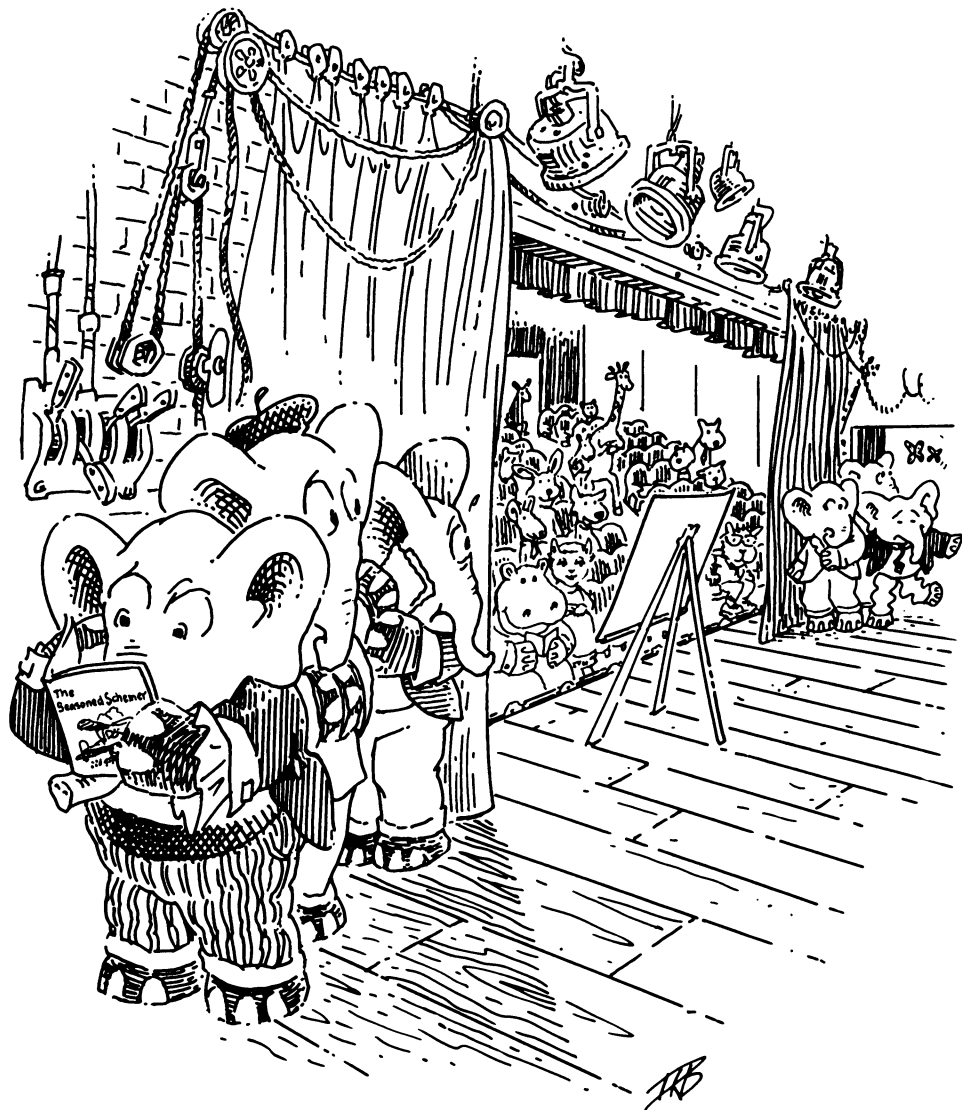


Welcome to the Show



You have reached the end of your introduction to computation. Are you now ready to tackle a major programming problem? Programming requires two kinds of knowledge: understanding the nature of computation, and discovering the lexicon, features, and idiosyncrasies of a particular programming language. The first of these is the more difficult intellectual task. If you understand the material in *The Little Schemer* and this book, you have mastered that challenge. Still, it would be well worth your time to develop a fuller understanding of all the capabilities in Scheme—this requires getting access to a running Scheme system and mastering those idiosyncrasies. If you want to understand the Scheme programming language in greater depth, take a look at the following books:

References

Abelson, Harold and Gerald J. Sussman, with Julie Sussman. *Structure and Interpretation of Computer Programs*, 2nd ed. The MIT Press, Cambridge, Massachusetts, 1996.

Dybvig, R. Kent. *The Scheme Programming Language*, 2nd ed. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1996.

Eisenberg, Michael. *Programming in Scheme*. The Scientific Press, Redwood City, California, 1988.

Ferguson, Ian with Ed Martin and Bert Kaufman. *The Schemer's Guide*, 2nd ed. Schemers Inc., Fort Lauderdale, Florida, 1995.

Harvey, Brian and Matthew Wright. *Simply Scheme: Introducing Computer Science*. The MIT Press, Cambridge, Massachusetts, 1994.

Manis, Vincent S. and James J. Little. *The Schematics of Computation*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1994.

Smith, Jerry D. *An Introduction to Scheme*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1989.

Springer, George and Daniel P. Friedman. *Scheme and the Art of Programming*. The MIT Press, Cambridge, Massachusetts, 1989.

Steele, Guy L., Jr. *Common Lisp: The Language*, 2nd ed. Digital Press, Burlington, Massachusetts, 1990.

Afterword

In Fortran you can speak of numbers, and in C of characters and strings. In Lisp, you can speak of Lisp. Everything Lisp does can be described as a Lisp program, simply and concisely. And where shall you go from here? Suppose you were to tinker with the programs in Chapter 20. Add a feature, change a feature . . . You will have a new language, perhaps still like Lisp or perhaps wildly different. The new language may be described in Lisp, yet it will be not Lisp, but a new creation.

*If you give someone Fortran, he has Fortran.
If you give someone Lisp, he has any language he pleases.*

—Guy L. Steele Jr.

Index

- *application*, 190
- *cond*, 195
- *const*, 192, 194
- *define*, 181
- *identifier*, 183
- *lambda*, 185
- *letcc*, 197
- *quote*, 183
- *set*, 184
- :car*, 191
- ???*, 17

- a-prim*, 191
- abort*, 198
- add-at-end*, 144
- add-at-end-too*, 145
- answer-of*, 200
- arguments-of*, 200
- atom-to-action*, 199

- b-prim*, 191
- bakers-dozen*, 147
- bakers-dozen-again*, 148
- bakers-dozen-too*, 148
- beglis*, 186
- biz*, 125
- body-of*, 200
- bons*, 146
- box*, 181
- box-all*, 186

- call-with-current-continuation & letcc*, 41
- chez-nous*, 103, 104
- cc-body-of*, 200
- consC*, 131, 132, 135
- cond-lines-of*, 200
- counter*, 132, 133, 135

- D*, 124
- deep*, 110, 115, 127, 132, 155
- deep&co*, 161
- deep&coB*, 163
- deepB*, 157, 158
- deepM*, 113, 114, 116, 118, 127–130, 136

- deepR*, 111
- define?*, 180
- depth**, 69, 70, 72–75, 122
- diner*, 94
- dinerR*, 94
- dozen*, 147

- eklist?*, 149
- else?*, 200
- evcon*, 195
- even?*, 188
- evlis*, 190
- expression-to-action*, 199
- extend*, 179

- fill*, 169
- find*, 113, 117
- finite-lenkth*, 153
- food*, 102
- formals-of*, 200
- four-layers*, 157
- function-of*, 200

- get-first*, 174
- get-next*, 171
- global-table*, 181
- glutton*, 102
- gobbler*, 98
- gourmand*, 93
- gourmet*, 92

- id*, 19
- ingredients*, 108
- intersect*, 37, 48
- intersectall*, 38, 39, 41, 49
- is-first-b?*, 6
- is-first?*, 5

- kar*, 146
- kdr*, 146
- kons*, 146, 147

- L*, 121
- last*, 107
- last-kons*, 151

leave, 167
leftmost, 63–66, 76, 78, 81, 82, 167
length, 17, 118–123
lenkth, 143,
letcc & call-with-current-continuation, 41
list-to-action, 199
lm, 78
long, 151
lookup, 179
lookup-in-global-table, 182
lots, 143

max, 75
meaning, 183
member?, 3, 26, 27, 29
mongo, 153
mr, 18
multi-extend, 187
multiremember, 17–19, 22, 25, 26
multiremember-f, 23, 24

name-of, 200
nibbler, 100
Ns, 111

odd?, 188
omnivore, 95, 96

question-of, 200

pick, 13

remember, 52
remember-beyond-first, 54
remember-eq?, 23
remember-f, 23
remember-upto-last, 57
*remember1**, 67, 68, 87, 88, 89, 139, 140
*remember1*C*, 139
*remember1*C2*, 140
rest1, 171
rest2, 172

right-side-of, 200
rm, 84, 85, 88, 89
Rs, 111

same?, 150
scramble, 15, 35, 76
scramble-b, 14
set-counter, 135
set-kdr, 147
setbox, 182
six-layers, 156
start-it, 167
start-it2, 169
sum-of-prefixes, 9, 11, 34
sum-of-prefixes-b, 10
supercounter, 134
sweet-tooth, 107
sweet-toothL, 107
sweet-toothR, 109

text-of, 200
the-empty-table, 179, 199
the-meaning, 182
toppings, 158
two-in-a-row?*, 175, 176
two-in-a-row-b?*, 175
two-in-a-row-b?, 7, 165
two-in-a-row?, 4–7, 33, 34, 165
two-layers, 162

unbox, 182
union, 27, 28, 31, 32

value, 180, 198

waddle, 169
walk, 167

x, 91, 96, 180

Y-bang, 123
Y₁, 123