

Table des matières

Introduction	13
Les langages de programmation	13
La culture LISP	15
Guide d'utilisation	18
Remerciements	19
I La Programmation Fonctionnelle	21
1 Les Expressions Préfixées	23
1.1 Les niveaux de langage dans RACKET	23
1.2 La notation préfixée complètement parenthésée	25
1.3 Le toplevel	28
1.3.1 Un endroit où l'on cause	28
1.3.2 Un environnement de variables	29
1.4 Nommer le résultat d'un calcul avec define	30
1.5 Un <i>teachpack</i> pour l'apprentissage	31
1.6 Les nombres	33
1.6.1 Les entiers \mathbb{Z}	34
1.6.2 Les rationnels \mathbb{Q}	35
1.6.3 Les réels \mathbb{R}	37
1.6.4 Les complexes \mathbb{C}	38
1.7 Les booléens et les expressions conditionnelles	39
1.7.1 La conditionnelle if	39
1.7.2 La conditionnelle générale cond	40
1.7.3 Les séquenceurs and et or	41
1.8 Sur l'évaluation d'une expression arithmétique	41
1.9 Exercices	44

2	Les Fonctions	47
2.1	Programmer et tester une fonction	47
2.2	Les fonctions anonymes	50
2.3	Le Garbage Collector	51
2.4	Les fonctions nommées	52
2.5	Les variables globales	52
2.6	Les variables locales	53
2.7	Exercices	54
3	Construire des Images	59
3.1	Images géométriques de base	59
3.1.1	L'accent aigü ou <i>quote</i>	60
3.1.2	Les images textuelles	61
3.1.3	Dimensions d'une image	61
3.2	Les images composées	62
3.2.1	Superposition d'images	62
3.2.2	Alignement horizontal ou vertical d'images	63
3.2.3	Rotation d'une image	63
3.2.4	Changement d'échelle	64
3.2.5	Encadrement d'une image	64
3.2.6	Extraction d'une sous-image	64
3.2.7	Utiliser de véritables images	65
3.3	Exemple : construction d'un smiley	65
3.4	Exercices	66
4	Animer le Monde	69
4.1	Trouver le modèle mathématique du monde	70
4.2	Décrire l'évolution du monde	70
4.3	Transformer le monde en une image	70
4.4	Prévoir la fin du monde	71
4.5	Le squelette d'une animation	71
4.6	Un exemple de monde à un seul paramètre	73
4.7	Interagir avec une animation	75
4.7.1	La gestion des évènements clavier	75
4.7.2	La gestion des évènements souris	76
4.8	Exercices	77
5	Les Structures	79
5.1	Les structures comme n-uplets de données	79
5.2	Exemple : modélisation d'un nombre rationnel	80
5.3	Exemple : modélisation d'une balle physique	81

5.4	Animation d'un monde à plusieurs paramètres	81
5.5	Déstructuration : la forme spéciale <code>match</code>	85
5.6	Exercices	86
6	Programmation par Récurrence	89
6.1	La démonstration par récurrence	89
6.2	La programmation par récurrence	93
6.3	La récurrence enveloppée	96
6.4	Exemples de fonctions récursives enveloppées	98
6.4.1	Un problème combinatoire	98
6.4.2	Construction récursive d'une image	99
6.4.3	Calcul du PGCD	99
6.5	La récurrence terminale ou itération	101
6.6	Exemples de fonctions itératives	103
6.7	Notion de complexité d'un calcul récursif	105
6.8	Exercices	108
7	Compléments sur la récursivité	113
7.1	Liaison statique et liaison dynamique	113
7.1.1	Quelle est la valeur d'une lambda-expression ?	114
7.2	Les fonctions d'ordre supérieur	116
7.2.1	Exemple : modéliser un couple par une fonction	116
7.2.2	La curryfication	117
7.2.3	Passer à l'ordre supérieur pour généraliser	118
7.2.4	Programmation par passage à la continuation : CPS	119
7.3	Exercices	121
8	Les Listes Chaînées	125
8.1	Construction d'une liste	125
8.2	Les symboles et la citation	126
8.3	Accès aux éléments d'une liste	127
8.4	L'égalité en général : <code>equal?</code>	128
8.5	Primitives de base sur les listes	128
8.5.1	La longueur d'une liste : <code>length</code>	128
8.5.2	L'appartenance d'un élément à une liste : <code>member</code>	129
8.5.3	La construction d'une liste en extension : <code>list</code>	129
8.5.4	La construction d'une liste en compréhension : <code>build-list</code>	130
8.5.5	La concaténation de deux listes : <code>append</code>	130
8.5.6	L'inversion d'une liste : <code>reverse</code>	131
8.5.7	La sélection dans une liste : <code>filter</code>	132
8.5.8	Décomposition d'une liste avec <code>match</code>	132

8.6	Recherche dans une liste	133
8.6.1	Accès à l'élément numéro k d'une liste : <code>list-ref</code>	133
8.6.2	Chercher un élément vérifiant une condition donnée	134
8.6.3	Chercher un élément et poursuivre la recherche	135
8.6.4	Chercher le milieu d'une liste	136
8.6.5	Chercher une clé dans une A-liste : <code>assoc</code>	136
8.7	Les ensembles	137
8.7.1	Recherche en profondeur dans une liste	138
8.8	Deux manières de trier une liste	139
8.8.1	Le tri par insertion	139
8.8.2	Le tri par fusion	140
8.9	Animation et listes : un éditeur de polygone	141
8.10	L'ordre supérieur sur les listes	142
8.10.1	L'abstraction du traitement parallèle : <code>map</code>	142
8.10.2	L'abstraction du parcours séquentiel enveloppé : <code>foldr</code>	143
8.10.3	L'abstraction du parcours séquentiel itératif : <code>foldl</code>	144
8.10.4	L'opérateur fonctionnel : <code>apply</code>	145
8.10.5	Les fonctions d'arité variable	147
8.11	Exemple approfondi : analyse par combinateurs	148
8.11.1	Un reconnaiseur	148
8.11.2	Du reconnaiseur à l'analyseur syntaxique	151
8.12	La notion de type abstrait de donnée	151
8.12.1	Un type abstrait <i>vecteur 2D</i>	152
8.12.2	Le fichier contenant le type abstrait doit être un <i>module</i>	154
8.13	Exercices	155
9	Les Arbres	161
9.1	Les arbres binaires d'expressions	161
9.2	Parcours en profondeur et en largeur	162
9.3	Implémentation du type abstrait	164
9.4	Exemples de parcours d'arbres en profondeur	164
9.4.1	Hauteur d'un arbre	164
9.4.2	Recherche d'une feuille	165
9.4.3	Le feuillage	165
9.4.4	Parcours postfixe d'un arbre	165
9.4.5	Valeur d'un arbre	166
9.5	Le parcours préfixe plat et sa réciproque	168
9.5.1	Une solution par le théorème des poids	168
9.5.2	Mieux : une solution en un seul passage	169
9.6	Dessiner un arbre	170
9.7	Piles et parcours d'arbres itératifs	172

9.7.1	Le type abstrait <i>pile fonctionnelle</i>	172
9.7.2	Application au parcours itératif d'un arbre	173
9.7.3	Utilisation du style CPS	174
9.8	Files d'attente et parcours d'arbres en largeur	175
9.8.1	Le type abstrait <i>file d'attente fonctionnelle</i>	175
9.8.2	Application au parcours en largeur	177
9.9	Arbres binaires de recherche	177
9.9.1	Recherche d'un élément	179
9.9.2	Insertion aux feuilles	179
9.9.3	Les arbres équilibrés	180
9.9.4	L'algorithme d'insertion équilibrante	182
9.9.5	Suppression du plus grand élément	184
9.9.6	Suppression d'un élément quelconque	185
9.10	Les ensembles ordonnés	185
9.11	Promenade multi-directionnelle dans un arbre	186
9.12	Exercices	188
10	Programmer avec des Arbres	193
10.1	La machine VRISC et la compilation des arbres	193
10.1.1	Le traducteur vers VRISC1	194
10.1.2	L'interprète de code VRISC1	195
10.1.3	Les nœuds conditionnels	197
10.1.4	La machine virtuelle VRISC2	197
10.1.5	Le traducteur vers VRISC2	198
10.1.6	L'interprète de code VRISC2	199
10.1.7	Exercices	200
10.2	Introduction au calcul formel	201
10.2.1	Le simplificateur	203
10.2.2	Le dérivateur	205
10.2.3	La série de Taylor d'un arbre	205
10.2.4	Exercices	206
10.3	Un démonstrateur automatique de théorèmes	207
10.3.1	Les formules logiques sont des arbres	208
10.3.2	Le type abstrait <i>formule bien formée</i>	209
10.3.3	Exemple : élimination des implications	210
10.3.4	L'algorithme de Wang	210
10.3.5	Exercices	213

11 Le Vrai Langage RACKET	217
11.1 Le niveau de langage est déterminé par le source	217
11.2 Les doublets : une nouvelle vision de la fonction <code>cons</code>	220
11.2.1 Les chaînages de doublets	221
11.2.2 La convention d’affichage du point-parenthèse	222
11.2.3 Les notations contractées <code>c--r</code>	222
11.2.4 L’égalité des doublets	222
11.2.5 La structure de liste	223
11.2.6 La quasiquote	224
11.3 Les booléens	224
11.4 Le déclenchement d’une erreur	225
11.4.1 La véritable syntaxe de la fonction <code>error</code>	225
11.4.2 Les erreurs sont des exceptions	225
11.5 Les variables locales en SCHEME standard	227
11.5.1 <code>let</code> et les liaisons parallèles	227
11.5.2 <code>let*</code> et les liaisons séquentielles	228
11.5.3 <code>letrec</code> et les liaisons récursives	229
11.5.4 Les définitions internes	229
11.6 Les macros, ou extensions syntaxiques	230
11.6.1 Le mécanisme <code>define-syntax</code>	230
11.6.2 La mise au point des macros	232
11.6.3 Les points de suspension	232
11.6.4 Une macro <code>show</code> pour tester nos fonctions	233
11.6.5 La boucle <code>do</code> fonctionnelle	234
11.6.6 Les transformateurs syntaxiques	234
11.7 Exercices	236
II L’Impératif et les Objets	239
12 La Mutation	241
12.1 Le séquençement avec <code>begin</code>	241
12.2 <code>void</code> et les fonctions sans résultat	242
12.3 La mutation des variables ou affectation	242
12.4 Les boucles <code>do</code> , <code>while</code> et <code>for</code>	245
12.5 Retour sur l’appel par valeur	248
12.6 Les générateurs	249
12.7 Les fonctions à mémoire	251
12.8 Les acteurs et les envois de messages	253
12.9 Le graphisme impératif	256
12.9.1 Les primitives graphiques	257

12.9.2	Le graphisme de la tortue	258
12.9.3	Exemples de programmes tortue	261
12.10	La mutation des structures	263
12.10.1	Animation dans un monde mutable	264
12.11	La mutation des vecteurs	265
12.11.1	Échange de deux éléments	266
12.11.2	Exemple : le drapeau hollandais	267
12.11.3	Les matrices	267
12.11.4	Exemple : vectorisation de code VRISC	269
12.12	La mutation des chaînes de caractères	270
12.13	La mutation des doublets	270
12.14	Les tables de hash-code	275
12.15	Exercices	276
13	Le Texte et les Entrées-Sorties	283
13.1	Les caractères	283
13.1.1	Le codage Unicode	283
13.1.2	Le code ASCII et l'encodage UTF-8	285
13.2	Les chaînes de caractères	286
13.2.1	Construction d'une chaîne	286
13.2.2	Accès aux caractères et mutation	287
13.2.3	Pourquoi pas un <code>return</code> comme en JAVA ?	288
13.2.4	Exemple : le codage de César	289
13.2.5	Exemple : un analyseur lexical	290
13.2.6	Les expressions régulières	291
13.3	Écriture sur un port de sortie	295
13.3.1	Écriture à l'écran	295
13.3.2	Écriture dans un fichier sur disque	295
13.3.3	Écriture dans une chaîne	297
13.4	Lecture dans un port d'entrée	297
13.4.1	Lecture au clavier, rôle de <code>eval</code> , et gestion des erreurs	297
13.4.2	Lecture d'un fichier sur disque	299
13.4.3	Lecture dans une chaîne	302
13.5	Entrée-sortie sur Internet : un client Web	302
13.6	L'exploitation du système	305
13.6.1	Les fonctions RACKET liées au système de fichiers	305
13.6.2	Lancement de programmes UNIX à partir de RACKET	306
13.6.3	Exécution de scripts RACKET sous UNIX	308
13.7	Exercices	310

14 La Programmation par Objets et l'API graphique	313
14.1 Rappel : les acteurs à états locaux	313
14.2 Classes et objets en RACKET	314
14.3 Les sous-classes et l'héritage	316
14.4 La construction d'interfaces graphiques	318
14.4.1 Présentation de résultats de calculs	319
14.4.2 Programmation d'un éditeur de texte	322
14.4.3 Dessins dans une fenêtre graphique	326
14.4.4 Déplacement d'objets à la souris	329
14.4.5 Boutons radio, cases à cocher, etc.	333
14.4.6 Animation gourmande et threads	334
14.4.7 Utilisation d'un double buffer	335
14.4.8 Utilisation d'une horloge	336
14.4.9 Simulation d'un mouvement planétaire	336
14.5 Exercices	338
III Syntaxe et Sémantique	341
15 Des Analyseurs Syntaxiques	343
15.1 LEX et la génération d'analyseurs lexicaux	343
15.2 YACC et la génération d'analyseurs syntaxiques	346
15.3 Incursion dans la théorie	348
15.4 Exemple : un traducteur du langage <i>Nano-C</i>	353
15.5 Exercices	358
16 Interprétation d'un sous-ensemble de SCHEME	361
16.1 Le langage MISS	361
16.2 La gestion des environnements	363
16.3 Le cœur de l'interprète : <code>eval/apply</code>	366
16.4 Le traitement des fermetures	367
16.5 La conditionnelle <code>\$if</code>	369
16.6 Les variables locales avec <code>\$let</code>	369
16.7 Les fonctions locales co-récurrentes avec <code>\$letrec</code>	370
16.8 Le séquençement avec <code>\$begin</code>	371
16.9 L'affectation avec <code>\$set!</code>	371
16.10 La définition au toplevel	371
16.11 La bibliothèque initiale	372
16.12 La boucle toplevel	372
16.13 Se protéger contre les erreurs	373
16.14 Exercices	374

17 La prise en main du Contrôle	377
17.1 Rappels sur CPS	377
17.1.1 Du style direct à CPS	377
17.1.2 Abandon et capture de continuation	378
17.1.3 Les continuations à plusieurs variables	381
17.1.4 Mise en attente d'un calcul sans pile	381
17.1.5 Générateurs et calculs pas à pas	382
17.1.6 Application au retour arrière : les N dames	383
17.2 Les continuations de première classe avec <code>call/cc</code>	384
17.2.1 S'échapper d'un calcul récursif	386
17.2.2 Capturer la continuation du toplevel	387
17.2.3 Capturer une continuation pour simuler un <code>GOTO</code>	387
17.2.4 Vers un interprète à continuation	391
17.2.5 Traduction automatique vers CPS	391
17.3 La programmation non déterministe	392
17.3.1 Un chercheur de nombres premiers	393
17.3.2 Un puzzle logique	394
17.3.3 Une implémentation de <code>amb</code>	395
17.4 La Programmation Paresseuse	396
17.4.1 Le langage HASKELL	397
17.4.2 LAZY RACKET	400
17.4.3 Rendre MISS paresseux	404
17.4.4 Une implémentation des flots en SCHEME strict	406
17.5 Exercices	411
18 Annexe : le teachpack valrose	415
Bibliographie	419