

Contents

Preface	xiii
Systematic Program Design	xiv
DrRacket and the Teaching Languages	xvi
Skills that Transfer	xviii
This Book and Its Parts	xix
The Differences	xxiii
Prologue: How to Program	3
Arithmetic and Arithmetic	7
Inputs and Output	12
Many Ways to Compute	18
One Program, Many Definitions	22
One More Definition	26
You Are a Programmer Now	28
Not!	30
I Fixed-Size Data	33
1 Arithmetic	33
1.1 The Arithmetic of Numbers	35
1.2 The Arithmetic of Strings	37
1.3 Mixing It Up	39
1.4 The Arithmetic of Images	40
1.5 The Arithmetic of Booleans	44
1.6 Mixing It Up with Booleans	45
1.7 Predicates: Know Thy Data	47
2 Functions and Programs	49
2.1 Functions	50
2.2 Computing	54
2.3 Composing Functions	58
2.4 Global Constants	62
2.5 Programs	64

3	How to Design Programs	77
3.1	Designing Functions	78
3.2	Finger Exercises: Functions	86
3.3	Domain Knowledge	86
3.4	From Functions to Programs	87
3.5	On Testing	88
3.6	Designing World Programs	90
3.7	Virtual Pet Worlds	100
4	Intervals, Enumerations, and Itemizations	102
4.1	Programming with Conditionals	103
4.2	Computing Conditionally	105
4.3	Enumerations	108
4.4	Intervals	112
4.5	Itemizations	118
4.6	Designing with Itemizations	127
4.7	Finite State Worlds	130
5	Adding Structure	138
5.1	From Positions to <code>posn</code> Structures	139
5.2	Computing with <code>posns</code>	139
5.3	Programming with <code>posn</code>	141
5.4	Defining Structure Types	143
5.5	Computing with Structures	148
5.6	Programming with Structures	152
5.7	The Universe of Data	161
5.8	Designing with Structures	166
5.9	Structure in the World	168
5.10	A Graphical Editor	169
5.11	More Virtual Pets	172
6	Itemizations and Structures	174
6.1	Designing with Itemizations, Again	174
6.2	Mixing Up Worlds	188
6.3	Input Errors	190
6.4	Checking the World	196
6.5	Equality Predicates	198
7	Summary	200

Intermezzo 1: Beginning Student Language	202
II Arbitrarily Large Data	231
8 Lists	231
8.1 Creating Lists	232
8.2 What Is '()', What Is <code>cons</code>	237
8.3 Programming with Lists	239
8.4 Computing with Lists	244
9 Designing with Self-Referential Data Definitions	246
9.1 Finger Exercises: Lists	254
9.2 Non-empty Lists	257
9.3 Natural Numbers	263
9.4 Russian Dolls	268
9.5 Lists and World	272
9.6 A Note on Lists and Sets	278
10 More on Lists	282
10.1 Functions that Produce Lists	283
10.2 Structures in Lists	286
10.3 Lists in Lists, Files	291
10.4 A Graphical Editor, Revisited	301
11 Design by Composition	314
11.1 The <code>list</code> Function	314
11.2 Composing Functions	317
11.3 Auxiliary Functions that Recur	318
11.4 Auxiliary Functions that Generalize	326
12 Projects: Lists	337
12.1 Real-World Data: Dictionaries	337
12.2 Real-World Data: iTunes	339
12.3 Word Games, Composition Illustrated	345
12.4 Word Games, the Heart of the Problem	350
12.5 Feeding Worms	352
12.6 Simple Tetris	356
12.7 Full Space War	359
12.8 Finite State Machines	360

13 Summary	367
Intermezzo 2: Quote, Unquote	369
III Abstraction	381
14 Similarities Everywhere	382
14.1 Similarities in Functions	382
14.2 Different Similarities	384
14.3 Similarities in Data Definitions	388
14.4 Functions Are Values	392
14.5 Computing with Functions	393
15 Designing Abstractions	397
15.1 Abstractions from Examples	397
15.2 Similarities in Signatures	404
15.3 Single Point of Control	409
15.4 Abstractions from Templates	410
16 Using Abstractions	411
16.1 Existing Abstractions	413
16.2 Local Definitions	416
16.3 Local Definitions Add Expressive Power	421
16.4 Computing with <code>local</code>	423
16.5 Using Abstractions, by Example	428
16.6 Designing with Abstractions	433
16.7 Finger Exercises: Abstraction	436
16.8 Projects: Abstraction	437
17 Nameless Functions	439
17.1 Functions from <code>lambda</code>	440
17.2 Computing with <code>lambda</code>	443
17.3 Abstracting with <code>lambda</code>	445
17.4 Specifying with <code>lambda</code>	449
17.5 Representing with <code>lambda</code>	457
18 Summary	462

Intermezzo 3: Scope and Abstraction	464
IV Intertwined Data	487
19 The Poetry of S-expressions	487
19.1 Trees	488
19.2 Forests	497
19.3 S-expressions	499
19.4 Designing with Intertwined Data	506
19.5 Project: BSTs	508
19.6 Simplifying Functions	512
20 Iterative Refinement	514
20.1 Data Analysis	516
20.2 Refining Data Definitions	517
20.3 Refining Functions	520
21 Refining Interpreters	522
21.1 Interpreting Expressions	523
21.2 Interpreting Variables	526
21.3 Interpreting Functions	529
21.4 Interpreting Everything	532
22 Project: The Commerce of XML	533
22.1 XML as S-expressions	534
22.2 Rendering XML Enumerations	541
22.3 Domain-Specific Languages	547
22.4 Reading XML	552
23 Simultaneous Processing	557
23.1 Processing Two Lists Simultaneously: Case 1	558
23.2 Processing Two Lists Simultaneously: Case 2	559
23.3 Processing Two Lists Simultaneously: Case 3	562
23.4 Function Simplification	566
23.5 Designing Functions that Consume Two Complex Inputs	568
23.6 Finger Exercises: Two Inputs	570
23.7 Project: Database	574
24 Summary	588

Intermezzo 4: The Nature of Numbers	589
V Generative Recursion	603
25 Non-standard Recursion	604
25.1 Recursion without Structure	604
25.2 Recursion that Ignores Structure	608
26 Designing Algorithms	614
26.1 Adapting the Design Recipe	615
26.2 Termination	617
26.3 Structural versus Generative Recursion	621
26.4 Making Choices	622
27 Variations on the Theme	627
27.1 Fractals, a First Taste	628
27.2 Binary Search	632
27.3 A Glimpse at Parsing	638
28 Mathematical Examples	643
28.1 Newton's Method	643
28.2 Numeric Integration	647
28.3 Project: Gaussian Elimination	655
29 Algorithms that Backtrack	661
29.1 Traversing Graphs	661
29.2 Project: Backtracking	671
30 Summary	679
Intermezzo 5: The Cost of Computation	680
VI Accumulators	695
31 The Loss of Knowledge	696
31.1 A Problem with Structural Processing	696
31.2 A Problem with Generative Recursion	701

32 Designing Accumulator-Style Functions	705
32.1 Recognizing the Need for an Accumulator	706
32.2 Adding Accumulators	708
32.3 Transforming Functions into Accumulator Style	711
32.4 A Graphical Editor, with Mouse	725
33 More Uses of Accumulation	727
33.1 Accumulators and Trees	727
33.2 Data Representations with Accumulators	734
33.3 Accumulators as Results	740
34 Summary	747
Epilogue: Moving On	751
Computing	751
Program Design	752
Onward, Developers and Computer Scientists	753
Onward, Accountants, Journalists, Surgeons, and Everyone Else .	754
Index	757