On présente enfin une utilisation raisonnée de bibliothèques de calcul.
- Dans la quatrième partie **Bases de données**, on s'intéresse à une représentation de l'information à la fois plus complexe et plus en lien avec les applications industrielles, par le biais du modèle relationnel des bases de données. On montre comment exprimer, dans le langage de l'algèbre relationnelle, des requêtes de recherche d'abord simples (**chapitre 10**), puis faisant intervenir plusieurs relations (**chapitre 11**) et on aborde la traduction de ces requêtes dans le langage SQL.
- La cinquième partie **Algorithmique et programmation avancées** couvre, avec la section sur les fonctions récursives du chapitre 5, le programme de deuxième année. On y montre qu'il existe d'autres structures de données comme la pile (**chapitre 12**) et on y compare plusieurs algorithmes de tri (**chapitre 13**) du point de vue de leurs complexités.

- On conclut ce manuel par une série de neuf propositions de travaux pratiques (**annexe A**) et par une brève documentation pratique sur quelques fonctions utiles au traitement de fichiers et à la production d'images (**annexe B**).

Chaque chapitre contient trois types de contenus :

- une partie de cours ;
- des sections intitulées « Savoir-faire », qui permettent d'acquérir les capacités essentielles ;
- des exercices, avec leur corrigé lorsque nécessaire. Les exercices les plus difficiles sont marqués d'un ou deux astérisques (*).

Trois types d'encadrés jalonnent cet ouvrage : ATTENTION signale un piège ou une erreur fréquente chez les programmeurs débutants ; EN PRATIQUE mentionne des considérations d'ordre pragmatique ; POUR ALLER PLUS LOIN propose des ouvertures vers des questions hors-programme.

Des compléments numériques à cet ouvrage sont proposés sur le site compagnon : `http://informatique-en-prepas.fr`.

Avertissement

Lorsqu'on conçoit un enseignement d'informatique, la question du choix du langage dans lequel on va programmer est incontournable, bien qu'*in fine* ce choix n'ait pas d'importance et que les compétences acquises dans un langage soient pour la plupart facilement transposables à un autre.

Le programme officiel de cet enseignement donne d'emblée la réponse à cette question puisqu'il impose le langage Python. Or ce langage existe en plusieurs versions incompatibles entre elles (un programme écrit pour une version ne fonctionnera pas toujours dans l'autre). On distingue notamment les versions 2.x (celles dont le numéro commence par 2) des versions 3.x, puisque c'est à la version 3.0 qu'ont eu lieu des changements incompatibles. Dans cet ouvrage, tous les programmes sont écrits en Python 3.x ; lorsque cela est nécessaire, un encadré précise les changements à apporter pour faire fonctionner ces programmes en Python 2.x.

Remerciements

Les auteurs tiennent à remercier Serge Abiteboul, Luc Albert, Jean-Pierre Archambault, Bruno Arsac, Jean-Philippe Berne, Hugues Bersini, Christophe Boilley, Sylvain Delpech, Francis Dorra, Pascal Lafourcade, Guillaume Le Blanc, Vincent Massart, Jean-Marie Monier, Pierre-Étienne Moreau, Franz Ridde, Landry Salle, Marie-Dominique Siefert et Thierry Viéville pour leur aide précieuse au cours de la rédaction de ce livre ainsi que l'équipe des éditions Eyrolles, Anne Bougnoux, Laurène Gibaud, Sébastien Mengin et Muriel Shan Sei Fan.

Première partie

Architecture matérielle et logicielle

Dans cette partie, nous abordons les mécanismes internes d'un ordinateur. Nous présentons les modèles théoriques qui régissent le fonctionnement d'un ordinateur, le système d'exploitation qui en permet l'usage quotidien, et les grands principes d'un environnement de programmation (chapitre 1). Nous donnons ensuite un premier aperçu de la traduction numérique de l'information via la représentation des nombres en machine (chapitre 2), qui aura des conséquences importantes lorsqu'on voudra effectuer du calcul numérique.

Deuxième partie

Algorithmique et programmation

Dans cette partie, nous présentons les notions clés de l'algorithmique (chapitres 3 et 4) en nous attachant systématiquement à démontrer que les algorithmes écrits produisent le résultat attendu. Nous abordons également la traduction de ces algorithmes sous forme de programmes. Nous présentons ensuite la notion de fonction (chapitre 5) qui permet d'organiser les programmes et leur développement, ainsi que les fonctions récursives, qui font partie du programme de deuxième année. Nous montrons enfin comment évaluer l'efficacité d'un algorithme, et nous présentons une première structure de données : les tableaux (chapitre 6).

Troisième partie

Ingénierie numérique et simulation

Dans cette partie, nous nous intéressons à des méthodes numériques pour la résolution de systèmes linéaires (chapitre 7), d'équations sur les réels (chapitre 8) et d'équations différentielles (chapitre 9). D'une part, nous rappelons les méthodes du cours de mathématiques, qui sont faciles à programmer, et dont la validité peut être démontrée formellement et sans grande difficulté. D'autre part, nous expliquons comment utiliser les fonctions « clés en main » fournies par Python et ses bibliothèques adaptées, avec des exemples rencontrés en mathématiques, sciences physiques ou chimie.

L'idée pénible à garder à l'esprit est que, *en calcul numérique, tout est faux !* En effet, les données prises en entrée sont des approximations des données « réelles » (parce qu'issues d'autres calculs ou de mesures physiques). De plus, on résout des équations qui sont une approximation de la vie réelle (linéarisation d'un phénomène en physique...) et on applique pour cela des schémas qui introduisent une erreur dans le résultat. Enfin, le moindre calcul en arithmétique flottante induit des erreurs d'arrondis, on note le résultat final sur un morceau de papier, et on se trompe en copiant la deuxième décimale.
Nous montrons sous quelles conditions on peut contrôler ces erreurs, et comment choisir les paramètres des méthodes utilisées pour obtenir un résultat satisfaisant.

Quatrième partie

Bases de données

Dans cette partie, nous nous intéressons à une représentation de l'information à la fois plus complexe et plus en lien avec les applications industrielles, par le biais du modèle relationnel des bases de données. Nous montrons comment exprimer, dans le langage de l'algèbre relationnelle, des requêtes de recherche d'abord simples (chapitre 10), puis faisant intervenir plusieurs relations (chapitre 11) et nous abordons la traduction de ces requêtes dans le langage SQL.

Cinquième partie

Algorithmique et programmation avancées

Cette partie couvre, avec la section sur les fonctions récursives du chapitre 5, le programme de deuxième année. Nous y montrons qu'il existe d'autres structures de données telles que la pile (chapitre 12) et nous y comparons plusieurs algorithmes de tri (chapitre 13) du point de vue de leurs complexités.

Annexe

Travaux pratiques et compléments

Cette annexe doit amener l'étudiant à goûter les implications concrètes des différentes parties du cours, d'abord à travers des travaux pratiques — ludiques pour certains, allant de la dissection d'un ordinateur pour en manipuler les composants matériels, à l'écriture de procédures cryptographiques que l'on utilise quotidiennement parfois sans même en avoir conscience, en passant par des algorithmes de création et de manipulation d'images.
La dernière annexe sur les entrées/sorties précise comment écrire dans des fichiers, des images, et produire des tracés géométriques.

Index