Gilles Dowek

Jean-Pierre Archambault, Emmanuel Baccelli, Claudio Cimelli,
Albert Cohen, Christine Eisenbeis, Thierry Viéville et Benjamin Wack
Préface de Gérard Berry, professeur au Collège de France

Informatique et sciences du numérique

Spécialité ISN en terminale S

Avec des exercices corrigés et idées de projets

 ISBN : 978-2-212-13543-5

EYROLLES

Préface

L'année 2012 voit l'entrée de l'informatique en tant qu'enseignement de spécialité en classe de Terminale scientifique. Cette entrée devenait urgente, car l'informatique est désormais partout. Créée dans les années 1950 grâce à une collaboration entre électroniciens, mathématiciens et logiciens (ces derniers en ayant posé les bases dès 1935), elle n'a cessé d'accélérer son mouvement depuis, envahissant successivement l'industrie, les télécommunications, les transports, le commerce, l'administration, la diffusion des connaissances, les loisirs, et maintenant les sciences, la médecine et l'art, tout cela en créant de nouvelles formes de communication et de relations sociales. Les objets informatiques sont maintenant par milliards et de toutes tailles, allant du giga-ordinateur équipé de centaines de milliers de processeurs aux micro-puces des cartes bancaires ou des prothèses cardiaques et auditives, en passant par les PC, les tablettes et smartphones, les appareils photos, ou encore les ordinateurs qui conduisent et contrôlent les trains, les avions et bientôt les voitures. Tous fonctionnent grâce à la conjonction de puces électroniques et de logiciels, objets immatériels qui décrivent très précisément ce que vont faire ces appareils électroniques. Au XXI^{e} siècle, la maîtrise du traitement de l'information est devenue aussi importante que celle de l'énergie dans les siècles précédents, et l'informatique au sens large est devenue un des plus grands bassins d'emploi à travers le monde. Cela implique que de nombreux lycéens actuels participeront à son essor dans l'avenir.

Ces jeunes lycéens sont bien sûr très familiers avec les appareils informatisés. Mais ce n'est pas pour cela qu'ils en comprennent le fonctionnement, même

sur des plans élémentaires pour certains. Une opinion encore fort répandue est qu'il n'y a pas besoin de comprendre ce fonctionnement, et qu'il suffit d'apprendre l'usage des appareils et logiciels. À l'analyse, cette opinion apparemment naturelle s'avère tout à fait simpliste, avec des conséquences néfastes qu'il faut étudier de près. Pour faire un parallèle avec une autre discipline, on enseigne la physique car elle est indispensable à la compréhension de la nature de façon générale, et aussi de façon plus spécifique au travail de tout ingénieur et de tout scientifique, c'est-à-dire aux débouchés naturels de beaucoup d'élèves de terminale scientifique. Mais qui penserait qu'il suffit de passer le permis de conduire pour comprendre la physique d'un moteur ou la mécanique une voiture ? Or, nous sommes tous autant confrontés à l'informatique qu'à la physique, même si elle ne représente pas un phénomène naturel préexistant ; comme pour la physique, les ingénieurs et scientifiques devront y être au moins autant créateurs que consommateurs. Pour être plus précis, sous peine de ne rester que des consommateurs serviles de ce qui se crée ailleurs, il est indispensable pour notre futur de former au cœur conceptuel et technique de l'informatique tout élève dont le travail technique sera relié à l'utilisation avancée ou à la création de l'informatique du présent ou du futur. Il est donc bien naturel que la nouvelle formation à l'informatique s'inaugure en terminale scientifique. Mais elle devra immanquablement ensuite être élargie à d'autres classes, car tout élève sera concerné en tant que futur citoyen.

Pour être efficace, toute formation scolaire demande un support adéquat. Ce premier livre va jouer ce rôle pour l'informatique, en présentant de façon pédagogique les quatre composantes scientifiques et techniques centrales de son cœur scientifique et technique : langages de programmation, numérisation de l'information, machines et réseaux, et algorithmes. Il a été écrit par des chercheurs et enseignants confirmés, tous profondément intéressés par le fait que les élèves comprennent, assimilent et apprécient les concepts et techniques présentées. Il insiste bien sur deux points essentiels : le fait que ces quatre composantes sont tout à fait génériques, c'est-à-dire valables pour tous les types d'applications, des méga-calculs nécessaires pour étudier l'évolution du climat aux calculs légers et rapides à effectuer dans les micro-puces enfouies partout, et le fait que les concepts associés resteront valables dans le temps. En effet, si les applications de l'informatique évoluent très vite, son cœur conceptuel reste très stable, au moins au niveau approprié pour la terminale scientifique. L'enseigner de façon adéquate est nécessaire autant à la compréhension des bases qu'à tout approfondissement ultérieur. À n'en pas douter, cet ouvrage y contribuera.

Gérard Berry, directeur de recherche Inria
Professeur au Collège de France,
Membre de l'Académie des sciences, de l'Académie des technologies,
et de l'Academia Europaea

Table des matières

Avant-propos

Il y a un siècle, il n'y avait pas d'ordinateurs ; aujourd'hui, il y en a plusieurs milliards. Ces ordinateurs et autres *machines* numériques que sont les réseaux, les téléphones, les télévisions, les baladeurs, les appareils photos, les robots, etc. ont changé la manière dont nous :

- concevons et fabriquons des objets,
- échangeons des informations entre personnes,
- gardons trace de notre passé,
- accédons à la connaissance,
- faisons de la science,
- créons et diffusons des œuvres d'art,
- organisons les entreprises,
- administrons les états,
- etc.

Si les ordinateurs ont tout transformé, c'est parce qu'ils sont polyvalents, ils permettent de traiter des *informations* de manières très diverses. C'est en effet le même objet qui permet d'utiliser des logiciels de conception assistée par ordinateur, des machines à commande numérique, des logiciels de modélisation et de simulation, des encyclopédies, des cours en ligne, des bases de données, des blogs, des forums, des logiciels de courrier électronique et de messagerie instantanée, des logiciels d'échange de fichiers, des logiciels de lecture de vidéos et musique, des tables de mixage numériques, des archives numériques, etc.

En fait, les ordinateurs sont non seulement capables de traiter des informations de manières diverses, mais également de toutes les manières possibles. Ce sont des machines universelles.

En 1936, soit quelques années avant la construction des premiers ordinateurs, **Alan Turing** (1912-1954) – et en même temps que lui Alonzo Church – a étudié les liens entre les notions d'algorithme et de raisonnement mathématique.

Cela l'a mené à imaginer un procédé de calcul, les machines de Turing, et à suggérer que ce procédé de calcul puisse être universel, c'est-à-dire capable d'exécuter tous les algorithmes possibles.

Aller plus loin **De la boîte à outils au couteau suisse**

Cette polyvalence s'illustre aussi par le nombre d'outils que les ordinateurs ont remplacé : machines à écrire, téléphones, machines à calculer, télévisions, appareils photos, électrophones, métiers à tisser…

Traiter des informations

Traiter des informations signifie appliquer, d'une manière systématique, des opérations à des symboles. La recherche d'un mot dans un dictionnaire, le chiffrement et le déchiffrement d'un message secret, l'addition et la multiplication de deux nombres, la fabrication des emplois du temps des élèves d'un lycée ou des pilotes d'une compagnie aérienne, le calcul de l'aire d'une parcelle agricole ou encore le compte des points des levées d'un joueur au Tarot sont des exemples de traitements d'informations.

Aller plus loin **Des algorithmes aussi vieux que l'écriture**

Il y a quatre mille ans, les scribes et les arpenteurs, en Mésopotamie et en Égypte, mettaient déjà en œuvre des algorithmes pour effectuer des opérations comptables et des calculs d'aires de parcelles agricoles. La conception d'algorithmes de traitement de l'information semble remonter aux origines mêmes de l'écriture. Dès l'apparition des premiers signes écrits, les hommes ont imaginé des algorithmes pour les transformer.

Un procédé systématique qui permet de traiter des informations s'appelle un *algorithme*. Ainsi, on peut parler d'algorithmes de recherche d'un mot dans un dictionnaire, d'algorithmes de chiffrement et de déchiffrement, d'algorithmes pour faire des additions et des multiplications, etc. De manière plus générale, un algorithme est un procédé systématique qui permet de faire quelque chose. Par exemple une recette de cuisine est un algorithme.

La notion d'algorithme est très ancienne. Depuis la nuit des temps, les hommes ont conçu et appris des algorithmes, pour fabriquer des objets en céramique, tisser des étoffes, nouer des cordages ou, simplement, préparer des aliments.

Le bouleversement survenu au milieu du XX^{e} siècle tient à ce que les hommes ont cessé d'utiliser exclusivement ces algorithmes à la main ; ils ont commencé à les faire exécuter par des machines, les ordinateurs. Pour y parvenir, il a fallu exprimer ces algorithmes dans des *langages* de programmation, accessibles aux ordinateurs. Ces langages sont différents des langues humaines en ce qu'ils permettent la communication non pas entre êtres humains, mais entre les êtres humains et les machines.

L'informatique est donc née de la rencontre de quatre concepts très anciens :

- machine,
- information,
- algorithme,
- langage.

Ces concepts existaient tous avant la naissance de l'informatique, mais l'informatique les a profondément renouvelés et articulés en une science cohérente.

Structure de l'ouvrage

L'objectif de ce cours est d'introduire les quatre concepts de machine, d'information, d'algorithme et de langage, mais surtout de montrer la manière dont ils fonctionnent ensemble. Quand nous étudierons les algorithmes fondamentaux, nous les exprimerons souvent dans un langage de programmation. Quand nous étudierons l'organisation des machines, nous verrons comment elles permettent d'exécuter des programmes exprimés dans un langage de programmation. Quand nous étudierons la notion d'information, nous verrons des algorithmes de compression, de chiffrement, etc.

Ce livre est donc organisé en quatre parties regroupant vingt-deux chapitres, dont certains d'un niveau plus avancé (indiqués par un astérisque) :

- Dans la **première partie** « **Langages** », nous apprendrons à écrire des programmes. Pour cela, nous allons découvrir les ingrédients dont les programmes sont constitués : l'affectation, la séquence et le test (**chapitre 1**), les boucles (**chapitre 2**), les types (**chapitre 3**), les fonctions (**chapitre 4***) et les fonctions récursives (**chapitre 5***). Pour finir, nous nous pencherons sur la notion de langage formel (**chapitre 6***). Dès que l'on commence à maîtriser ces concepts, il devient possible de créer ses propres programmes.
- Dans la **deuxième partie**, « **Informations** », nous abordons l'une des problématiques centrales de l'informatique : représenter les informations que l'on veut communiquer, stocker et transformer. Nous apprendrons à représenter les nombres entiers et les nombres à virgule (**chapitre 7**), les caractères et les textes (**chapitre 8**), les images et les sons (**chapitre 9**). La notion de valeur booléenne, ou de bit, qui apparaît dans ces trois chapitres, nous mènera naturellement à la notion de fonction booléenne (**chapitre 10**). Nous apprendrons ensuite à structurer de grandes quantités d'informations (**chapitre 11***), à optimiser la place occupée grâce à la compression, corriger les erreurs qui peuvent se produire au moment de la transmission et du stockage de ces informations, et à les protéger par le chiffrement (**chapitre 12***).
- Dans la **troisième partie**, « **Machines** », nous verrons que derrière les informations, il y a toujours des objets matériels : ordinateurs, réseaux, robots, etc. Les premiers ingrédients de ces machines sont des portes booléennes (**chapitre 13**) qui réalisent les fonctions booléennes vues au chapitre 10. Ces portes demandent à être complétées par d'autres circuits, comme les mémoires et les horloges, qui introduisent une dimension temporelle (**chapitre 14**). Nous découvrirons comment fonctionnent les machines que nous utilisons tous les jours (**chapitre 15**). Nous verrons que les réseaux, comme les oignons, s'organisent en couches (**chapitre 16***). Et nous découvrirons enfin les entrailles des robots, que nous apprendrons à commander (**chapitre 17***).
- Dans la **quatrième partie**, « **Algorithmes** », nous apprendrons quelques-uns des savoir-faire les plus utiles au XXI^{e} siècle : ajouter des nombres exprimés en base deux (**chapitre 18**), dessiner (**chapitre 19**), retrouver une information par dichotomie (**chapitre 20***), trier des informations (**chapitre 21***) et parcourir un graphe (**chapitre 22***).

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.

Remarque **Chapitres élémentaires et chapitres avancés***

Les chapitres avancés sont notés ici d'un astérisque. Il s'agit des deux ou trois derniers chapitres de chaque partie. Ils sont signalés en début de chapitre.

Exercices difficiles

Les exercices notés d'un cactus sont d'un niveau plus difficile.

Des encadrés « **Aller plus loin** » donnent des ouvertures vers des questions hors-programme. Chaque chapitre se conclut de trois questions de cours sous forme d'encadré intitulé « **Ai-je bien compris ?** ».

Les propositions de projets sont regroupées en fin de manuel.

Parcours possibles

Cet ouvrage peut être parcouru de plusieurs manières. Nous proposons par exemple de commencer par les chapitres élémentaires de la partie **Informations** (7, 8, 9 et 10), de poursuivre par ceux de la partie **Langages** (1, 2 et 3), de continuer par les chapitres avancés de la partie **Informations** (11 et 12), les chapitres élémentaires de la partie **Algorithmes** (18 et 19) et de la partie **Machines** (13, 14 et 15), et enfin de passer aux chapitres avancés de la partie **Machines** (16 et 17), de la partie **Langages** (4, 5 et 6) et de la partie **Algorithmes** (20, 21 et 22).

Il n'est pas nécessaire de lire ces chapitres au même degré de détails. À chaque élève de choisir les thématiques qu'il souhaite approfondir parmi celles proposées, en particulier par le choix de ses projets.

La seule contrainte est d'acquérir assez tôt les bases des langages de programmation, aux chapitres 1, 2 et 3, pour pouvoir écrire soi-même des programmes. Quand on apprend l'informatique, il est en effet important non seulement d'écouter des cours et de lire des livres, mais aussi de mettre en pratique les connaissances que l'on acquiert en écrivant soi-même des programmes, en se trompant et en corrigeant ses erreurs.

Remerciements

Les auteurs tiennent à remercier Ali Assaf, Olivier Billet, Alain Busser, David Cachera, Vint Cerf, Julien Cervelle, S. Barry Cooper, Ariane Delrocq, Lena Domröse, Raffi Enficiaud, Monique Grandbastien, Guillaume Le Blanc, Fabrice Le Fessant, Philippe Lucaud, Pierre-Étienne Moreau, Claudine Noblet, François Périnet, Gordon Plotkin, François Pottier, Laurent Sartre, Dana Scott, Adi Shamir et Joseph Sifakis pour leur aide au cours de la rédaction de ce livre ainsi que l'équipe des éditions Eyrolles, Anne Bougnoux, Laurène Gibaud, Muriel Shan Sei Fan, Gaël Thomas et Camille Vorng.

Merci également à Christine Paulin, Raphaël Hertzog, Pierre Néron, Grégoire Péan, Jonathan Protzenko et Dominique Quatravaux pour leurs témoignages vivants.

Première partie

Langages

Dans cette première partie, nous apprenons à écrire des programmes. Pour cela, nous découvrons les ingrédients dont les programmes sont constitués : l'affectation, la séquence et le test (chapitre 1), les boucles (chapitre 2), les types (chapitre 3), les fonctions (chapitre 4*) et les fonctions récursives (chapitre 5*). Pour finir, nous nous penchons sur la notion de langage formel (chapitre 6*).

Dès que l'on commence à maîtriser ces concepts, il devient possible de créer ses propres programmes.

```
int getNombreAleatoire()
{
    return 4;   // Nombre choisi au dé (non truqué)
                // Garanti 100% aléatoire.
}
```

XKCD

Les ingrédients des programmes

1

Un ordinateur peut faire bien des choses,
mais il faut d'abord les lui expliquer.

Apprendre la programmation, ce n'est pas seulement savoir écrire un programme, c'est aussi comprendre de quoi il est fait, comment il est fait et ce qu'il fait. Un programme est essentiellement constitué d'expressions et d'instructions.

Nous introduisons dans ce premier chapitre les trois premières instructions fondamentales que sont l'affectation, la séquence et le test. Pour mettre en évidence leur structure, nous indentons les programmes et utilisons les accolades lorsque l'écriture est ambiguë. Nous étudions les instructions en observant les transformations qu'elles opèrent sur l'état de l'exécution du programme, c'est-à-dire sur l'ensemble des boîtes pouvant contenir des valeurs, avec leur nom et leur valeur, le cas échéant.

John Backus (1924-2007) est l'auteur de l'un des premiers langages de programmation : le langage Fortran (1954). Il a par la suite proposé, avec Peter Naur, la *notation de Backus et Naur* qui permet de décrire des grammaires, en particulier celles des langages de programmation (voir le chapitre 6).

Grace Hopper (1906-1992) est, elle aussi, l'auteur d'un des premiers langages de programmation : le langage Cobol (1959). Avant cela, elle a été l'une des premières à programmer le Harvard Mark I de Howard Aiken, l'un des tous premiers calculateurs électroniques.

Deuxième partie

Informations

Dans cette deuxième partie, nous abordons l'une des problématiques centrales de l'informatique : représenter les informations que l'on veut communiquer, stocker et transformer. Nous apprenons à représenter les nombres entiers et les nombres à virgule (chapitre 7), les caractères et les textes (chapitre 8), et les images et les sons (chapitre 9). La notion de valeur booléenne, ou de bit, qui apparaît dans ces trois chapitres, nous mène naturellement à la notion de fonction booléenne (chapitre 10).

Nous apprenons ensuite à structurer de grandes quantités d'informations (chapitre 11*), à optimiser la place occupée grâce à la compression, corriger les erreurs qui peuvent se produire au moment de la transmission et du stockage de ces informations et à les protéger par le chiffrement (chapitre 12*).

XKCD

Représenter des nombres entiers et à virgule

7

Au commencement étaient le 0 et le 1, puis nous créâmes les nombres, les textes, les images et les sons.

Dans ce chapitre, nous voyons comment les nombres sont représentés dans les ordinateurs avec des 0 et des 1.

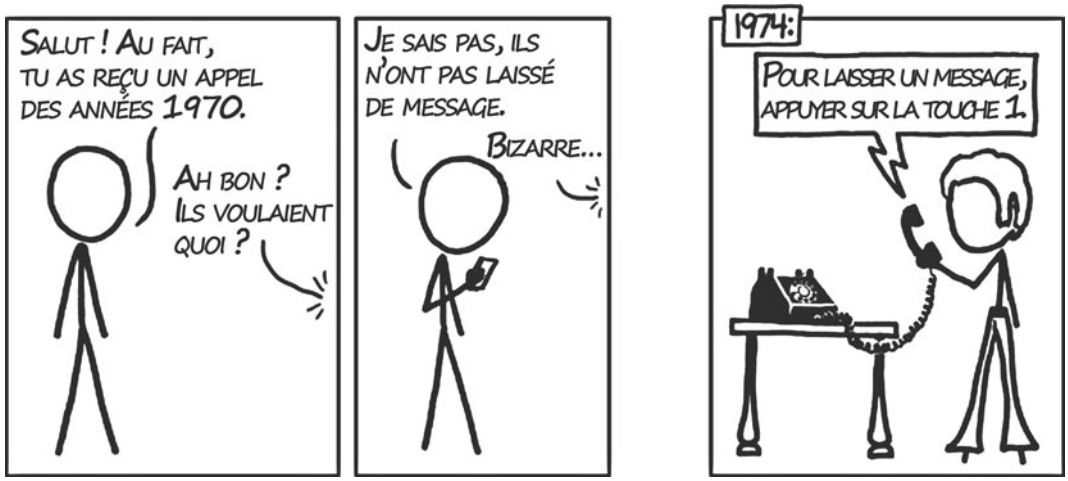
Nous introduisons la notion de base, en partant de la notation décimale que nous utilisons ordinairement pour écrire les nombres entiers. Nous passons par la base cinq puis décrivons la base deux, aussi appelée représentation binaire. Nous généralisons ensuite aux nombres relatifs en utilisant la notation en complément à deux, puis aux nombres à virgule, représentés par leur signe, leur mantisse et leur exposant.

Le livre de l'addition et de la soustraction d'après le calcul indien de **Muhammad al-Khwarizmi** (783 ? - 850 ?), qui présente la numération décimale à position et des algorithmes permettant d'effectuer les opérations sur les nombres exprimés dans ce système, a été le vecteur de la diffusion de ce système de numération dans le bassin méditerranéen. Le mot algorithme est dérivé du nom d'al-Khwarizmi et le mot algèbre (*al-jabr*) du titre d'un autre de ses livres.

Troisième partie

Machines

Dans cette troisième partie, nous voyons que derrière les informations, il y a toujours des objets matériels : ordinateurs, réseaux, robots, etc. Les premiers ingrédients de ces machines sont des portes booléennes (chapitre 13) qui réalisent les fonctions booléennes vues au chapitre 10. Ces portes demandent à être complétées par d'autres circuits, comme les mémoires et les horloges, qui introduisent une dimension temporelle (chapitre 14). Nous découvrons comment fonctionnent ces machines que nous utilisons tous les jours (chapitre 15). Nous verrons que les réseaux, comme les oignons, s'organisent en couches (chapitre 16*). Et nous découvrons enfin les entrailles des robots, que nous apprenons à commander (chapitre 17*).


SALUT ! AU FAIT, TU AS REÇU UN APPEL DES ANNÉES 1970.
AH BON ? ILS VOULAIENT QUOI ?
JE SAIS PAS, ILS N'ONT PAS LAISSÉ DE MESSAGE.
BIZARRE...
1974:
POUR LAISSER UN MESSAGE, APPUYER SUR LA TOUCHE 1.

XKCD

Les portes booléennes

13

Au commencement était le transistor,
puis nous créâmes les portes booléennes
et, à la fin de la journée, les ordinateurs.

Dans ce chapitre, nous voyons de quoi sont faits les ordinateurs à l'échelle microscopique. Nous partons du transistor et construisons successivement des circuits *non* et *ou* qui vont nous permettre ensuite de construire les circuits de toutes les fonctions booléennes, comme nous l'avons vu au chapitre 10.

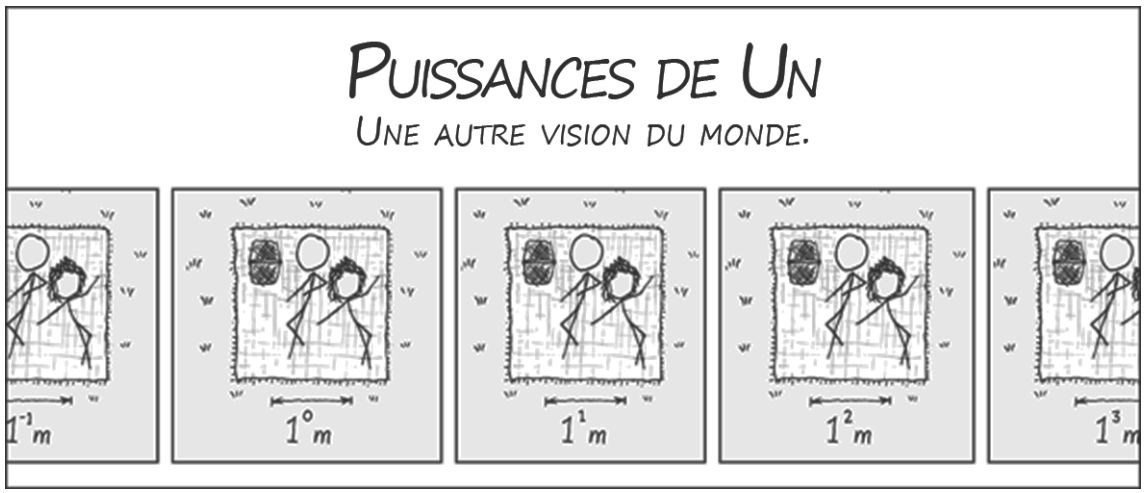
Frances Allen (1932-) est une pionnière de la parallélisation automatique des programmes, c'est-à-dire de la transformation de programmes destinés à être exécutés sur un ordinateur séquentiel – contenant un unique processeur – en des programmes destinés à être utilisés sur un ordinateur parallèle – contenant plusieurs processeurs. Elle est aussi à l'origine de nouvelles méthodes, fondées sur la théorie des graphes, pour optimiser les programmes. Elle a reçu le prix Turing en 2006 pour ces travaux.

Quatrième partie

Algorithmes

Dans cette quatrième partie, nous apprenons quelques-uns des savoir-faire les plus utiles au XXI^{e} siècle : ajouter des nombres exprimés en base deux (chapitre 18), dessiner (chapitre 19), retrouver une information par dichotomie (chapitre 20*), trier des informations (chapitre 21*) et parcourir un graphe (chapitre 22*).


ET LÀ, CE SONT LES GARDIENS DU LABYRINTHE.
L'UN D'EUX MENT TOUJOURS, UN AUTRE DIT TOUJOURS LA VÉRITÉ, ET LE DERNIER FOUDROIE TOUS CEUX QUI POSENT DES QUESTIONS TROP SUBTILES.


PUISSANCES DE UN
UNE AUTRE VISION DU MONDE.
1^{-1} m
1^{0} m
1^{1} m
1^{2} m
1^{3} m

XKCD

Ajouter deux nombres exprimés en base deux

18

Pour faire une addition, l'ordinateur fait comme on lui a appris sur les bancs de l'école.

Dans ce chapitre, nous détaillons l'algorithme de l'addition en base deux, ce qui est surtout un prétexte pour comprendre comment démontrer qu'un algorithme est correct.

L'algorithme est le même que celui que nous utilisons couramment lorsque nous effectuons une addition ordinaire, c'est-à-dire en base dix. Nous démontrons ensuite que l'algorithme que nous avons programmé calcule bien la somme de deux nombres. Pour cela, nous utilisons la notion importante d'invariant de boucle qui est une propriété vraie à chaque tour de boucle. Nous montrons un telle propriété par récurrence sur le numéro du tour de boucle.

Ada Lovelace (1815-1852) est l'auteur du premier algorithme destiné à être exécuté par une machine. Cet algorithme, qui permettait de calculer une suite de nombres de Bernouilli, devait être exécuté sur la machine analytique conçue par Charles Babbage. Malheureusement, Babbage n'a jamais réussi à terminer sa machine. Ada Lovelace est parfois considérée comme le premier programmeur de l'histoire. Le langage de programmation Ada est ainsi nommé en son honneur.

Témoignage **Jonathan, 24 ans**

« *J'ai commencé la programmation avec une (mauvaise) réplique de Mastermind qui permettait de jouer contre l'ordinateur. Après quelques essais manqués, quelques programmes ici et là, je découvre le Web : c'est le coup de foudre – je n'ai jamais décroché depuis. D'abord des sites web en PHP, puis le monde de Mozilla Firefox. De cette rencontre, naît un livre paru chez Eyrolles, rédigé pendant mon temps libre en terminale. Quelques années plus tard, je reprends l'informatique à l'École Normale Supérieure en section informatique : j'y découvre le lambda calcul et la programmation fonctionnelle. Je suis maintenant en thèse à Inria, dans l'équipe qui conçoit le langage OCaml, et je continue d'améliorer le code de Firefox et de Thunderbird sur mon temps libre au sein de la communauté Mozilla.* »

Témoignage **Grégoire, 27 ans**

« Tombé dans la marmite de l'électronique et de la « bidouille », grâce à son papa, Grégoire passe rapidement à l'informatique et décide d'apprendre le C. S'ensuit une vraie passion pour les technologies dites nouvelles, et pour le lien intime et complexe entre matériel et logiciel. Math-sup, math-spé (pour faire plaisir à maman !), puis une formation d'ingénieur, et un rêve américain, concrétisé lors d'un stage dans la Silicon Valley à travailler sur une puce d'encodage vidéo. Grégoire revient ensuite en France et, à Paris, met sa créativité à profit pour inventer la « set-top box » de demain, qui tient au creux de la main et propose une expérience visuelle digne des derniers jeux vidéos. À titre personnel, il songe à des manières de simplifier l'informatique et la programmation, mais sans perdre le lien avec le matériel, et développe secrètement le concept de « Software Atoms ».

Convaincu que la valeur et l'innovation sont portés par ceux qui créent (les programmeurs et les passionnés), il milite pour le développement de hiérarchies d'entreprise moins verticales, pour la revalorisation du développement logiciel en France et de la créativité technologique. »

Idées de projets

Un générateur d'exercices de calcul mental

Programmer un générateur d'exercices de calcul mental : le programme choisit aléatoirement une opération et deux nombres, et vérifie la réponse de l'utilisateur. On peut ensuite poser une série de questions et compter le score total. On pourra enfin prévoir plusieurs niveaux de difficulté selon les opérations proposées ou la taille des nombres à calculer, et laisser l'utilisateur choisir son niveau de difficulté ou attribuer des scores variables aux réponses.

Mastermind

Écrire un programme qui lit deux tableaux de quatre éléments au clavier et indique le nombre d'éléments en commun dans ces deux tableaux. Écrire un programme qui joue au Mastermind : tire au hasard la combinaison secrète et répond aux propositions de l'autre joueur.

Brin d'ARN

Chercher sur le Web ce qu'est un brin d'ARN messager et comment il code pour une protéine. Écrire un programme qui détermine la protéine pour laquelle un brin d'ARN messager code.

Bataille navale

Écrire un programme de bataille navale avec plusieurs bateaux.

Cent mille milliards de poèmes

Chercher sur le Web, ou dans une bibliothèque, ce qu'est le recueil *Cent mille milliards de poèmes* de Raymond Queneau. Écrire un programme qui affiche ces *cent mille milliards de poèmes*.

Site de rencontres

Programmer le moteur d'un site de rencontres, sur le principe suivant :

- Chaque personne inscrite sur le site répond à un questionnaire de personnalité dont les réponses sont des entiers entre 1 et 10.
- On stocke les réponses dans un tableau à double entrée : la case de la ligne `i` et de la colonne `j` contient la réponse de l'inscrit numéro `i` à la question numéro `j`.
- Pour mettre en relation un nouvel inscrit avec une personne déjà inscrite, on parcourt ce tableau en recherchant la ligne qui contient les réponses les plus proches de celles données par le nouvel inscrit. Pour déterminer si des réponses sont proches, on pourra par exemple compter le nombre de réponses identiques ou calculer le total des différences.
- On pourra enfin, à partir d'un tableau déjà rempli, chercher à former autant de couples que possible.

Témoignage **Christine**

« Quand j'étais adolescente, il n'y avait d'ordinateurs ni à l'école, ni à la maison. Pourtant nos cours comportaient de nombreuses activités liées à l'informatique : représenter la même information dans plusieurs formats, élaborer différentes méthodes pour réaliser le même objectif (je me souviens de cartes perforées et d'aiguilles à tricoter pour trier) ou encore raisonner et calculer à partir d'un ensemble donné de règles.

Ma première calculatrice ? Je ne l'ai eue qu'après le baccalauréat et, si elle était programmable, c'était à partir d'un jeu d'instructions très élémentaire. Cela n'en était qu'un défi plus intéressant. Face à une série de problèmes à résoudre, élaborer la bonne séquence d'instructions qui permettra d'obtenir toutes les solutions sans effort est gratifiant, et aussi distrayant que de résoudre un casse-tête. Contrairement à un jeu tout fait, le champ des possibles est immense : les moyens de calcul, de communication, et les données, sont à notre portée pour pouvoir les explorer.

Écrire soi-même un programme qui marche requiert connaissances, expérience, et imagination. La satisfaction est immense d'avoir créé un nouvel objet.

Ce plaisir que j'ai eu à transformer les problèmes en programmes est encore intact après 25 ans de carrière de chercheuse au CNRS puis de professeur à l'université. J'ai choisi l'informatique comme métier (assez tardivement) grâce à un professeur qui m'a fait découvrir tout ce que cette discipline nouvelle offrait d'opportunités en termes de métier et d'activité. J'y ai trouvé une voie qui répondait à mes goûts et compétences. Aujourd'hui, j'essaie de donner aux étudiantes et étudiants les clés théoriques et pratiques du monde numérique qu'ils « consomment » chaque jour, pour qu'ils sachent l'adapter à leurs besoins et contribuent à l'enrichir. »

Tracer la courbe représentative d'une fonction polynôme du second degré

Écrire un programme qui, étant donné une fonction polynôme du second degré, trace sa courbe représentative à l'écran en adaptant automatiquement la fenêtre pour faire apparaître le sommet de la parabole et les éventuels zéros. On pourra faire réaliser les calculs intermédiaires dans des fonctions séparées.

Gérer le score au tennis

Écrire un programme qui gère automatiquement le score au tennis :

1. À quelles conditions un joueur gagne-t-il un jeu ?
2. Définir une fonction qui compte les points au cours d'un jeu. En entrée, on demande répétitivement quel joueur, 1 ou 2, gagne le point ; au fur et à mesure, on calcule et on affiche le score. Le programme s'arrête dès qu'un joueur gagne le jeu, après avoir affiché le nom du vainqueur.
3. Pour faciliter les tests, écrire une seconde version de cette fonction avec pour seule entrée une chaîne de caractères qui contient les numéros successifs des joueurs marquant les points ; la fonction lit cette chaîne caractère par caractère pour compter les points. Ainsi, l'entrée 211222 sera comprise comme : le joueur 2 gagne un point, puis le joueur 1 en gagne deux, puis le joueur 2 en gagne trois et le joueur 2 gagne donc le jeu.
4. Écrire une deuxième fonction qui compte les jeux au cours d'un set et s'arrête lorsqu'un joueur gagne le set. Cette fonction fera appel à la précédente pour savoir qui gagne les jeux. On n'oubliera pas de prévoir le cas particulier du jeu décisif.
5. Écrire une troisième fonction qui compte les sets et s'arrête lorsqu'un joueur gagne le match. On pourra, avant de commencer le match, demander en combien de sets gagnants il est joué.

Automatiser les calculs de chimie

Programmer une boîte à outils pour automatiser les différents calculs que l'on a l'occasion de faire en chimie : durée d'une réaction, pH d'une solution, masses et concentrations, équilibre d'une équation-bilan simple…

Tours de Hanoï

Chercher sur le Web ce que sont les tours de Hanoï et écrire un programme qui trouve une solution à ce jeu.

Tortue Logo

Chercher sur le Web ce qu'est une tortue Logo. Programmez ses principales fonctionnalités. Chercher sur le Web ce qu'est le flocon de Von Koch et dessiner ce flocon à l'aide de cette tortue.

Dessins de plantes

Programmer des dessins de plantes suivant un modèle récursif, par exemple une fougère. On pourra introduire un élément aléatoire dans l'algorithme, afin de varier les résultats obtenus. On pourra utiliser une tortue Logo programmée par soi-même ou par un camarade, ou fournie dans une bibliothèque du langage de programmation utilisé.

Langage CSS

Rechercher sur le Web comment le langage CSS permet de donner une présentation différente à des informations, selon qu'elles sont lues sur un ordinateur ou sur l'écran d'un téléphone. Construire un site Web sur le sujet de son choix qui peut ainsi être consulté sur un écran ou un autre.

Calcul sur des entiers de taille arbitraire

En représentant chaque nombre par un tableau qui contient la suite de ses chiffres, écrire une bibliothèque de fonctions calculant sur des entiers de taille arbitraire, sans les dépassements de capacité qu'engendre l'utilisation du type `int`.

Témoignage **Raphaël, 35 ans**

« L'informatique, je suis tombé dedans très jeune, et mon premier programme en BASIC, qui affichait une carte de France, m'a fait découvrir le plaisir de dominer l'ordinateur. Avec un peu d'attention, on peut lui faire faire ce que l'on veut. Après avoir découvert Windows, je suis devenu un grand fan de Bill Gates. Mais alors que j'étais au lycée, Internet est arrivé et grâce à lui, j'ai redécouvert l'informatique sous un nouvel angle. Je suis entré dans un nouvel univers, celui du logiciel libre. Finis les bricolages : il était désormais possible de diagnostiquer tous les problèmes, voire de les corriger grâce à l'accès au code source. Lorsque cela dépassait mes compétences, je pouvais contacter l'auteur du logiciel et obtenir un correctif en l'espace de quelques jours.

Très rapidement, j'ai ressenti le besoin de m'impliquer dans cette communauté pour apporter ma pierre à l'édifice. C'est ainsi que je suis devenu développeur Debian, distribution GNU/Linux, alors que j'entrais à l'INSA de Lyon. Deux ans après avoir obtenu mon diplôme d'ingénieur en informatique, j'écrivais le premier livre français sur Debian et je lançais ma propre société, Freexian, pour faire de ma passion, mon activité principale. »

Calcul en valeur exacte sur des fractions

Proposer un type représentant une fraction en valeur exacte. Écrire une bibliothèque de fonctions pour effectuer des calculs en valeur exacte sur des fractions. Rechercher sur le Web ce qu'est la méthode de Héron pour calculer une valeur approchée d'une racine carrée et la programmer en utilisant la bibliothèque de calcul sur les fractions.

Représentation des dates et heures

Les systèmes numériques passent automatiquement à l'heure d'été et détectent quand on change de fuseau horaire. Chercher sur le Web des informations sur la représentation des dates et heures : la norme ISO 8601 et le *Network Time Protocol*. Écrire un programme qui donne l'heure dans différents pays.

Transcrire dans l'alphabet latin

Choisir une langue qui utilise un autre alphabet que l'alphabet latin (par exemple le chinois, l'arabe, le japonais, l'hébreu ou le grec) et une trentaine de mots écrits dans cette langue. Associer à chaque syllabe de l'un de ces mots une transcription phonétique écrite dans l'alphabet latin. Écrire un programme d'apprentissage qui tire au hasard des mots dans cette liste, les affiche à l'écran et demande à l'utilisateur de les lire, c'est-à-dire de les transcrire dans l'alphabet latin.

Correcteur orthographique

Réaliser un correcteur orthographique. Un tel programme prend en entrée un fichier texte, le découpe en mots, cherche chaque mot dans un dictionnaire et donne la liste des mots qui ne s'y trouvent pas.

Daltonisme

Rechercher sur le Web ce qu'est le daltonisme et quelles en sont les différentes formes. Écrire un programme qui lit une image dans un fichier au format PPM et l'affiche à l'écran comme la verrait une personne atteinte de chacune des formes de daltonisme.

Système audio par syllabe

Enregistrer un fichier audio par syllabe « ba », « be », « bi », « bo », « bu », « bou », « bon », etc. Utiliser ces fichiers pour écrire un programme qui envoie une séquence de ces grains sonores au système audio, en fonction d'un texte écrit phonétiquement « bon », « jou », « re », « i », « lé », « ne », « ve », « re ». Comparer le résultat avec un système professionnel de synthèse vocale et mettre en lumière les difficultés d'un tel mécanisme.

Déchiffrer automatiquement un message codé selon la méthode de César

Écrire un programme qui déchiffre automatiquement un message codé selon la méthode de César sans connaître a priori la valeur du décalage. Rechercher ce qu'est le chiffre de Vigenère et écrire un programme qui code un message selon ce principe. Rechercher une méthode pour décoder un message codé selon ce principe sans connaître la clé utilisée, et écrire un programme qui effectue ce décodage.

Logisim

Pour construire et simuler des circuits, on peut utiliser logiciel Logisim disponible à l'adresse :

http://ozark.hendrix.edu/~burch/logisim/

Ce logiciel libre fonctionne sur la plupart des ordinateurs. Il n'est pas encore traduit en français, mais il est très bien documenté : on peut commencer sa lecture par le tutoriel. En utilisant ce logiciel, on peut par exemple construire le multiplexeur et les circuits de décalage définis au chapitre 13 et le compteur huit bits défini au chapitre 14.

Banc de registres

Rechercher sur le Web ce qu'est un banc de registres, ainsi que les notions de port de lecture et de port d'écriture sur un banc de registres. À l'aide de l'horloge et des circuits vus aux chapitres 13 et 14, réaliser un banc de 8 registres 8 bits et dessiner le circuit correspondant.

Témoignage **Pierre, 24 ans**

« J'étais en sixième, lorsque mon premier ordinateur est arrivé à la maison, mais je dois reconnaître que pendant bien longtemps, il n'avait d'utilité pour moi que comme simple console de jeux. C'est en classes préparatoires, sur les conseils de mon professeur de mathématiques de l'époque, que j'ai choisi de prendre l'option Informatique plutôt que Sciences Industrielles. Je n'avais aucune idée encore de ce que pouvait être un langage de programmation ni même de la manière dont fonctionnait un ordinateur en général. Se lancer dans une matière totalement inconnue alors qu'on a un emploi du temps déjà chargé n'est pas un choix aisé. Il s'est cependant révélé judicieux : j'étais alors porté sur les mathématiques mais j'ai retrouvé dans l'enseignement de l'informatique certains concepts (ensembles, fonctions…), tout en évitant le calcul (et les erreurs qui vont avec) que je détestais. C'est en école d'ingénieur que mon choix de spécialisation s'est définitivement porté sur l'informatique. Au fil de mes lectures diverses je me faisais (enfin) une idée de ce que je voudrais faire « plus tard » : de la logique, comprendre comment sont construits le raisonnement et les objets des mathématiques. J'ai donc commencé ma quête dans le département de mathématiques où l'on m'a expliqué que ces aspect étaient plutôt étudiés dans le laboratoire d'à côté… Là, j'ai retrouvé un professeur qui m'avait impressionné l'année précédente en présentant les modifications des cases mémoires d'un ordinateur, à l'aide de boites à chaussures. Après un master de recherche en informatique, j'ai commencé ma thèse dans le cadre d'un partenariat entre Inria et la NASA. Je développe un algorithme pour améliorer la précision des calculs sur les ordinateurs. »

Simuler le comportement d'un processeur

Écrire un programme simulant le comportement du processeur lorsqu'il doit exécuter un des programmes écrits en langage machine décrits au chapitre 15. Ce programme lira dans un fichier le contenu de la mémoire, qui contient également le programme à exécuter. Il affichera l'état de la mémoire et des registres au fur et à mesure de l'exécution.

Effectuer des calculs sur les adresses de cases mémoires

Le langage machine décrit au chapitre 15 ne permet d'accéder qu'à un ensemble fini de cases mémoires, dont les adresses sont fournies dans le code des instructions LDA, STA, etc. Pour effectuer des calculs complexes, et notamment pour manipuler des tableaux, on doit pouvoir effectuer des calculs sur les adresses de cases mémoires elles-mêmes. Proposer une extension du langage machine pour effectuer de tels calculs, en définissant la syntaxe des nouvelles instructions, en choisissant leur code machine (binaire) et en expliquant leur fonctionnement. Programmer des boucles simples réalisant des calculs sur des tableaux, par exemple : la somme des éléments d'un tableau, le compteur du nombre d'éléments positifs, etc.

Utilisation du logiciel Wireshark

Installer et lancer le logiciel Wireshark. Capturer des paquets Ethernet ou WiFi depuis la carte réseau et afficher leur contenu à l'écran. Quelles sont les adresses MAC utilisées pour la destination et la source de chaque paquet ? Quels ordinateurs sont identifiés par ces adresses ? Quelle est la taille de chaque paquet ?

Algorithme de pledge

Chercher sur le site web interstices.info ce qu'est l'algorithme de *pledge*. Programmer cet algorithme. Expliquer son utilité et en quoi il se distingue de l'algorithme de sortie d'un labyrinthe du chapitre 22.

Algorithme calculant le successeur d'un nombre entier naturel *n*

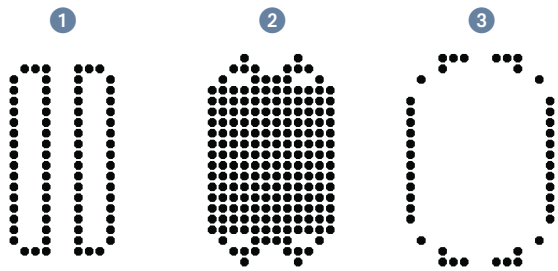
On considère un algorithme qui calcule le successeur d'un nombre entier naturel n, c'est-à-dire le nombre $n + 1$. Cet algorithme est similaire à celui de l'addition, mais il s'applique à un unique nombre : il procède de la droite vers la gauche en posant un chiffre et en propageant une retenue à chaque étape. Identifier les fonctions booléennes qui à un chiffre binaire et une retenue associent le chiffre à poser et la retenue à propager. Programmer cet algorithme et démontrer sa correction en suivant les lignes de la démonstration de correction de l'algorithme de l'addition (voir le chapitre 18). Pour aller plus loin : dessiner un circuit booléen (voir le chapitre 13) qui ajoute 1 à un nombre binaire de quatre bits.

Le jeu de la vie

Sur un damier carré, on dispose des créatures de manière aléatoire. La population évolue d'un état au suivant selon les règles suivantes :

- Une créature survit si elle a 2 ou 3 voisines dans les 8 cases adjacentes et elle meurt à cause de son isolement ou de la surpopulation.
- Une créature naît dans une case vide s'il y a exactement 3 créatures dans les 8 cases voisines, et rien ne se passe dans cette case sinon.

Par exemple, en partant de l'état initial 1, la population évolue à l'état 2, puis à l'état 3, etc.



Écrire un programme qui simule le développement d'une population.

Témoignage **Dominique, 36 ans**

« Je suis artisan de profession ; mon métier consiste à programmer des ordinateurs.

J'avais 9 ans lorsque mon père, professeur de mathématiques, rapporta une calculatrice TI-57 à la maison. Je me souviens encore de mon tout premier programme, expliqué dans le manuel avec des organigrammes dessinés sous la forme de petites voies ferrées que la locomotive-microprocesseur parcourt : [LRN] ("Learn" - Passer en mode programmation), [+], [1], [=], [RST] (revenir au début), [LRN], [RST], [R/S] (Run / Stop). Et c'est parti, la calculatrice compte 1, 2, 3, 4… L'exercice suivant — programmer le jeu de « devine un nombre » — me vit mettre en œuvre toute mon habileté et mon astuce pour tenir dans les 40 pas de programme disponibles dans l'appareil. J'étais pris au jeu ! Et de MO-5 en Atari ST, et puis de Logo en Turbo Pascal, j'ai appris à programmer. Le temps a passé, et matériels et logiciels ont progressé à pas de géant.

Vous qui apprenez l'informatique aujourd'hui, vous vous moquerez peut-être gentiment de la TI-57. Vous auriez tort, car programmer le jeu de « devine un nombre » dans le langage de votre choix est le genre de question qu'on pourrait vous poser lors d'un entretien téléphonique pour une embauche chez Google (où je travaille depuis fin 2007). Les admirables fondements mathématiques de la discipline nous enseignent que les ordinateurs se ressemblent tous ; mais pas les humains qui s'en servent ou qui les programment. Ces derniers se distinguent des premiers parce qu'avec l'ignorance disparaît la peur de l'instrument qui révolutionne nos vies ; et les programmeurs se distinguent entre eux par leurs centres d'intérêt parmi les nombreuses spécialités de l'informatique, leurs langages et styles de programmation préférés, et le but qu'ils poursuivent à titre personnel ou professionnel. Mais tous les programmeurs, ou presque, partageons le goût d'apprendre, l'attention au travail bien fait et l'esprit de géométrie, cher à Blaise Pascal.

Que vous souhaitiez ou non en faire votre métier, je ne saurais trop vous recommander d'aborder la programmation comme j'ai abordé la guitare (et non le violon) : en commençant par vous amuser, puis en continuant par vous perfectionner, comme un forgeron qui cent fois sur son enclume remet son ouvrage. La perfection existe en informatique ; c'est même une joie inépuisable que de se mettre à sa quête, en parcourant un territoire immense et encore si largement inexploré ! »

Une balle

Dessiner une balle qui rebondit sur les parois de la fenêtre graphique. Écrire pour cela une fonction qui dessine une balle sphérique à un endroit donné de l'image, une autre fonction qui calcule si cette balle touche le bord de la fenêtre ou non, une troisième qui calcule sa position suivante selon qu'elle rebondit sur le bord ou non. On pourra prendre en compte la gravité, le ralentissement de la balle à cause des frottements, etc. Produire une suite de quelques centaines d'images et agglomérer cette suite sous la forme d'un film en utilisant un logiciel de création de vidéos ou de GIF animés.

Générateur d'œuvres aléatoires

Trouver des tableaux sur le site web d'un musée. Choisir aléatoirement un petit détail d'un de ces tableaux et agrandir ce détail en un nouveau tableau. Changer les couleurs, le contraste, mélanger des détails issus de deux tableaux différents, etc. Vérifier la licence de ces images et comprendre s'il est possible de publier ses résultats ou non.

Détecteur de mouvement visuel

À l'aide d'une webcam, prendre deux photos successives avec un délai minimal entre les deux, soustraire pixel à pixel ces deux photos et stocker l'image obtenue. Écrire un programme qui utilise un seuil pour détecter dans cette image un mouvement non négligeable et qui donne la taille en pixels de la tache de mouvement obtenue. Tester ce procédé en situation réelle. Quelles en sont les possibilités et les limites ? Appliquer ce procédé à un objet qui « donne un coup de boule » à la caméra, c'est-à-dire qui s'en approche à vitesse constante le long de son axe optique. Avec un modèle géométrique très simple, où l'on considère l'objet comme un rectangle plat parallèle à la caméra, calculer le temps restant avant la collision avec la caméra. Vérifier expérimentalement les résultats obtenus.

Qui est-ce ?

Créer une version numérique et graphique du jeu de société « Qui est-ce ? ». Quelle est l'utilité de la notion de dichotomie pour jouer à ce jeu ?

Un joueur de Tic-tac-toe

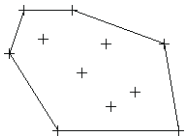
Le Tic-tac-toe est un jeu où deux joueurs placent à tour de rôle l'un des ronds O et l'autre des croix X sur un plateau de trois cases sur trois cases, jusqu'à ce que l'un des joueurs ait aligné trois symboles ou que les neuf cases soient remplies. C'est le joueur O qui commence. Décrire ce jeu sous forme d'un ensemble d'états et d'un ensemble de transitions. Combien y a-t-il d'états possibles ? Un état du jeu peut être gagnant pour O, gagnant pour X, match nul ou en cours de jeu. Exprimer chacun des 3^9 états à l'aide d'un nombre entier. Construire un tableau qui, pour chacun de ces états, indique s'il est gagnant pour l'un des joueurs, nul ou en cours de jeu et, dans ce cas, les transitions possibles pour chacun des joueurs. Calculer ensuite, de proche en proche, les états dans lesquels :

- le joueur O est certain de gagner,
- le joueur X est certain de gagner,
- le joueur O est certain de gagner ou de faire match nul,
- le joueur X est certain de gagner ou de faire match nul.

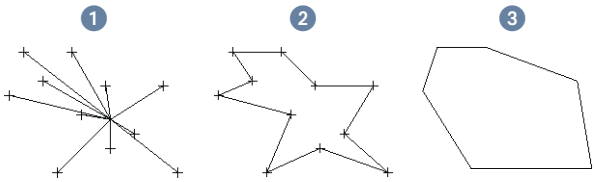
Écrire un programme qui joue contre un joueur humain.

Enveloppe convexe

Écrire un programme qui dessine l'enveloppe convexe d'un ensemble fini de points du plan.



Une manière de faire consiste à trier ces points par angle polaire croissant (1), à relier entre eux les points consécutifs (2), puis à supprimer l'un après l'autre les angles rentrants (3).



Chemins les plus courts

L'algorithme de parcours d'un graphe en largeur d'abord permet de déterminer s'il existe un chemin entre deux sommets d'un graphe et de calculer un plus court chemin, si ces deux sommets sont effectivement reliés. L'algorithme de Roy-Warshall-Floyd va plus loin en déterminant une fois pour toutes s'il existe un chemin entre toutes les paires de sommets d'un graphe et en calculant un plus court chemin pour chaque paire de sommets effectivement reliés.

Plus précisément, cet algorithme calcule deux tableaux :

- un tableau `L` tel que `L[x][y]` soit la longueur du plus court chemin reliant le sommet `x` au sommet `y` si ces deux sommets sont effectivement reliés et ∞ sinon,
- un tableau `R` tel que `R[x][y]` soit le premier sommet après `x` sur un plus court chemin reliant `x` à `y`, si ces deux sommets sont effectivement reliés, et si le plus court chemin les reliant n'est pas de longueur nulle, c'est-à-dire si `x` est différent de `y`.

Comme dans un jeu de pistes, on peut retrouver l'intégralité de ce chemin en partant de `x`, en allant en `R[x,y]`, puis en `R[R[x,y],y]`, etc. jusqu'à arriver en `y`.

Pour calculer ces deux tableaux, on commence par initialiser le tableau `L` en mettant dans la case `L[x][y]` la valeur 0 si `x` est égal à `y`, la valeur 1 si `x` est différent de `y` et s'il y a une arête qui relie `x` à `y`, et la valeur ∞ sinon. On initialise de même le tableau `R` en mettant dans la case `R[x][y]` la valeur `y`.

Ensuite, on imbrique trois boucles :

- pour tous les sommets intermédiaires `z`,
- pour tous les sommets de départ `x`,
- pour tous les sommets d'arrivée `y`,

si `L[x][z] + L[z][y] < L[x][y]`, c'est-à-dire si aller de `x` à `y` en passant par `z` est strictement plus court que le plus court chemin connu, on remplace `L[x][y]` par `L[x][z] + L[z][y]` et `R[x][y]` par `R[x][z]`.

Il n'est pas difficile de montrer que, après avoir effectué les tours de la boucle la plus externe correspondant aux sommets z_1, …, z_k, la case `L[x][y]` du tableau `L` contient la longueur du plus court chemin qui relie `x` à `y` en passant possiblement par les sommets z_1, …, z_k, mais pas par les autres sommets du graphe et que la case `R[x][y]` du tableau `R` contient le premier sommet de ce chemin, si un tel chemin existe et s'il n'est pas de longueur nulle.

Programmer cet algorithme et l'appliquer à un réseau de métro ou de bus.

Utilisation des réseaux sociaux

En utilisant le réseau social le plus répandu dans sa classe, et avec l'accord des personnes concernées, construire le graphe qui a pour sommets les élèves de sa classe et pour arêtes la relation « est l'ami de ». Au fur et à mesure de l'analyse, on pourra raffiner ses données en considérant des sommets collectifs, par exemple la participation à un club. Entrer ces données dans un tableau bidimensionnel dont chaque ligne et colonne correspond à un sommet et chaque case à une arête et calculer les composantes connexes de ce graphe, c'est-à-dire les sous-ensembles maximaux de sommets connectés. Déterminer le nombre de composantes connexes. Ce projet doit être réalisé en respectant l'anonymat des personnes.