
Contents

<i>Preface</i>	<i>page ix</i>
1 Getting started	1
1.1 Values, types, identifiers and declarations	1
1.2 Simple function declarations	2
1.3 Anonymous functions. Function expressions	4
1.4 Recursion	6
1.5 Pairs	11
1.6 Types and type checking	13
1.7 Bindings and environments	14
1.8 Euclid's algorithm	15
1.9 Evaluations with environments	17
1.10 Free-standing programs	19
Summary	19
Exercises	20
2 Values, operators, expressions and functions	21
2.1 Numbers. Truth values. The <code>unit</code> type	21
2.2 Operator precedence and association	23
2.3 Characters and strings	24
2.4 If-then-else expressions	28
2.5 Overloaded functions and operators	29
2.6 Type inference	31
2.7 Functions are first-class citizens	31
2.8 Closures	34
2.9 Declaring prefix and infix operators	35
2.10 Equality and ordering	36
2.11 Function application operators <code> ></code> and <code>< </code>	38
2.12 Summary of the basic types	38
Summary	39
Exercises	39
3 Tuples, records and tagged values	43
3.1 Tuples	43
3.2 Polymorphism	48
3.3 Example: Geometric vectors	48
3.4 Records	50

3.5	Example: Quadratic equations	52
3.6	Locally declared identifiers	54
3.7	Example: Rational numbers. Invariants	56
3.8	Tagged values. Constructors	58
3.9	Enumeration types	62
3.10	Exceptions	63
3.11	Partial functions. The option type	64
	Summary	65
	Exercises	66
4	Lists	67
4.1	The concept of a list	67
4.2	Construction and decomposition of lists	71
4.3	Typical recursions over lists	74
4.4	Polymorphism	78
4.5	The value restrictions on polymorphic expressions	81
4.6	Examples. A model-based approach	82
	Summary	88
	Exercises	89
5	Collections: Lists, maps and sets	93
5.1	Lists	93
5.2	Finite sets	104
5.3	Maps	113
	Summary	119
	Exercises	119
6	Finite trees	121
6.1	Chinese boxes	121
6.2	Symbolic differentiation	127
6.3	Binary trees. Parameterized types	131
6.4	Traversal of binary trees. Search trees	133
6.5	Expression trees	137
6.6	Trees with a variable number of sub-trees. Mutual recursion	138
6.7	Electrical circuits	142
	Summary	144
	Exercises	145
7	Modules	149
7.1	Abstractions	149
7.2	Signature and implementation	150
7.3	Type augmentation. Operators in modules	153
7.4	Type extension	155
7.5	Classes and objects	156
7.6	Parameterized modules. Type variables in signatures	157
7.7	Customizing equality, hashing and the <code>string</code> function	159
7.8	Customizing ordering and indexing	161
7.9	Example: Piecewise linear plane curves	162

Summary	170
Exercises	170
8 Imperative features	175
8.1 Locations	175
8.2 Operators on locations	176
8.3 Default values	179
8.4 Sequential composition	179
8.5 Mutable record fields	180
8.6 References	182
8.7 While loops	183
8.8 Imperative functions on lists and other collections	184
8.9 Imperative tree traversal	185
8.10 Arrays	186
8.11 Imperative set and map	188
8.12 Functions on collections. Enumerator functions	190
8.13 Imperative queue	194
8.14 Restrictions on polymorphic expressions	195
Summary	195
Exercises	196
9 Efficiency	197
9.1 Resource measures	197
9.2 Memory management	198
9.3 Two problems	204
9.4 Solutions using accumulating parameters	206
9.5 Iterative function declarations	209
9.6 Tail recursion obtained using continuations	212
Summary	216
Exercises	216
10 Text processing programs	219
10.1 Keyword index example: Problem statement	219
10.2 Capturing data using regular expressions	221
10.3 Text I/O	229
10.4 File handling. Save and restore values in files	230
10.5 Reserving, using and disposing resources	232
10.6 Culture-dependent information. String orderings	232
10.7 Conversion to textual form. Date and time	235
10.8 Keyword index example: The <code>IndexGen</code> program	238
10.9 Keyword index example: Analysis of a web-source	242
10.10 Keyword index example: Putting it all together	245
Summary	248
Exercises	249
11 Sequences	251
11.1 The sequence concept in F#	251
11.2 Some operations on sequences	254

11.3	Delays, recursion and side-effects	256
11.4	Example: Sieve of Eratosthenes	258
11.5	Limits of sequences: Newton-Raphson approximations	260
11.6	Sequence expressions	262
11.7	Specializations of sequences	266
11.8	Type providers and databases	267
	Summary	277
	Exercises	277
12	Computation expressions	279
12.1	The agenda when defining your own computations	280
12.2	Introducing computation expressions using sequence expressions	281
12.3	The basic functions: <code>For</code> and <code>Yield</code>	282
12.4	The technical setting when defining your own computations	284
12.5	Example: Expression evaluation with error handling	285
12.6	The basic functions: <code>Bind</code> , <code>Return</code> , <code>ReturnFrom</code> and <code>Zero</code>	286
12.7	Controlling the computations: <code>Delay</code> and <code>Start</code>	288
12.8	The basic function: <code>Delay</code>	290
12.9	The fundamental properties of <code>For</code> and <code>Yield</code> , <code>Bind</code> and <code>Return</code>	291
12.10	Monadic parsers	293
	Summary	309
	Exercises	309
13	Asynchronous and parallel computations	311
13.1	Multi-core processors, cache memories and main memory	311
13.2	Processes, threads and tasks	312
13.3	Challenges and pitfalls in concurrency	314
13.4	Asynchronous computations	316
13.5	Reactive programs	321
13.6	Parallel computations	328
	Summary	335
	Exercises	336
Appendix A	Programs from the keyword example	339
A.1	Web source files	339
A.2	The <code>IndexGen</code> program	342
A.3	The <code>NextLevelRefs</code> program	344
Appendix B	The <code>TextProcessing</code> library	346
Appendix C	The dialogue program from Chapter 13	350
	<i>References</i>	353
	<i>Index</i>	355