

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

Informatique **pour tous** **en classes préparatoires** **aux grandes écoles**

Manuel d'algorithmique
et programmation structurée
avec Python

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

© Groupe Eyrolles, 2013, ISBN : 978-2-212-13700-2

EYROLLES

Table des matières

Table des matières	V
PREMIÈRE PARTIE	
Architecture matérielle et logicielle	1
CHAPITRE 1	
Machine, système d'exploitation et environnement de développement	3
1.1 Qu'est-ce qu'un ordinateur ?	4
1.1.1 Observations externes	4
1.1.2 L'ordinateur, une machine universelle	4
1.1.3 Architecture des ordinateurs	7
1.2 Notion de système d'exploitation	13
1.2.1 Le multitâche	14
1.2.2 Identification des utilisateurs	14
1.2.3 Système de fichiers	16
1.2.4 Contrôle d'accès	19
1.2.5 Lancement d'applications	23
1.2.6 Protections	25
1.3 Environnement de développement intégré	25
1.3.1 Console interactive	27
1.3.2 Éditeur	29
1.3.3 Débogueur	30
CHAPITRE 2	
Représentation des nombres	33
2.1 Représentation des entiers naturels	34
2.1.1 Représentation de l'information par des booléens	34

2.1.2	La numération à position et les bases	34
2.1.3	La base deux	37
2.2	Représentation des entiers relatifs	39
2.2.1	Notation en complément à deux	39
2.2.2	Dépassements de capacité	41
2.3	Représentation des nombres à virgule	45
2.3.1	L'arithmétique flottante	45
2.3.2	Quelques cas particuliers	46
2.3.3	Dépassements de capacité et problèmes de précision	47
2.3.4	Les arrondis	48

DEUXIÈME PARTIE

Algorithmique et programmation **53**

CHAPITRE 3

Expressions : types et opérations..... **55**

3.1	Expressions et types simples	56
3.1.1	Expression	56
3.1.2	Entiers	57
3.1.3	Flottants	60
3.1.4	Booléens	64
3.2	Variables	68
3.2.1	Notion de variable	68
3.2.2	État et valeur d'une expression	68
3.2.3	Déclaration et initialisation	70
3.2.4	Affectation	71
3.3	Types composés	74
3.3.1	Les n -uplets	74
3.3.2	Chaînes de caractères : <i>strings</i>	77
3.3.3	Listes : une première approche	79
3.3.4	Conversions	80

CHAPITRE 4

Instructions : langage minimal de l'algorithmique..... **83**

4.1	Instructions	84
4.1.1	Notion d'algorithme	84
4.1.2	Notion de programme	84
4.1.3	Langage minimal de l'algorithmique	85
4.1.4	Entrées/sorties	85
4.1.5	Séquence d'instructions	86
4.2	Instructions conditionnelles	88
4.2.1	Test simple	88

4.2.2	Indentation signifiante	88
4.2.3	Test avec alternative	90
4.2.4	Tests imbriqués	92
4.3	Boucles conditionnelles	96
4.3.1	Nécessité des boucles	96
4.3.2	Syntaxe d'une boucle conditionnelle	96
4.3.3	Terminaison de boucle	98
4.3.4	Invariant de boucle	101
4.3.5	Boucle infinie	102
4.4	Boucles inconditionnelles	104
4.4.1	Boucle for	104
4.4.2	Valeurs itérables	106
4.4.3	L'itérable range	107
4.4.4	Interrompre une boucle	108
4.4.5	Boucles imbriquées	109
4.5	Exercices	111
CHAPITRE 5		
Fonctions		113
5.1	La notion de fonction	115
5.1.1	Le retour de valeur	115
5.1.2	Variables globales et locales	119
5.1.3	Ordre d'évaluation	121
5.1.4	Passage par valeur	122
5.2	Mécanismes avancés	124
5.2.1	Fonctions locales	124
5.2.2	Fonctions comme valeurs de première classe	124
5.2.3	Fonctions partielles	126
5.2.4	Fonctions de bibliothèque	127
5.2.5	Méthodes	129
5.3	La récursivité	130
5.3.1	Concevoir une fonction récursive	132
5.3.2	Terminaison et correction d'une fonction récursive	135
5.3.3	Complexité d'une fonction récursive	138
5.4	Exercices	139
CHAPITRE 6		
Notions de complexité et algorithmique sur les tableaux		143
6.1	Complexité d'un algorithme	144
6.1.1	Plusieurs algorithmes pour un même problème	144
6.1.2	Complexité et notation O	146
6.1.3	Différentes nuances de complexité	148

6.2 Structure de tableau	150
6.2.1 Construction d'un tableau	150
6.2.2 Accès aux éléments d'un tableau	151
6.2.3 Parcours de tous les éléments d'un tableau	153
6.3 Recherche dans un tableau	155
6.3.1 Recherche séquentielle	155
6.3.2 Recherche dichotomique dans un tableau trié	156
6.4 Recherche d'un mot dans un texte	158
6.5 Matrices	160
6.5.1 Création	162
6.5.2 Copie	163
6.5.3 Dimensions	164
6.5.4 Transposition	164
6.5.5 Produit matriciel	165
6.6 Mode de passage des tableaux	166
6.7 Exercices	168

TROISIÈME PARTIE

Ingénierie numérique et simulation..... 171

CHAPITRE 7

Pivot de Gauss et résolution de systèmes..... 173

7.1 Résolution de $AX = Y$: principe du pivot	175
7.1.1 Le cas des systèmes triangulaires	175
7.1.2 Les transvections	176
7.1.3 Le problème de la comparaison à zéro	177
7.1.4 Formalisation de l'algorithme	179
7.1.5 Le formalisme matriciel	180
7.2 Mise en œuvre	182
7.2.1 Découper le travail	182
7.2.2 Recoller les morceaux	183
7.2.3 Comparaison avec numpy	185
7.3 Complexité	187
7.3.1 Mise sous forme triangulaire	187
7.3.2 Phase de remontée	188
7.3.3 Peut-on faire mieux que n^3 ?	188
7.4 Conditionnement d'une matrice	191
7.4.1 Mesurer les propagations d'erreurs	191
7.4.2 Le conditionnement	191
7.4.3 Quelques exemples caractéristiques	192
7.5 Exercices	194

CHAPITRE 8

Résolution numérique d'équations sur les réels	199
8.1 Méthode dichotomique	200
8.1.1 Principe théorique	200
8.1.2 Terminaison, correction et complexité de l'algorithme	202
8.1.3 Mise en place, essais	203
8.2 Méthode de Newton	204
8.2.1 Extraction de racine	204
8.2.2 Algorithme général, terminaison, correction et complexité	206
8.2.3 Évaluation de la dérivée	207
8.2.4 Mise en œuvre	209
8.3 Quelle méthode choisir ?	210
8.3.1 Bien cerner le contexte	210
8.3.2 Utiliser <code>numpy/scipy</code>	211
8.4 Exercices	212

CHAPITRE 9

Résolution numérique d'équations différentielles	219
9.1 Méthode d'Euler	220
9.1.1 Principe de la méthode d'Euler	220
9.1.2 Quelques notions d'analyse numérique	222
9.1.3 Choix du pas	224
9.2 Mise en œuvre	226
9.2.1 Équations scalaires d'ordre 1	226
9.2.2 Équations scalaires d'ordre 2 ou plus	228
9.3 Utilisation des bibliothèques <code>scipy</code> et <code>matplotlib</code>	230
9.3.1 Intégration des équations différentielles avec <code>odeint</code>	231
9.3.2 Représentation de graphes avec <code>plot</code>	232
9.3.3 De jolis graphes	234
9.3.4 Où on observe quelques limitations	237
9.4 Exercices	239

QUATRIÈME PARTIE

Bases de données	253
-------------------------	------------

CHAPITRE 10

Algèbre relationnelle	255
10.1 Limites des structures de données plates pour la recherche d'informations	256
10.2 Représentation dans le modèle relationnel	257
10.3 Opérateurs sur le modèle relationnel	260
10.3.1 Description des recherches	260
10.3.2 Opérateurs ensemblistes usuels	261

10.3.3 Projection	263
10.3.4 Sélection	264
10.3.5 Renommage	266
10.3.6 Algèbre relationnelle	266
10.4 Utilisation d'un gestionnaire de bases de données relationnelles	268
10.4.1 Description du langage SQL	269
10.4.2 Projection	269
10.4.3 Sélection	269
10.4.4 Opérations ensemblistes	270
10.4.5 Renommage	270
10.5 Base de données et architecture logicielle	271
10.5.1 Architecture client-serveur	271
10.5.2 Architecture trois-tiers	272
10.6 Exercices	273
CHAPITRE I I	
Base de données relationnelle	275
11.1 Clé primaire	276
11.1.1 Clé	276
11.1.2 Clé primaire	276
11.1.3 Lien entre deux tables	277
11.2 Opérateurs complexes de l'algèbre relationnelle	278
11.2.1 Produit cartésien et division cartésienne	279
11.2.2 Jointure	281
11.2.3 Agrégation	285
11.2.4 Composition de requêtes complexes	289
11.3 Traduction en langage SQL	290
11.3.1 Jointure	290
11.3.2 Application simple d'une fonction d'agrégation	291
11.3.3 Agrégation	292
11.4 Exercices	293

CINQUIÈME PARTIE

Algorithmique et programmation avancées	297
--	------------

CHAPITRE I 2

Structure de pile	299
12.1 Opérations caractérisant une structure de pile	300
12.2 Réalisation d'une structure de pile	302
12.2.1 Piles à capacité finie	302
12.2.2 Piles non bornées	304

12.3 Applications	306
12.3.1 Analyse des mots bien parenthésés	306
12.3.2 Évaluation d'une expression arithmétique en notation polonaise inverse	308
12.3.3 Construction d'un labyrinthe parfait	310
12.4 Exercices	314
CHAPITRE 13	
Algorithmes de tri	315
13.1 Tri par insertion	316
13.1.1 Réalisation	316
13.1.2 Complexité	317
13.2 Tri rapide	318
13.2.1 Réalisation	318
13.2.2 Complexité	321
13.3 Tri fusion	322
13.3.1 Réalisation	323
13.3.2 Complexité	325
13.4 Exercices	327
ANNEXE A	
Travaux pratiques	331
A.1 Création de programmes autonomes	331
A.1.1 Compilation d'un programme	331
A.1.2 Exécution autonome d'un programme Python	333
A.2 Mémoire virtuelle et performances de l'ordinateur	334
A.3 Démontage d'un PC de bureau	337
A.3.1 Sécurité	337
A.3.2 Repérage des composants	338
A.3.3 Mise en œuvre	344
A.4 Résolution d'une équation du second degré avec gestion de la comparaison à zéro	352
A.5 Représentation des nombres dans les calculatrices scientifiques	352
A.6 Arithmétique et cryptographie	354
A.6.1 Algorithme d'Euclide	354
A.6.2 Décomposition en facteurs premiers	355
A.6.3 Recherche de grands nombres premiers	356
A.6.4 Application à la cryptographie : la méthode RSA	358
A.7 Manipulation d'images bitmap	359
A.7.1 Traitement pixel par pixel	360
A.7.2 Traitement local	361
A.7.3 Traitement global	362
A.7.4 En couleurs	362

A.8	Prise en main de phpMyAdmin	363
A.8.1	Création d'une table	363
A.8.2	Insertion de valeurs	366
A.9	Clés primaires et clés étrangères	368
A.9.1	Définition d'une clé primaire	368
A.9.2	Clé primaire auto-incrémentée	371
A.9.3	Lien entre deux tables	372
A.9.4	Lancement de requêtes	373
ANNEXE B		
	Compléments sur les entrées/sorties	375
B.1	Lecture et écriture dans des fichiers	375
B.1.1	Lire les lignes d'un fichier	375
B.1.2	Extraction des données dans une ligne	376
B.1.3	Écrire des données dans un fichier	377
B.2	Lecture et écriture dans des images	379
B.2.1	Lecture d'image	379
B.2.2	Traitement	379
B.2.3	Écriture dans une image	380
B.3	Utilisation du module graphique turtle	380
	Références	383

Avant-propos

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

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

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

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

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

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

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

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

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

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

Structure de l'ouvrage

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

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

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

Chaque chapitre contient trois types de contenus :

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

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

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

Avertissement

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

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

Remerciements

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

Première partie

Architecture matérielle et logicielle

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

Deuxième partie

Algorithmique et programmation

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

Troisième partie

Ingénierie numérique et simulation

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

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

Nous montrons sous quelles conditions on peut contrôler ces erreurs, et comment choisir les paramètres des méthodes utilisées pour obtenir un résultat satisfaisant.

Quatrième partie

Bases de données

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

Cinquième partie

Algorithmique et programmation avancées

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

Annexe

Travaux pratiques et compléments

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

Index

- affectation simultanée, 75
- agrégation, 285
- algèbre relationnelle, 266
- algorithme, 84
- alias, 152, 167
- appel d'une fonction, 115
- argument
 - effectif, 115, 123
 - fonction en tant qu', 124
 - formel, 115
- array**, 167
- arrondi
 - erreur d', 49
 - sur les décimaux, 48
- AS, 270
- assert**, 126, 202
- attribut, 258
- base
 - 2, 37
 - 16, 36
 - numération, 34
- base de données
 - architecture, 271
 - clés, 368
 - gestionnaire, 268
 - interface graphique, 363
 - types, 364
- bibliothèque, 127
- binary search*, 156
- bit, 8, 34
- bitmap, 359
- bloc, 87, 115
- booléen, 34
- boucle, 96
 - correction, 101
 - imbrication, 109, 145
 - interrompre, 108
 - invariant, 101
 - terminaison, 98
 - variant, 98
- break**, 108
- Brent
 - méthode de, 211
- bus, 7
- calculatrice, 352
- carte mère, 338
- cas de base, 132
- chaîne de caractères, 77, 152
 - str**, 79
- clé, 276
 - auto-incrémentée, 371
 - étrangère, 277
 - primaire, 276, 368
- close**, 376
- commentaires, 90
- compilateur, 24
- complexité, 146, 187, 203
 - amortie, 149, 226

- au pire, 148
 - dans le meilleur des cas, 149
 - en espace, 149
 - en moyenne, 149
 - en temps, 146
 - optimale, 326
- compréhension
 - tableau par, 151, 231
- conditionnement, 191
- corps d'une fonction, 115
- correction, 202
 - d'une boucle, 101
 - d'une fonction récursive, 135
- débogueur, 30, 103, 118, 137
- décomposition
 - LU , 174, 182, 195
 - QR , 196
 - de Bruhat, 174, 182, 195
 - de Choleski, 196
- définition d'une fonction, 114
- dérivée (évaluation), 207, 212
- dichotomie, 200
- diviser pour régner, 156
- diviseurs d'un entier, 144
- division cartésienne, 279
- domaine, 258
- droits d'accès, 20
- échange
 - fichier d', 335
- elif**, 92
- embarqué
 - système, 7
- enregistrement, 258
- ensemblistes
 - opérateurs, 261
- enumerate**, 154
- entrées/sorties
 - input**, 72
 - print**, 85
- Euclide
 - algorithme d', 354
- Euler
 - méthode d'
 - implicite, 241
 - méthode d', 220
- EXCEPT, 270
- exponentiation rapide, 104, 134
- factorielle, 104
- Fermat
 - méthode de, 356
- fichier
 - lecture, 375
 - système de, 16
 - écriture, 377
- file, 300
- firmware, 12
- flocon, 142
- fonction, 114
 - anonyme, 125, 203, 227
 - appel de, 115
 - arguments d'une, 115
 - comme argument, 124
 - d'agrégation, 286
 - locale, 124
 - mutuellement récursives, 135
 - partielle, 126
 - récursive, 130
 - totale, 126
- for**, 104
- fractions**, 194, 195
- Gauss
 - méthode de Gauss-Seidel, 198
 - pivot de, 173
- Gilbreath, 314
- Givens
 - méthode de, 196
- GROUP BY, 292
- Halley
 - méthode de, 210
- HAVING, 292
- help**, 117
- Heun
 - méthode de, 223, 225, 239, 242
- Horner
 - méthode de, 154
- Householder
 - méthode de, 196
- IEEE 754 (norme), 45

- if, 88
 - elif, 90
 - else, 90
 - imbriqués, 92
- Image, 194, 379
- image
 - conversion, 360
 - couleurs, 362
 - lecture, 379
 - traitement, 360
 - écriture, 380
- import, 127
- in, 107
- indentation, 88, 97, 115
- inode, 19
- input, 72
- instruction, 9, 85
 - boucle conditionnelle, 96
 - boucle inconditionnelle, 104
 - conditionnelle (test), 88
 - de branchement, 9
 - for, 104
 - if, 88
 - séquence, 86
 - while, 96
- interpréteur, 24
- INTERSECT, 270
- invariant, 194, 202
- itérable, 106

- Jacobi
 - méthode de, 197
- JOIN, 290
- join, 129
- jointure symétrique, 281

- Knuth
 - mélange de, 168

- labyrinthe parfait, 310
- Lambda, 125, 203, 226
- Ten, 129
- Tinalg, 186
 - choleski, 196
 - eigvals, 193
 - inv, 194
 - jacobi, 197
 - Tu, 195
 - solve, 186, 188
- liste, 150, 304

- map, 168
- matplotlib.pyplot, 232
 - clf, 234
 - plot, 232
 - savefig, 232
 - show, 232
- matrice, 160
 - copie, 163
 - création, 162
 - de Hilbert, 186, 193, 194
 - de Virginie, 190, 193, 194
 - dimensions, 164
 - produit de, 165
 - transposition, 164
- mélange, 314
 - de Knuth, 168
- mémoire
 - de masse, 12
 - morte, 12
 - virtuelle, 334
 - vive, 7
- métadonnée, 19
- méthode, 129
- microcontrôleur, 7
- module, 127
- modèle relationnel, 257
- mot mémoire, 8, 34
- multitâche, 14

- Newton
 - méthode de, 200, 204
- nombres à virgule flottante
 - dépassement de capacité, 47
 - erreur d'arrondi, 48
 - exposant, 45
 - mantisse, 45, 191
 - test à zéro, 49
- None, 116, 155
- notation polonaise inverse, 308
- NPI, 308
- numpy, 185
 - arange, 231
 - array, 167, 229

- linspace**, 231
 - roots**, 211
 - vectorize**, 232
- n*-uplet, 74
- $O(\dots)$, 146
- objet, 129
 - langage orienté, 129
- octet, 8, 37
- open**, 376
- ordre d'évaluation, 121
- parenthèses
 - langage de, 306
- pas
 - choix du, 224
- pas (choix du), 225
- passage
 - d'arguments, 116
 - par valeur, 122
- pendule
 - amorti, 246
 - non amorti, 236
- périphériques, 8
- permission, 20
- phpMyAdmin, 363
- pile, 299
 - sommet, 300
- pipeline, 10, 198
- pivot, 176
 - de Gauss, 173
 - partiel, 179
- portée, 120
- portrait de phase, 235, 242, 246
- précondition, 126
- print**, 85
- processeur, 8
- produit cartésien, 279
- programme, 84
 - exécuter un, 333
- projection, 263
- pseudo-aléatoire, 95, 119
- RAM, 8
- random**, 95
- range**, 107
- rank**, 179
- readline**, 376
- recherche
 - d'un mot dans un texte, 158
 - dichotomique
 - dans un tableau, 156
 - séquentielle, 155
- récurrence
 - démonstration par, 132, 135
 - forte, 134, 136
- récurtivité, 130
- registre, 9
- regroupement, 287
- relation, 258
 - clé, 276
- relationnel(le)
 - algèbre, 266
 - calcul, 261
 - modèle, 257
 - schéma, 258
- renommage, 266
- représentation des nombres
 - binaire, 37
 - complément à deux, 39
 - entiers longs, 42
 - virgule flottante, 45
- requête, 260
 - croisée, 283
- return**, 116
- reversed**, 154
- ROM, 12
- RSA, 358
- Runge-Kutta
 - méthode de, 223, 225, 239
- schéma relationnel, 258
 - compatible, 262
- scipy.integrate**
 - odeint**, 231
- scipy.optimize**
 - bisect**, 211
 - brentq**, 211
 - fsolve**, 211
 - newton**, 211
- sécante
 - méthode de la, 214
- SELECT, 269
- sélection, 264

- séquence d'instructions, 86
- shell, 15
- sommet d'une pile, 300
- `split`, 130, 376
- SQL, 269, 373
- `str`, 79
- `strip`, 377
- structure de données, 299
- swap, 335
- système d'exploitation, 13
- système embarqué, 7

- table, 258, 363
- tableau, 150, 256
 - copie de, 152
 - par compréhension, 151
 - parcours, 153
- terminaison, 202, 207
 - d'une boucle, 98
 - d'une fonction récursive, 135
- terminal, 15
- tortue, 380
- tours de Hanoï, 141
- tri, 315
 - complexité optimale, 316
 - fusion, 322
 - par insertion, 316
 - rapide, 318
- `tuple`, 74, 81
- Turing
 - machine de, 6

- `turtle`, 380
- type, 56
 - `bool`, 64
 - `float`, 60
 - `int`, 57
 - `list`, 79
 - `str`, 79
 - `tuple`, 74

- UNION, 270
- unité arithmétique et logique, 9

- valeur
 - de première classe, 124
 - immuable, 163
 - itérable, 106
- variable
 - affectation, 71
 - déclaration, 70
 - échanger, 75
 - globale, 119
 - initialisation, 70
 - locale, 119
- variant, 98
- von Neumann
 - architecture de, 7

- WHERE, 269
- `while`, 96

- zéro
 - comparaison à, 174, 178, 195, 352