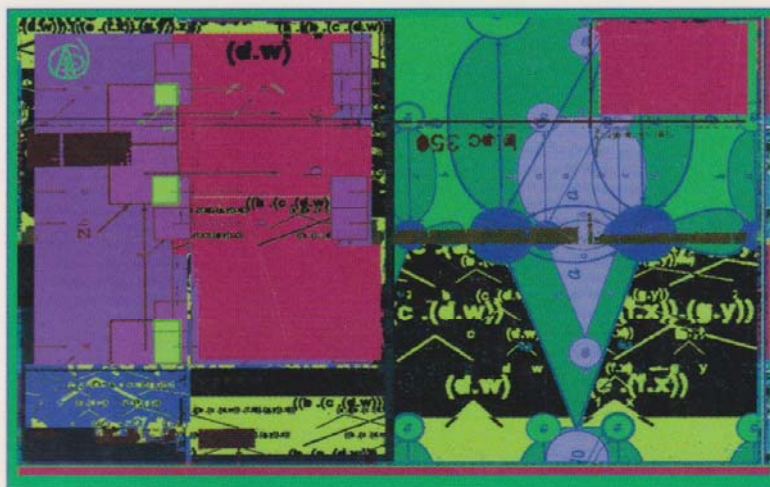


**Pascal Gribomont**

*1<sup>er</sup> et 2<sup>e</sup> CYCLES • ÉCOLES D'INGÉNIEURS*

# **Éléments de programmation en Scheme**

Cours et exemples d'application



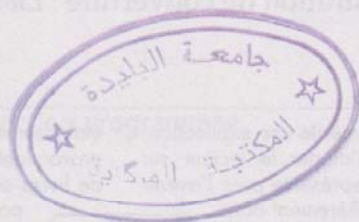
**DUNOD**

005-339-1

2-005-339-1

# Éléments de programmation en Scheme

Cours et exemples d'application



**Pascal Gribomont**  
Professeur à l'université de Liège

DUNOD

# Table des matières

## CHAPITRE 1 • INTRODUCTION

|       |  |    |
|-------|--|----|
| 1.1   | De la fonction mathématique à la fonction programmée | 2  |
| 1.2   | La récursivité                                       | 4  |
| 1.2.1 | Principe et exemples                                 | 4  |
| 1.2.2 | Domaine de validité                                  | 6  |
| 1.2.3 | Terminaison, totalité                                | 7  |
| 1.2.4 | Récurrence simple, récurrence complète               | 8  |
| 1.2.5 | Notation lambda                                      | 10 |
| 1.2.6 | Aspects opérationnels de la récursivité              | 10 |
| 1.3   | Structures de données                                | 12 |
| 1.3.1 | Les entiers naturels                                 | 12 |
| 1.3.2 | Les arbres binaires                                  | 13 |
| 1.3.3 | Les listes   | 15 |
| 1.3.4 | Listes plates, listes profondes                      | 16 |
| 1.4   | Apprendre à programmer avec SCHEME                   | 18 |

## CHAPITRE 2 • LES BASES DE SCHEME

|     |                                       |    |
|-----|---------------------------------------|----|
| 2.1 | Principe de l'interprète              | 21 |
| 2.2 | Les expressions                       | 21 |
| 2.3 | La forme spéciale <code>define</code> | 23 |
| 2.4 | Les symboles et leur double statut    | 24 |

|   |  |    |
|---|--|----|
| 2.5                                       | Les listes   | 25 |
| 2.6                                       | Booléens, prédicats, forme spéciale <code>if</code>      | 28 |
| 2.7                                       | La forme spéciale <code>lambda</code>                    | 29 |
| 2.7.1                                     | Définition   | 30 |
| 2.7.2                                     | Le modèle de substitution                                | 31 |
| 2.7.3                                     | Exemples de calcul par le modèle de substitution         | 34 |
| 2.7.4                                     | Exemples de définitions de procédures                    | 35 |
| 2.7.5                                     | La forme <code>lambda</code> généralisée                 | 37 |
| 2.7.6                                     | Statut « première classe » des procédures                | 38 |
| <b>CHAPITRE 3 • RÈGLES D'ÉVALUATION</b>   |  | 41 |
| 3.1                                       | Résumé des règles  | 41 |
| 3.2                                       | Mode d'application et environnements                     | 44 |
| 3.2.1                                     | Introduction et exemple                                  | 44 |
| 3.2.2                                     | Notion d'environnement                                   | 45 |
| 3.2.3                                     | Les fermetures et le processus d'application             | 45 |
| 3.3                                       | Portée des variables                                     | 48 |
| 3.3.1                                     | Variables globales, variables locales, variables libres  | 48 |
| 3.3.2                                     | Conflits de noms   | 49 |
| 3.3.3                                     | Exemple supplémentaire                                   | 51 |
| 3.3.4                                     | Inférence statique et lexicale des liaisons              | 53 |
| 3.3.5                                     | Deux exercices   | 55 |
| 3.3.6                                     | Occurrences libres, occurrences liées                    | 55 |
| 3.4                                       | Procédures <code>eval</code> et <code>apply</code>       | 57 |
| 3.5                                       | Autres formes conditionnelles                            | 59 |
| 3.5.1                                     | La forme spéciale <code>cond</code>                      | 59 |
| 3.5.2                                     | La fonction <code>not</code>                             | 60 |
| 3.5.3                                     | Les formes spéciales <code>and</code> et <code>or</code> | 60 |
| 3.5.4                                     | Relations entre <code>if</code> et <code>cond</code>     | 61 |
| <b>CHAPITRE 4 • PROCÉDURES RÉCURSIVES</b> |  | 63 |
| 4.1                                       | Préliminaires  | 63 |
| 4.2                                       | Récurtivité et équations                                 | 64 |
| 4.3                                       | Quelques exemples  | 66 |
| 4.3.1                                     | Récurtion sur les nombres                                | 66 |
| 4.3.2                                     | Récurtion sur les listes                                 | 68 |
| 4.3.3                                     | Schémas de récurtion                                     | 70 |
| 4.4                                       | Le double rôle de <code>define</code>                    | 70 |
| 4.5                                       | Le processus de calcul récurtif                          | 71 |
| 4.6                                       | Récurtivité croisée                                      | 74 |

|  |     |
|--|-----|
| <b>CHAPITRE 5 • RÉCURSIVITÉ STRUCTURELLE</b>             | 77  |
| 5.1 Réversivité structurelle sur le domaine des naturels | 77  |
| 5.2 Les listes   | 80  |
| 5.3 Réversivité superficielle sur les listes             | 81  |
| 5.3.1 Le schéma de récursion superficielle               | 81  |
| 5.3.2 Exemples élémentaires                              | 81  |
| 5.3.3 Les listes sans répétition                         | 83  |
| 5.3.4 Le tri par insertion et l'ordre lexicographique    | 84  |
| 5.3.5 Réversivité sur les suites                         | 87  |
| 5.4 Réversivité profonde sur les listes et les arbres    | 90  |
| 5.5 Remarque sur les schémas de programmes               | 93  |
| 5.6 Réversivité structurelle complète et mixte           | 93  |
| 5.7 La séparation fonctionnelle                          | 95  |
| 5.7.1 Application : le double comptage                   | 96  |
| 5.7.2 Application : le produit d'une liste de nombres    | 98  |
| <b>CHAPITRE 6 • CONCEPTION DE PROGRAMME</b>              | 99  |
| 6.1 Première étude                                       | 99  |
| 6.1.1 L'énoncé   | 99  |
| 6.1.2 Analyse, structuration, solution                   | 99  |
| 6.1.3 Variantes  | 100 |
| 6.1.4 Éliminer une fonction auxiliaire?                  | 101 |
| 6.1.5 Généralisation et réutilisation                    | 102 |
| 6.1.6 Filtrage et transformation                         | 105 |
| 6.2 Deuxième étude                                       | 107 |
| 6.2.1 L'énoncé   | 107 |
| 6.2.2 Solution directe, solution par réutilisation       | 107 |
| 6.2.3 L'itérateur  | 108 |
| 6.3 Troisième étude                                      | 109 |
| 6.3.1 Deux énoncés classiques                            | 109 |
| 6.3.2 Solution du premier problème                       | 110 |
| 6.3.3 Solution du second problème                        | 111 |
| 6.3.4 Approche descendante, approche ascendante          | 114 |
| 6.4 Quatrième étude                                      | 114 |
| 6.4.1 Clarifier le problème                              | 115 |
| 6.4.2 Inversion d'une fonction réelle                    | 117 |
| 6.4.3 Solution du problème simplifié                     | 120 |
| 6.4.4 Première cause de divergence                       | 120 |
| 6.4.5 Deuxième cause de divergence                       | 123 |

|  |     |
|--|-----|
| <b>CHAPITRE 7 • ACCUMULATEURS ET PROCESSUS ITÉRATIFS</b> | 125 |
| 7.1 Le principe de l'accumulateur                        | 125 |
| 7.2 Autres exemples numériques                           | 127 |
| 7.3 Accumulateurs et traitement de listes                | 129 |
| 7.4 Conception de programme, cinquième étude             | 131 |
| 7.4.1 L'énoncé   | 131 |
| 7.4.2 Première solution                                  | 131 |
| 7.4.3 Deuxième solution                                  | 133 |
| 7.4.4 Troisième solution                                 | 134 |
| 7.5 Simplification syntaxique d'une récursion            | 135 |
| 7.5.1 La fonction 91                                     | 135 |
| 7.5.2 La fonction d'Ackermann                            | 136 |
| 7.6 Base fonctionnelle de l'accumulateur, style CPS      | 137 |
| 7.6.1 Le produit d'une liste de nombres                  | 139 |
| 7.6.2 Le double comptage                                 | 141 |
| <br>   |     |
| <b>CHAPITRE 8 • EXPRESSIONS SYMBOLIQUES</b>              | 145 |
| 8.1 Arbres binaires                                      | 146 |
| 8.2 Représentation en mémoire                            | 146 |
| 8.3 Notation pointée et notation usuelle                 | 148 |
| 8.4 Représentation des listes                            | 149 |
| 8.5 Récursivité structurelle et expressions symboliques  | 150 |
| 8.6 Égalité, identité                                    | 153 |
| 8.7 Déconstruction des expressions symboliques           | 154 |
| <br>   |     |
| <b>CHAPITRE 9 • ABSTRACTION ET BLOCS</b>                 | 159 |
| 9.1 La forme spéciale <code>let</code>                   | 160 |
| 9.2 Portée   | 163 |
| 9.3 La forme <code>let*</code>                           | 165 |
| 9.4 La forme spéciale <code>letrec</code>                | 167 |
| 9.5 Schémas récursifs avec <code>let</code>              | 170 |
| 9.6 Un exemple de structuration                          | 172 |
| 9.7 Le problème des cavaliers                            | 173 |
| 9.8 Le problème des cruches                              | 181 |

|  |     |
|--|-----|
| <b>CHAPITRE 10 • ABSTRACTION : DONNÉES ET ALGORITHMES</b>  | 185 |
| <b>10.1 Abstraction sur les données</b>                    | 185 |
| 10.1.1 Deux exemples classiques                            | 186 |
| 10.1.2 Arbres binaires complètement étiquetés              | 188 |
| 10.1.3 Arbres, tas et tri                                  | 191 |
| 10.1.4 Le type enregistrement                              | 194 |
| <b>10.2 Les graphes</b>                                    | 195 |
| 10.2.1 Deux modes de représentation                        | 195 |
| 10.2.2 Nœuds successeurs                                   | 197 |
| 10.2.3 Nœuds accessibles                                   | 199 |
| <b>10.3 Abstraction et schémas de récursion</b>            | 201 |
| 10.3.1 Sur quel(s) argument(s) faire porter la récursion ? | 201 |
| <b>10.4 Le problème du sac à dos</b>                       | 203 |
| 10.4.1 Énoncé  | 203 |
| 10.4.2 Stratégie de résolution, données abstraites         | 204 |
| 10.4.3 Développement du programme                          | 204 |
| 10.4.4 Données concrètes et essais                         | 206 |
| <b>10.5 Le problème de la monnaie</b>                      | 208 |
| <b>10.6 Ensembles : type abstrait et application</b>       | 209 |
| 10.6.1 Structures de données ensemblistes                  | 209 |
| 10.6.2 Trois réalisations concrètes du type ensemble       | 210 |
| 10.6.3 Fonctions d'interface                               | 210 |
| 10.6.4 Version abstraite des opérations de base            | 212 |
| 10.6.5 Un opérateur plus général                           | 213 |
| 10.6.6 Solution alternative                                | 214 |
| <b>10.7 Un exercice à propos des automates finis</b>       | 217 |
| 10.7.1 Les automates finis                                 | 218 |
| 10.7.2 Représentation des automates finis                  | 219 |
| 10.7.3 Les automates finis déterministes                   | 219 |
| 10.7.4 Le programme de conversion                          | 222 |
| 10.7.5 Le programme de réduction                           | 223 |
| <b>CHAPITRE 11 • ABSTRACTION PROCÉDURALE</b>               | 225 |
| <b>11.1 Le problème des huit reines</b>                    | 226 |
| 11.1.1 Introduction  | 226 |
| 11.1.2 Idée algorithmique                                  | 226 |
| 11.1.3 Développement du programme                          | 227 |
| <b>11.2 Généralisation</b>                                 | 230 |
| 11.2.1 Première technique                                  | 230 |
| 11.2.2 Deuxième technique                                  | 230 |
| 11.2.3 Troisième technique                                 | 231 |
| 11.2.4 Quatrième technique                                 | 232 |
| <b>11.3 Arbre de recherche</b>                             | 233 |
| 11.3.1 Introduction  | 233 |

|  |  |     |
|--|--|-----|
| 11.3.2   | Programmation                                      | 234 |
| 11.3.3   | Indentation  | 236 |
| 11.4   | La technique du retour arrière                     | 237 |
| 11.4.1   | Un problème de taille                              | 237 |
| 11.4.2   | Fusionner construction et exploitation             | 237 |
| 11.4.3   | Programmation de la méthode de retour arrière      | 238 |
| 11.5   | Le problème du voyageur de commerce                | 240 |
| 11.6   | Programme générique de recherche                   | 243 |
| 11.7   | Processus d'approximations successives             | 244 |
| <b>CHAPITRE 12 • INSTRUCTIONS ALTÉRANTES ET VECTEURS</b> |  | 245 |
| 12.1   | Introduction                                       | 245 |
| 12.2   | L'instruction d'affectation                        | 247 |
| 12.3   | Altération de structures                           | 249 |
| 12.4   | Les vecteurs                                       | 250 |
| 12.4.1   | Pourquoi des vecteurs ?                            | 250 |
| 12.4.2   | Les primitives vectorielles                        | 251 |
| 12.4.3   | Tri par insertion                                  | 252 |
| 12.4.4   | Retournement d'une liste et d'un vecteur           | 254 |
| 12.4.5   | Vecteurs aléatoires                                | 255 |
| 12.5   | Efficacité expérimentale des programmes            | 255 |
| 12.5.1   | Un chronomètre                                     | 256 |
| 12.5.2   | Vérification expérimentale : la suite de Fibonacci | 257 |
| 12.5.3   | Comptage d'instructions                            | 258 |
| 12.5.4   | Ensembles ou multi-ensembles ?                     | 260 |
| 12.6   | Tabulation   | 261 |
| 12.6.1   | Introduction                                       | 261 |
| 12.6.2   | Une solution générale                              | 264 |
| 12.6.3   | Quelques exemples                                  | 264 |
| <b>ANNEXES</b>   |  | 267 |
| A.1  | Vérification formelle des programmes               | 267 |
| A.1.1  | Programmes numériques                              | 267 |
| A.1.2  | Fonctions de listes                                | 270 |
| A.2  | Étude formelle de l'efficacité des programmes      | 272 |
| A.2.1  | Première version                                   | 272 |
| A.2.2  | Deuxième version                                   | 272 |
| A.2.3  | Troisième version                                  | 273 |
| A.3  | Compléments sur le langage SCHEME                  | 274 |
| A.3.1  | Quasi-citation et macros                           | 274 |
| A.3.2  | Autres constructions utiles                        | 275 |
| <b>BIBLIOGRAPHIE</b>                                     |  | 279 |
| <b>INDEX</b>   |  | 281 |