Introduction

Ce livre a pour objectif de proposer une approche pédagogique de l'algorithmique. Il est structuré en deux grandes parties, la conception d'algorithmes et l'étude d'algorithmes existants.

La première partie traite de l'algorithmique et de l'analyse des données. Elle montre comment créer ses propres algorithmes, et présente la gestion de structures complexes comme les listes chaînées et les tableaux d'enregistrements ou d'objets qui servent de support à de nombreux algorithmes. Elle montre comment, à partir de l'analyse d'un problème, aboutir à un algorithme utilisant des mécanismes comme les boucles. Cette partie correspond à de l'algorithmique élémentaire. Elle permet d'assimiler les méthodes usuelles dans la conception logique des programmes, utiles à la compréhension d'algorithmes plus complexes.

La seconde partie présente des algorithmes connus. Elle en analyse le fonctionnement et détaille, pour chacun d'eux, les processus logiques utilisés. On y trouve, par exemple, des algorithmes de tri et de recherche.

Les exercices permettent au lecteur de mettre en pratique les notions présentées. Les algorithmes ainsi que les programmes C, C++ et Java associés sont fournis sur le site http://compagnons.pearson.fr/algorithmique, ce qui facilite le travail d'apprentissage en épargnant la saisie.

Qu'est-ce qu'un algorithme ?

Un algorithme est une *méthode logique de résolution d'un problème donné*, en vue d'être développé dans un langage de programmation. C'est la suite logique des actions à entreprendre pour aboutir à la solution finale, c'est donc… *un programme*.

Ce qui différencie un algorithme d'un programme écrit dans un langage de programmation est l'aspect *logique*. En algorithmique, le développeur se concentre sur la logique d'action pour produire un programme qui fournit une solution, en essayant de minimiser le temps de calcul et/ou l'espace mémoire occupé, sans se soucier des contraintes techniques de l'écriture du programme. La logique et l'efficacité du programme sont privilégiées. L'algorithme utilise souvent un langage simplifié (appelé pseudo-langage) pour décrire la méthode qui aboutit à la solution, et présenter la logique d'action. L'intérêt principal du pseudo-langage est de s'affranchir des contraintes techniques liées à un langage de programmation particulier. À l'inverse, ce n'est qu'un langage théorique, inventé pour les besoins de l'écriture de l'algorithme, qui ne peut pas être compilé directement.

L'étape finale du développement d'une application consiste à réécrire l'algorithme dans un langage de programmation pour produire un programme exécutable. Le travail porte alors sur la traduction de l'algorithme dans le langage choisi. Cette étape n'est que technique, car la logique du programme est déjà décrite dans l'algorithme.

L'algorithme est donc primordial, car ce sont les solutions qu'il propose qui seront programmées. La qualité et la rapidité du programme final découlent directement de l'algorithme.

Pseudo-langage et langages C, C++ ou Java comme support de développement

Le choix du pseudo-langage ou d'un langage de programmation particulier pour écrire l'algorithme est un débat qui n'est toujours pas tranché. Les partisans du pseudo-langage mettent en avant que la priorité de l'algorithme est de présenter la méthode qui aboutit à la solution, et qu'un langage de programmation risque de noyer le lecteur dans des détails techniques, et de spécialiser la solution selon les contraintes du langage choisi. Les adeptes du langage de programmation argumentent qu'un programme écrit en pseudo-langage ne peut être ni compilé, ni exécuté, ce qui lui enlève tout intérêt puisque l'utilisateur ne peut pas l'évaluer réellement. Nous pensons que c'est un faux débat, et que les deux méthodes sont complémentaires. Il est souhaitable de présenter la logique sous-jacente à l'algorithme en pseudo-langage pour se concentrer sur l'essentiel, et indispensable d'en proposer une version finale en langage de programmation pour permettre de le tester. Avec ces deux versions, le lecteur peut différencier la logique du programme de son aspect technique, ce qui lui laisse la liberté de le réécrire dans un autre langage que ceux qui sont proposés dans cet ouvrage.

De plus, un algorithme présenté uniquement en pseudo-langage revêt souvent une connotation théorique pour la plupart des étudiants qui ne perçoivent pas toujours son utilité, et surtout qui ne savent pas le traduire dans un langage de programmation. Il est donc important de présenter l'algorithme réécrit dans un langage de programmation usuel, sous la forme d'un programme qui peut être compilé et exécuté.

Cet ouvrage utilise les langages C, C++ et Java comme langages de développement. Ces langages présentent beaucoup d'avantages. Ils sont à la fois évolués et structurés, ce qui en font d'excellents supports pour l'apprentissage de la programmation en général, mais ils sont également très utilisés dans le monde industriel, scientifique ou Internet.

Les langages C++ et Java sont des langages objet. Ils permettent d'écrire plus facilement certains algorithmes qui portent sur des données structurées possédant leurs propres méthodes de traitement, comme les files, les piles ou les arbres. Cependant, cet ouvrage d'algorithmique n'a pas vocation à être spécifique à la programmation objet, les algorithmes présentés en pseudo-langage utilisent une programmation classique pour privilégier la logique du traitement, seuls les programmes sources C++ et Java en présentent une version objet.

Enfin, le langage C a fortement inspiré les concepteurs d'autres langages comme le PHP. Les techniques utilisées pour réécrire les algorithmes en langage C peuvent facilement être adaptées aux autres langages.

Malheureusement il serait fastidieux de présenter in-extenso les programmes sources des trois langages pour chaque algorithme décrit dans cet ouvrage. Les programmes sources sont donc téléchargeables sur le site des éditions Pearson. Nous présentons l'algorithme en pseudo-code ainsi qu'un exemple de compilation et d'exécution du programme source équivalent écrit en C, C++ ou Java.

Le fond et la forme

Cet ouvrage est conçu pour donner des bases d'algorithmique avant d'aborder des algorithmes plus complexes.

Chaque chapitre est divisé en deux parties, *Théorie* et *Exercices*. La partie théorique présente les notions algorithmiques et l'analyse des données. Elle comprend des figures et des encadrés qui insistent sur les points importants, et s'appuie sur des exemples. Elle se termine par un résumé qui rappelle les notions importantes étudiées. La partie *Exercices* reprend les notions présentées dans la partie *Théorie*. Les corrigés des exercices ainsi que des exercices complémentaires sont téléchargeables sur le site http://compagnons.pearson.fr/algorithmique.

Les exercices sont très importants. Leur rôle est d'asseoir l'apprentissage de l'algorithmique sur la pratique. Chaque développeur possède sa propre logique de programmation. Faire les exercices permet à chacun de trouver *sa* solution. Les solutions commentées aident le lecteur dans cette démarche.

Les programmes sources téléchargeables sont développés en respectant les langages C, C++ ou Java normalisés. Ils sont donc exploitables dans n'importe quel environnement de programmation. Le but recherché est de proposer des programmes facilement lisibles. Par conséquent, ils n'intègrent pas certaines astuces de programmation particulières à chaque langage, qui permettent de faire des codes plus courts. Les programmeurs avertis pourront donc encore améliorer ces programmes sources.

Prérequis

Cet ouvrage est conçu pour permettre l'écriture des algorithmes présentés dans n'importe quel langage de programmation. Cependant, le choix des langages C, C++ et Java comme supports de développement du programme final suppose la maîtrise d'un de ces langages de la part du lecteur, pour comprendre les codes sources (aucune description technique de ces langages n'est présentée dans ce livre).

Conventions

L'emploi de polices différentes permet de distinguer les syntaxes écrites en pseudo-langage, ou bien de faire ressortir un nouveau terme ou un concept important.

- Les instructions en pseudo-langage sont dans une police à espacement fixe, par exemple `tab[i]`.
- Les termes nouveaux ou importants sont en italique, par exemple *modélisation des données*.
- Dans les programmes donnés en exemples ou en exercices, les syntaxes nouvelles ou importantes sont en police à espacement fixe et en gras, comme la syntaxe suivante : **`Lire(article.prix)`**.
- Les exemples d'exécution des programmes C, C++ et Java présentent la saisie en gras pour la différencier de l'affichage.

Le contenu

La première partie de l'ouvrage traite de la conception d'algorithmes. Elle englobe les chapitres 1, 2 et 3.

Chapitre 1 : Environnement algorithmique et conventions. Il détaille le rôle de l'algorithme dans le développement des applications, et les règles à respecter pour son écriture.

Chapitre 2 : Les traitements logiques. Il présente les grands mécanismes utilisés dans les algorithmes comme les tests, les boucles et les sous-programmes. Il présente également la notion de complexité algorithmique.

Chapitre 3 : La gestion des données. Ce chapitre détaille les différents types de données qui servent de support aux algorithmes, comme les tableaux ou les listes chaînées.

La seconde partie de l'ouvrage, qui présente des algorithmes importants ou des méthodes algorithmiques spécifiques, est constituée des chapitres 4, 5, 6, 7, 8 et 9.

Chapitre 4 : La récursivité. Ce chapitre détaille le fonctionnement de la programmation récursive, et en présente l'intérêt.

Chapitre 5 : Les données abstraites. Ce chapitre présente les piles, les files et les arbres qui sont utilisés dans de nombreux algorithmes.

Chapitre 6 : Les tris. Ce chapitre porte sur les algorithmes de tri, qui sont répartis en tris élémentaires et en tris avancés.

Chapitre 7 : Les recherches. Ce chapitre présente quelques algorithmes de recherche. Cela va de la recherche séquentielle aux algorithmes plus sophistiqués, comme les tables de hachage.

Chapitre 8 : Les méthodes numériques. Ce chapitre présente, la mise en œuvre d'algorithmes mathématiques, tels que la recherche des racines d'une équation ou l'interpolation polynomiale.

Chapitre 9 : Les algorithmes classiques. Ce chapitre présente quelques algorithmes incontournables.

Les suppléments

Les programmes des exercices et des exemples au format source « .c », « .cpp » ou « .java » sont disponibles au téléchargement sur le site : http://compagnons.pearson.fr/algorithmique. Le lecteur pourra les utiliser pour son apprentissage, mais aussi les compléter et se constituer ainsi sa propre base d'exemples. Le site contient également les corrigés des exercices du livre et des exercices supplémentaires entièrement corrigés. Pour y accéder, le mot de passe est : **tdqKvB569Exn**

Remerciements

Je tiens à remercier Frédéric Jacquenod pour sa relecture avisée. Une attention toute particulière pour Sylvie, Guillaume et Julien qui ont fait preuve de beaucoup de patience.