Violeta Felea
Victor Felea

DUT ET LICENCE 1
SCIENCES ET TECHNIQUES
BTS SERVICES
INFORMATIQUES
AUX ORGANISATIONS

Introduction à l'informatique

Apprendre à concevoir des algorithmes

- Cours complet
- Raisonnement algorithmique
- Problèmes intégralement résolus
- Commentaires détaillés sur les solutions erronées

Vuibert

Table des matières

Préface

Objet de l'ouvrage

L'algorithmique représente, pour un informaticien, le premier pas à faire pour entrer dans le monde de l'automatisme des traitements. Elle est la base de la méthode utilisée dans le cycle de développement des programmes informatiques. Proposer une solution informatique à un problème donné, grâce à un programme informatique, est le but final de la programmation.

Mais il ne faut pas oublier que la réalisation d'un programme comprend plusieurs étapes, incluses dans le cycle de développement, chacune apportant une pierre à l'édifice. Cet ouvrage est dédié aux deux premières étapes, fondamentales dans le processus de développement d'une application informatique : l'analyse d'un problème et la conception d'algorithmes, qui répondent aux besoins du problème. Étapes essentielles, sans lesquelles le codage ultérieur, dans un langage de programmation, serait difficile à réaliser, notamment pour des problèmes complexes.

Volontairement, aucun langage de programmation n'est abordé dans cet ouvrage pour la phase de codage ; cette phase n'est pas considérée superflue, mais découle très naturellement des phases précédentes, d'analyse et de conception, quand elles sont bien approfondies et réalisées. À cause de la multiplicité des langages de programmation, il serait difficile d'inclure les détails, syntaxiques et sémantiques, de chacun d'entre eux. De même, la dernière étape du cycle de développement, celle du test de l'application, n'est pas traitée ici. Des méthodes de spécification de tests sont élaborées afin de détecter les différents types de défauts d'un programme. Des ouvrages dédiés peuvent être consultés à propos de ces sujets. Cependant, la base commune de tous ces langages est le mode de conception de la solution et il peut s'abstraire des détails d'implémentation. Ceci nous amène aux algorithmiques et à dédier cet ouvrage à leur conception.

L'objectif majeur est d'approfondir les notions de base de l'algorithmique (variables, expressions, structures de données, instructions, modélisation, techniques de recherche, de tri, etc.), pas seulement en présentant des définitions et les solutions aux problèmes, mais aussi en apportant des explications sur les solutions erronées ou inexactes. Ce dernier aspect reste essentiel dans l'apprentissage de l'algorithmique et n'est pas suffisamment présent dans les livres dédiés à l'initiation à l'algorithmique.

Ce livre a été inspiré par des enseignements dispensés en première année de licence, à l'Université de Franche-Comté - Besançon et en première année à l'EPU [1] de Lille, ainsi que par les discussions menées dans les équipes pédagogiques respectives.

Lectorat du livre

L'enseignement de la programmation, auprès des étudiants débutant en informatique, rencontre une difficulté majeure : comment structurer les idées de résolution d'un problème afin de les rendre traductibles en programme informatique ? L'enjeu n'est pas du tout dans la traduction purement syntaxique, mais dans la manière dont un problème est compris, celle dont son idée de résolution est trouvée et exprimée informatiquement. Il paraît évident quand on se présente devant un automate à café, de pouvoir commander et payer un café, mais il est moins évident de structurer le traitement qui permet de réaliser les actions du client. De même, il est très naturel de pouvoir réserver des billets d'avion en ligne, mais la mise en place de l'application est, de loin, plus difficile. Dans le domaine des mathématiques, le calcul du plus grand commun diviseur est une technique couramment utilisée, nécessaire dans de multiples problèmes. Comment pouvons-nous le rendre automatique, calculable par un ordinateur ? Ou encore, un problème beaucoup plus simple : vérifier la parité d'un nombre - quelle technique informatique pouvons-nous imaginer ?

Dans ce livre, des problèmes des plus simples aux plus compliqués, de divers domaines (mathématiques, économie, logistique, jeux, systèmes d'exploitation, etc.) s'adressent, à la fois :

- aux étudiants débutants, pour lesquels l'informatique reste encore un outil inconnu par son côté programmation,
- aux étudiants déjà passionnés par la programmation, pour lesquels la démarche d'apprentissage a été moins systématique,
- aux étudiants expérimentés aux outils de base, qui souhaiteraient aborder des problèmes plus complexes, orientés vers l'analyse, la modélisation et le découpage en traitements.

Comprendre pourquoi un algorithme n'est pas correct, ou pourquoi il accepterait des améliorations, fait partie de cet ouvrage apportant des solutions à tout public étudiant.

Les différents problèmes corrigés présentés, comme application de la notion abordée, constituent un bon entrainement. D'autres exercices, dont la correction n'est pas donnée dans cet ouvrage, sont proposés à titre d'approfondissement et ont, pour but d'encourager l'autonomie dans le travail.

Pour les enseignants en informatique, cet ouvrage présente un double intérêt :

- pédagogique – de par la présentation des notions de base liées à l'analyse et la conception d'algorithme, pouvant offrir un support pour la conception des cours,
- expérimental – grâce à la multitude d'exercices proposés, de difficulté graduée, corrigés ou non, pouvant aider à la préparation des sujets proposés aux étudiants.

[1] École Polytechnique Universitaire - actuellement Polytech

Très récemment, l'enseignement de spécialité optionnel "Informatique et science du numérique", en Terminale Scientifique, est dispensé en France. Parmi les domaines inclus, l'algorithmique. Ce livre constitue une bonne base pour la préparation de cours (et aussi la formation) des enseignants amenés à dispenser cet enseignement (souvent ayant une formation en mathématiques), ainsi qu'un support pour la préparation des fiches d'exercices pour les élèves. Les rappels et démonstrations mathématiques contenus dans l'ouvrage, en relation avec certains algorithmes, permettent de rendre concrètes les notions mathématiques, encore trop abstraites pour les lycéens.

Organisation de cet ouvrage

Les bases élémentaires de l'analyse des problèmes et de la conception algorithmique sont abordées de manière progressive, en sept chapitres. Le huitième chapitre regroupe des exercices. Le premier est dédié à une introduction générale, motivant l'intérêt de l'algorithmique et ses besoins. L'avant dernier chapitre se constitue comme une application de toutes les notions précédemment vues, sur des problèmes plus complexes, nécessitant une phase approfondie d'analyse et la conception de plusieurs algorithmes. Les autres chapitres présentent les outils de l'algorithmique, leur utilisation en solo, ou en combiné avec d'autres.

Voici un aperçu du contenu de chaque chapitre :

- Chapitre 1 - introduction de l'intérêt de l'algorithmique, dans le cadre du cycle de développement des programmes informatiques.
- Chapitre 2 - les notions de constantes, variables et expressions.
- Chapitre 3 - les instructions : les primitives (de lecture et d'écriture), l'instruction simple (d'affectation), les instructions de contrôle (le bloc d'instructions, les instructions conditionnelles et les instructions répétitives).
- Chapitre 4 - la structuration des algorithmes avec l'objectif de la modularité (les fonctions et les actions).
- Chapitre 5 - les types composés (les structures et les tableaux) permettant de représenter des informations multiples (homogènes et hétérogènes).
- Chapitre 6 - la mesure comparative des algorithmes (la complexité) et des techniques algorithmiques élémentaires, liées à la recherche d'informations, au tri d'informations et au principe récursif de résolution de problèmes.
- Chapitre 7 - processus d'analyse de problèmes complexes et de conception de solution algorithmique, en appliquant les notions introduites aux chapitres précédents.
- Chapitre 8 - exercices proposés, groupés selon les différentes notions présentées dans les chapitres précédents.

Les algorithmes sont rédigés en français, dans un pseudo-langage proche des langages procéduraux.

En plus des exemples qui accompagnent la présentation des notions, cet ouvrage inclut 81 exercices corrigés et 9 problèmes complets, ainsi que 65 exercices.

Conventions typographiques

L'ouvrage intègre, à la fois les explications des notions et des principes, ainsi que du pseudo-langage (pour les algorithmes). Afin d'en faire la distinction, ou pour mettre en évidence certaines idées, quelques conventions typographiques s'imposent :

- définitions de *notions* ou *introduction d'idées importantes* (Chapitre 1) : italique,
- `identificateurs de variables informatiques` : police de caractères de type machine à écrire (type `Courier`),
- *variables mathématiques* : italique,
- mise en forme des algorithmes : police de caractères de type machine à écrire (type `Courier`), petite taille ; les **`mots clés`** algorithmiques liés aux déclarations, primitives, instructions, aux début et fin d'algorithmes, ainsi que les classes de paramètres sont écrits en caractères gras. Le *titre des algorithmes* est en italique, ainsi que les littéraux ***`vrai`***, ***`faux`***,
- affichages/saisies : police à caractères linéaux (type Arial).

Remerciements

Nombreux sont ceux qui nous ont aidés à concevoir cet ouvrage.

Nous voudrions d'abord remercier nos professeurs, qui nous ont fait découvrir cette discipline, si enrichissante pour la formation d'un esprit scientifique.

Ensuite, nos remerciements vont à nos étudiants qui, inconsciemment, nous ont poussés à mettre en forme les concepts de cette discipline.

Puis, nous remercions nos collègues, pour la richesse de nos discussions.

Enfin, un merci particulier va à Jean-Pierre Steen, Professeur Honoraire en Informatique, de l'Université des Sciences et Technologies de Lille (Lille 1), pour les nombreux encouragements, conseils et critiques, qu'il nous a manifestés tout au long de la rédaction.

Et un autre remerciement va aux membres de notre famille, pour leur soutien pendant l'écriture de cet ouvrage. Connaissant les métiers de l'informatique et notre passion, ils ont trouvé la patience et la compréhension pour accepter tant de discussions, de soirées et de week-ends, dédiés à sa réalisation.

Besançon, le 1er septembre 2013

Violeta Felea, Maître de Conférences en Informatique, à l'Université de Besançon

Victor Felea, Professeur Honoraire en Informatique de l'Université Al. I. Cuza, Iasi, Roumanie

CHAPITRE 1

Algorithme - intérêt, définition, démarche

Les appareils informatiques et plus particulièrement les ordinateurs, ont été conçus pour être, d'abord, des machines à calculer. Les premières applications ont été du domaine des mathématiques et aussi de la comptabilité. Dans ces domaines, les nombres sont au cœur des manipulations. Avec le temps, tous ces calculs ont été programmés dans les ordinateurs et même dans les calculatrices de poche, qui sont les ordinateurs miniatures. Y voir la courbe d'une fonction est maintenant usuel [1].

Ceci sous-entendait de pouvoir enregistrer des nombres et de pouvoir les utiliser. Avec l'évolution des techniques (vitesse de calcul, capacité des mémoires, etc.), toutes les représentations de l'information (textes, images, sons, etc.) ont pu être numérisées, c'est-à-dire représentées par des nombres et donc, susceptibles d'être traitées par un ordinateur.

Les calculs s'effectuant sur ces nombres portent souvent un nom. Par exemple, calculer une moyenne, résoudre une équation, dessiner une courbe, etc. Beaucoup de ces calculs sont déjà intégrés dans les services offerts par un ordinateur. Pour les autres, il faut décrire le calcul à réaliser, qui sera ensuite communiqué à l'ordinateur. De manière générale, les différents calculs répondent aux besoins d'un problème posé.

La résolution d'un problème à l'aide d'un ordinateur est faite par l'humain en deux étapes :

- la première est la recherche d'une solution qui doit s'exprimer, puis se décrire, de façon très précise sous la forme d'un ensemble d'actions portant sur des informations - il s'agit d'une étape *conceptuelle* (qui induit le nom de *concepteur* pour la personne réalisant cette tâche),
- la deuxième est l'expression de cette solution, sous une forme compréhensible par un ordinateur, c'est-à-dire en l'exprimant dans un des nombreux langages de programmation existants - il s'agit d'une étape de *codage* (qui induit le nom de *codeur*).

La conception et le codage peuvent être réalisés par une même personne, appelée *développeur*.

[1] En même temps, il ne faut pas oublier que des applications particulières ont du être conçues pour permettre à ces appareils de fonctionner.

La première étape correspond à la description des calculs à réaliser : il s'agit de trouver la façon dont les calculs doivent être effectués et les noter de manière suffisamment formelle. C'est ce qui constitue les *algorithmes*. Afin d'élaborer ces algorithmes, une approche progressive est exigée : la solution, exprimée d'abord de façon très générale, est affinée en ajoutant, à chaque étape intermédiaire, de plus en plus de détails. La deuxième étape tient compte des contraintes liées au langage de programmation choisi et construit des *programmes*, qui correspondent à la traduction des actions algorithmiques selon les règles syntaxiques du langage.

Les programmes font partie d'un ensemble plus vaste d'informations traitées par l'ordinateur, appelés *logiciels*. Les logiciels peuvent être :

- *applicatifs*, destinés à l'usage des utilisateurs, appelés aussi *applications informatiques* : par exemple, logiciel de réservation de billets d'avion, ou de gestion de la comptabilité d'une entreprise ;
- *système*, destinés à contrôler l'utilisation de l'appareil informatique : par exemple, les systèmes d'exploitation d'un ordinateur.

Cet ouvrage a pour objectif l'apprentissage et la maîtrise de la démarche de résolution des problèmes posés, à l'aide de l'informatique, indépendamment d'un langage de programmation - donc seule la première étape présentée précédemment est abordée. La démarche aboutit à une organisation des informations et des actions. La succession des actions appliquées, pour aboutir au résultat souhaité, est donnée par l'intermédiaire des algorithmes. En termes informatiques, les informations sont appelées *données* et les actions sont décrites par l'intermédiaire des *instructions*. L'instruction permet d'exprimer une action dans une forme plus précise dans le cadre d'un algorithme.

Nous reprenons dans cet ouvrage une approche traditionnelle dans la démarche de résolution d'un problème à l'aide des algorithmes : le *principe procédural*. Ce principe décrit la marche à suivre pour donner une solution à un problème, c'est-à-dire l'enchaînement des différentes actions à entreprendre.

Pour présenter la démarche et afin de justifier, en partie, les notions introduites dans les chapitres suivants, nous prenons comme cas d'étude une classe de problèmes nécessitant l'interaction entre un usager et un automate piloté par un système informatique (réunissant à la fois l'ordinateur et les logiciels). Nous illustrons ici ce type de problèmes par deux exemples : les distributeurs de café et de billets. Ces exemples d'applications, apparemment simples, nous permettront d'identifier la démarche générale nécessaire à leur résolution, qui aboutira à l'écriture d'un programme informatique.

Afin de fournir une vision d'ensemble, de la démarche de programmation, nous concluons ce premier chapitre en donnant les étapes nécessaires à la rédaction de programmes informatiques.

1.1 Introduction des concepts sur une étude de cas : le distributeur de café

Considérons l'application simulant le fonctionnement d'un distributeur de café. Le service de base offert par de ce type de dispositif est de permettre à l'usager de préciser le type de café à lui fournir, de le payer et de récupérer la monnaie rendue, le cas

échéant. *Identifier ce qu'on doit attendre de l'application* constitue le point clé de la démarche de résolution. Dans le langage informatique, il s'agit ici de *dresser le cahier des charges*. Ce cahier des charges décrit ce que le commanditeur de l'application, appelé *maître d'ouvrage*, attend de celle-ci. Elle sera réalisée par le *développeur*, appelé aussi *maître d'œuvre* ou encore *programmeur*. Le développeur est un informaticien qui conçoit les logiciels.

Si les besoins attendus sont clairement identifiés dans le cahier des charges, il reste à la charge du maître d'œuvre de trouver la manière de répondre à ces besoins : concevoir la solution. Souvent, cela revient à proposer un découpage en *opérations élémentaires*, permettant de simplifier la complexité du problème traité. Ces aspects sont traités lors de *l'analyse du problème*.

Nous identifions tout de suite une caractéristique importante de ce problème : l'ordre dans lequel les opérations sont à réaliser. En effet, les opérations sont effectuées les unes après les autres, dans un ordre précis.

Ainsi, les actions à réaliser concernant l'achat de café suivent l'ordre suivant :

1. permettre à l'usager de choisir le type de café souhaité,
2. permettre à l'usager de donner une somme d'argent (celle-ci doit être suffisante pour payer le café choisi et seulement dans ce cas l'opération suivante est effectuée),
3. lui rendre la monnaie, le cas échéant,
4. donner à l'usager le café commandé.

Ici, nous simplifions volontairement le problème : d'abord, le choix de café peut être effectué parmi des saveurs bien identifiées (le niveau du sucre est fixé - pas de paramétrage possible), toutes les options sont valides - impliquant des quantités illimitées pour la matière première, ainsi que pour les différents types de monnaies ; ensuite, la somme d'argent donnée par l'usager se fait en une seule fois. Enfin, nous considérons, pour des raisons pratiques, que la restitution de la monnaie à l'usager se fait avec le minimum de pièces (ce qui est généralement le cas, mais dans certaines circonstances, ce principe peut ne pas être appliqué).

L'ordre est une propriété importante, car chacune des opérations reçoit une place bien déterminée dans la suite des actions à effectuer : la monnaie ne peut pas être rendue avant que le choix du café soit fait, par exemple. Bien évidemment, la monnaie pourrait être rendue après que le café soit prêt (mais pour des raisons pratiques, disposer des deux mains pour récupérer la monnaie, l'ordre logique est : le rendu de la monnaie, d'abord, puis la préparation du café), mais dès qu'une décision est prise sur la séquence de ces actions, tout comportement y sera conforme.

L'ensemble des actions à réaliser est communément appelé *traitement* en langage informatique. De la même façon qu'une action se décompose en une suite d'opérations, un traitement se décompose en un ensemble d'*instructions*, données dans un ordre précis, définissant une *séquence d'instructions*. L'ordre des actions identifiées d'un problème nécessite une structuration des traitements à réaliser, processus connu sous le terme d'*ordonnancement*. Une action est réalisée, ou effectuée, tandis qu'une instruction est *exécutée*. L'exécution d'une instruction permet, d'un point de vue algorithmique, d'effectuer l'action correspondante décrite d'une manière informelle.

Il est aussi important de marquer que chaque opération est réalisée dans un environnement précis.

L'environnement définit l'ensemble des informations qui sont traitées par l'application. Une information peut être imaginée comme un contenant. Elle doit pouvoir être désignée ou nommée - comme la somme d'argent donnée par l'usager - *et possède un domaine de définition*, qui est l'ensemble des valeurs possibles que l'information peut prendre. Une information possède généralement un *état*

- *défini* - comme la valeur de la somme d'argent après l'introduction des pièces dans l'automate,
- *indéfini* - comme la valeur de la somme d'argent avant l'introduction des pièces dans l'automate.

L'état de l'environnement est l'ensemble des états de chacune des informations le composant.

Les opérations effectuées interagissent sur l'environnement en modifiant, ou non, certaines, ou toutes ses informations (par modification de leur état). Par exemple, la somme d'argent donnée par l'usager est dans un état indéfini avant qu'il l'introduise, tandis qu'elle passe dans un état défini, ayant une valeur précise, après l'introduction des pièces.

On distingue dans l'environnement deux parties :

- la partie *interne* qui est constituée par les informations dont l'état ne peut être modifié que par des actions propres aux traitements,
- la partie *externe* qui est ce qui permet de faire le lien entre l'utilisateur de l'application et l'environnement interne. Ceci se fait par l'intermédiaire de deux types d'opérations spécifiques : les *saisies* (de l'utilisateur) et les *affichages* (voir la figure 1.1).

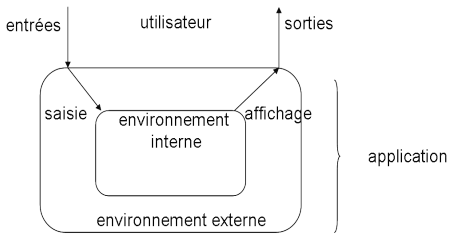

Fig. 1.1. Environnements dans l'algorithmique

Les saisies permettent de fournir les entrées à l'algorithme (les informations inconnues à l'avance, lors de la définition des traitements), tandis que les affichages permettent de faire connaître à l'utilisateur, les informations calculées au cours des traitements, sous la forme de sorties. Ces informations de l'environnement externe sont généralement caractérisées uniquement par un état défini, mais elles ne feront partie de l'environnement interne qu'au moment de l'association de cet état à une information désignable.

La somme de l'argent avancée par l'usager est une information de l'environnement interne, dont l'état change à la saisie de la somme, par l'intermédiaire de l'environnement externe. De la même façon, quand la monnaie est rendue à l'usager, ce dernier

s'en rend compte grâce à l'affichage de l'état de l'information liée à la pièce, par l'intermédiaire de l'environnement externe. Ici, il s'agit, encore une fois, de l'état qui est transmis à l'environnement externe pour affichage. L'affichage et la saisie constituent les premières actions nécessaires aux communications entre l'usager et l'application. De par leur propriété basique, celles-ci sont nommées *primitives* d'écriture, respectivement de lecture.

Synthèse Le cahier des charges permet d'identifier clairement ce qu'on doit attendre de l'application pour le problème posé. C'est le premier point clé de la démarche de résolution d'un problème. Le cahier des charges est un document contractuel décrivant ce qui est attendu du développeur (le maître d'œuvre) par le demandeur (le maître d'ouvrage). L'analyse en est un autre point clé, dans la démarche du développeur, qui consiste à identifier les informations manipulées, les nommer, décrire leur nature, définir la méthode adéquate de résolution du problème et formuler son principe.

Les besoins de l'application, identifiés jusqu'à présent, mettent en évidence deux concepts :

- l'environnement, contenant des informations ; chaque information est caractérisée par un nom, un domaine de définition et un état,
- la propriété d'ordonnancement des instructions.

Les instructions de base, permettant la communication entre l'usager et l'environnement, sont les primitives de lecture et d'écriture.

En termes informatiques, l'environnement correspond à la mémoire ; les informations correspondent aux variables/données, qui possèdent un nom, un domaine de définition et peuvent éventuellement avoir une valeur. Pour l'instant, nous avons vu l'utilité de disposer de primitives de lecture et d'écriture.

Regardons plus en détail le traitement nécessaire pour répondre à certaines des opérations, lors d'une commande de café.

- Permettre à l'usager de choisir le type de café voulu : il s'agit de la primitive de lecture, car plusieurs possibilités sont offertes à l'usager.
- Lui afficher le prix du café choisi : en fonction de son choix, les prix peuvent éventuellement être différents. Le traitement correspondant aussi bien pour la préparation de la commande, que pour le calcul de la monnaie à rendre, varie en fonction de ce choix. Nous identifions ici le besoin d'une *structure de contrôle* de l'action à entreprendre, variant en fonction du choix, structure appelée *instruction de sélection multiple.*
  Les structures (ou instructions) de contrôle permettent de définir un ordre sur les actions à entreprendre.
- Permettre à l'usager de donner une somme d'argent : il convient de permettre à l'usager d'introduire plusieurs pièces de monnaie dans l'automate, successivement, et cela, tant que la somme totale, correspondant aux pièces fournies, soit strictement inférieure au prix du café sélectionné. Il s'agit ici d'une autre structure de contrôle, permettant de réexécuter un ensemble d'instructions, *l'instruction répétitive.*

– Lui rendre la monnaie, le cas échéant : il s'agit ici de calculer, en fonction de la valeur du rendu calculé précédemment, le nombre de pièces à rendre. Le calcul s'effectue seulement si le montant à rendre est positif. Ceci traduit la nécessité d'utiliser une autre structure de contrôle, de sélection entre seulement deux choix. Cette structure est appelée *conditionnelle* ou *alternative*. Le calcul du reste à rendre est une opération mathématique classique de soustraction du prix de la commande de la somme donnée par l'usager. Pour effectuer ce calcul, nous devons nous appuyer sur l'information liée aux types de pièces disponibles. Cette information fait partie de l'environnement interne de l'application.
Certains traitements peuvent être formalisés par des expressions (comme notion scientifique) et des formules mathématiques. Le calcul du nombre minimal de pièces composant un montant donné utilise des opérations mathématiques élémentaires comme le quotient entier et le reste d'une division. Les divisions commencent par la plus grande coupure, en allant vers la plus petite, pour obtenir le nombre minimal de pièces.
Le nombre de pièces différentes possibles (huit dans la monnaie européenne : 1, 2, 5, 10, 20 et 50 centimes et 1, 2 euros) fait augmenter le nombre d'informations individuelles à manipuler. Ceci entraîne souvent des solutions peu lisibles et difficilement extensibles. Les informations liées au nombre de pièces pour chaque type de monnaie disponible sont homogènes (d'un même type). Généraliser le calcul pour beaucoup de monnaies différentes offre un caractère plus flexible à l'application. Dans ce cas, vu l'homogénéité des informations, une organisation particulière de ces données est nécessaire, concrétisée ici par le *tableau*. Ce moyen d'organiser les données et qui fournit aussi des opérations élémentaires de traitement sur cette organisation, définit les *structures de données*. Les *tableaux* en font partie.
Un tableau est aussi une information de l'environnement interne, information désignable et ayant plusieurs valeurs. Chacune de ses valeurs peut être obtenue par un *adressage indicé*. Il s'agit d'une des opérations élémentaires fournies avec cette représentation de données multiples. Ce concept permettrait l'application du principe de calcul de monnaie au cas des billets.
– La fonctionnalité de l'application est obtenue en regroupant toutes les actions citées ci-dessus dans un ordre précis : il s'agit de l'enchaînement des traitements à exécuter, grâce à une structure de contrôle, nommée : *bloc d'instructions*.

Synthèse Les techniques de résolution des problèmes nécessitent la considération des structures de contrôle : conditionnelle, de sélection multiple, répétitive, bloc d'instructions. Le formalisme mathématique peut aider à définir l'approche de résolution ; celui-ci impose la définition des expressions. De plus, certaines structures de données s'avèrent utiles.

Conception Plus l'analyse est approfondie, plus simple sera la traduction en langage de programmation. Pour faciliter cette traduction et pour formaliser le résultat de l'analyse, un ou plusieurs algorithmes sont conçus. Les algorithmes sont exprimés de façon structurée, commentée et à un haut niveau, en pseudo-langage indépendant d'un

langage de programmation. Plusieurs formalismes existent pour décrire les algorithmes. Parmi eux, nous avons choisi, pour exemplification dans cet ouvrage, le PDL (Pseudo Description Language), qui est un pseudo-langage textuel. Étant un formalisme, des règles d'écriture, de pseudo-langage, s'imposent, sans qu'elles soient aussi strictes que celles imposées par la syntaxe d'un langage de programmation. Celles-ci seront illustrées, au fur et à mesure de l'introduction des différents concepts, dans les chapitres suivants.

1.2 Étude de cas : le distributeur de billets

Considérons maintenant le cas d'un distributeur de billets. Nous identifions tout d'abord, la fonctionnalité de base de ce type de dispositif : l'obtention automatique de billets de banque à l'aide d'une carte bancaire et d'un code confidentiel. D'autres services sont généralement possibles sur ce même type de distributeur, mais nous ne les traitons pas en détail ici.

Cahier des charges L'objectif principal de l'application serait donc de pouvoir, à partir d'un code confidentiel (et de la carte, bien sûr), en précisant un montant souhaité (ne dépassant pas une limite maximale fixée par le contrat de la carte), obtenir des billets correspondant au montant demandé.

Analyse

Identifier les actions à entreprendre Une suite d'actions est nécessaire afin de fournir le résultat attendu (le schéma donné correspond à une simplification du cas réel) :

- permettre à l'usager de saisir un code confidentiel (ou éventuellement de le ressaisir, pas plus de trois fois généralement),
- vérifier la validité du code,
- en cas de code valide, permettre à l'usager de continuer l'opération d'obtention de billets, en précisant le montant souhaité,
- vérifier que le montant choisi respecte certaines règles (inférieur à une limite imposée par les clauses contractuelles de la carte),
- en cas de montant valide, demander à l'usager s'il souhaite un reçu, calculer les coupures à lui rendre, lui fournir les billets, puis le cas échéant, le reçu, et lui rendre la carte.

Ces opérations sont généralement accompagnées de validations ou d'annulations de la part de l'usager. Nous ne nous intéressons pas ici au principe appliqué pour calculer le nombre de coupures de chaque type de billet à rendre. Généralement, ce calcul est effectué en essayant de minimiser le nombre de coupures, pour la facilité et le confort de l'usager, mais cette règle n'est pas appliquée systématiquement, pour des raisons d'équilibre entre les coupures restantes. Il ne faudrait pas oublier qu'un distributeur est censé satisfaire plusieurs clients dans la journée, sans qu'un message du type "plus de coupures disponibles" soit renvoyé à l'usager.

Identification des données manipulées Dans l'application du distributeur de billets, l'environnement est déterminé par un ensemble d'informations :

- des informations liées à l'utilisateur :
  - le code confidentiel, saisi par l'usager,
  - le montant qu'il souhaite retirer,
  - le fait que l'usager souhaite recevoir un reçu ou non,
  - et qu'il souhaite annuler ou valider le montant saisi ;
- des informations liées au distributeur :
  - le code de la carte,
  - le crédit courant du compte,
  - la limite maximale du montant à extraire,
  - pour chaque type de coupure, le nombre de coupures à rendre et le nombre total de coupures disponibles pour ce type de coupure.

Analyse détaillée des actions

Dans la première étape du processus de retrait d'une somme d'argent, l'usager est invité à saisir le code d'accès de la carte. Cette opération peut être reprise un certain nombre de fois donné (généralement trois), si le code saisi ne correspond pas au code de la carte.

Il va donc falloir concevoir un système permettant de pouvoir reprendre la même opération (la saisie de code) pour un nombre de fois maximum donné, si le code n'est pas correct. Nous identifions donc l'instruction répétitive de saisie du code, dont le nombre de répétitions varie en fonction de la saisie de l'utilisateur.

La même démarche, répétitive (voir la page 5), est présente dans la saisie du montant à retirer. Ici, la seule contrainte est que le montant ne doit pas dépasser une limite maximale (généralement, aucune limite dans le nombre de saisies n'est imposée).

Une fois le montant saisi, l'usager doit indiquer s'il souhaite obtenir un reçu. Cette opération renvoie vers l'action d'édition de reçu, si la réponse est positive, ou, dans le cas contraire, directement vers l'action suivante, permettant le calcul des coupures. Une autre structure de contrôle permet de définir ce type de comportement : l'instruction alternative ou conditionnelle (voir la page 6).

Le calcul du nombre de coupures s'effectue de manière similaire au calcul de la monnaie à rendre, dans le cas d'un distributeur de café, avec une généralisation aux billets de différentes valeurs.

Si d'autres fonctionnalités sont intégrées (le rechargement de la carte du téléphone portable auprès d'un opérateur de téléphonie, la demande de relevé d'identité bancaire, la demande de chéquier, etc.), l'instruction alternative devrait permettre le choix entre plusieurs actions. L'instruction de sélection multiple (voir la page 5) permet, dans ce cas de figure, la multiplication des choix d'actions.

Synthèse Les primitives de lecture et d'écriture permettent, comme dans l'exemple précédent, d'assurer l'interface entre l'environnement externe et celui interne. Dans l'environnement interne, où se déroule tout le traitement nécessaire pour fournir les résultats attendus de l'application, les structures de contrôle, définissant un ordre des actions à entreprendre, sont inévitables : l'instruction alternative/conditionnelle, l'instruction de sélection multiple, l'instruction répétitive et le bloc d'instructions.

Les exemples, donnés dans ce chapitre introductif, ont pour but d'illustrer succinctement l'analyse d'une première étape appartenant au développement des applications, qui aboutit à la conception d'un algorithme. Les notions algorithmiques, comme les variables, ainsi qu'en termes d'actions, les primitives ou les structures de contrôle, apparaissent comme des outils indispensables pour la résolution des problèmes.

1.3 Démarche générale de programmation

Les sections précédentes ont illustré essentiellement l'analyse d'un problème à résoudre, proposé par le maître d'œuvre au maître d'ouvrage. Cette analyse constitue une des étapes de la démarche complète de programmation, que nous illustrons dans la figure 1.2.

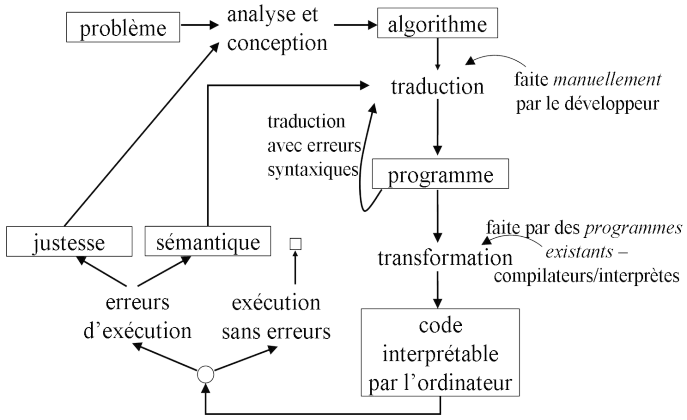

Fig. 1.2. Démarche complète de programmation

L'analyse permet d'identifier l'idée de résolution du problème, qui sera formalisée et exprimée au moyen des algorithmes, grâce à la conception. Ces algorithmes seront ultérieurement traduits en programmes, écrits dans un langage de programmation précis. Cette traduction n'est pas automatique, car elle exige la connaissance des règles d'écriture du langage de programmation, données par la syntaxe. Ces programmes restent textuels, à un degré d'abstraction assez élevé pour l'ordinateur. Pour faire exécuter ce programme par un ordinateur, une traduction en code interprétable par l'ordinateur est exigée, réalisée par des applications spécifiques, appelées *interprètes* ou *compilateurs*. Durant cette phase, si les règles syntaxiques n'ont pas été respectées, des erreurs peuvent se produire. Dans ce cas, il faut revenir à la phase de traduction de l'algorithme vers le programme correspondant, pour corriger ces erreurs.

Il convient ensuite de procéder à l'exécution proprement dite de ce code interprétable, pour la phase de test. Deux situations peuvent se présenter : soit l'exécution est réussie, sans aucune erreur, auquel cas l'application est validée et peut être fournie au maître d'ouvrage, soit des erreurs apparaissent. Les erreurs peuvent avoir deux sources possibles :

Violeta Felea & Victor Felea

Introduction à l'informatique

Apprendre à concevoir des algorithmes

Destiné aux étudiants des premières années de l'enseignement supérieur en sciences et techniques, cet ouvrage dédié à l'introduction à l'algorithmique est composé d'un cours complet, de nombreux problèmes résolus et d'exercices d'approfondissement.
Il aborde de manière progressive les notions élémentaires de l'analyse des problèmes informatiques et de la conception de solutions algorithmiques. Des commentaires détaillés sur les fausses solutions intuitives avertissent les étudiants sur les erreurs qui peuvent apparaître dans le raisonnement amenant à la conception d'algorithmes.

Sommaire

Docteur en informatique, **Violeta Felea** est Maître de conférences à l'Université de Franche-Comté. Ses travaux concernent les algorithmes et les systèmes parallèles et distribués.

Victor Felea est professeur honoraire en informatique de l'Université Al. I. Cuza de Iasi, en Roumanie, où il enseigne des cours d'algorithmes, de systèmes d'exploitation et de bases de données.

ISBN 978-2-311-01388-7
9 782311 013887

www.VUIBERT.FR