8.2. Programming Basics

Programming in the Go programming language (Golang) is explained in preceding chapters. For ease of reference, we summarize the programming constructs in four tables: shell commands, packages, statements, and data types.

Table 8.1 Commands in a Linux shell used in this book

Command	Purpose	Example
cat	Print file to standard output	>cat hello.go print hello.go to screen
cd	Change directory	>cd .. change to the parent directory
display	Display a picture	>display ucas.bmp display ucas.bmp on screen
./hello	Execute binary code hello in current directory	
go build	Compile Go source file into executable file	>go build hello.go compile hello.go into executable file hello
go run	Compile and execute Go program	>go run hello.go compile and execute hello.go
ls	List files in current directory	>ls . list files of current directory
Tab key	Automatically complete a command	
↑	Execute the previous command	
Ctr-C	Exit the current command	
Ctr-S	Save the file in editing	

Table 8.2 Golang packages used in this book

Package	Example

fmt	For input and output functions, e.g.,
	fmt.Println(“Hello”) //print “Hello” to display
	fmt.Printf(“One=%d”,1) //print “One=1” to display
	fmt.Scanf(“%d”,&A) //enable user to enter integer to variable A

io/ioutil	For reading/writing files, e.g.,
	p, _ := ioutil.ReadFile("./ucas.bmp") //read data in ucas.bmp to variable p
	ioutil.WriteFile("./mucas.bmp", p, 0666) //write p to mucas.bmp

math	For mathematical functions, e.g., math.Pow(2,3) //returns 2^3=8
os	For interaction with operating system, e.g., V := os.Args[0] //os.Args is an array of command parameters

Table 8.3 Golang statements used in this book

Statement	Example
Assignment	v := 1 // assign 1 to v
	a,b := true,false // assign true to a and false to b
Break	Terminate a loop, e.g.,
	for i:=0;i<5;i++{
	if(i>=3) break
	fmt.Printf("%d ",i)
	}
	The code above will print 0 1 2, because the loop is terminated by break when i==3
Continue	Go to beginning of the next loop iteration, e.g.,
	for i:=0;i<5;i++{
	if(i<3) continue
	fmt.Println("%d ",i)
	}
	The code above will print 3 4, because when i<3, fmt.Println() is skipped
Declaration	v := 3 // declare a variable v and assign 3 to it
	var v int = 3
	const c = 3 // declare a constant c and assign 3 to it
	const c int = 3
For loop	// Compute sum of an integer array
	sum := 0
	var arr [5]int = [5]int{0,1,2,3,4}
	for i:=0; i<len(arr); i++{ // i:=0; is init statement, i<len(arr); is
	sum += arr[i] // condition and i++ is post statement
	}
Function definition	// Define an addition function
	func Add(a int, b int) int { // Add is the function name
	return a+b // a,b are parameters of the function
	}
Function call	c := Add(1,2) // call Add function to obtain c = 3
If statement	if a<b {
	fmt.Println("Smaller") // if a<b, this statement will be executed
	}else{
	fmt.Println("Not smaller") // if a>=b, this statement will be executed
	}
Return	Specify the return value of a function, e.g., "return a+b" in Add

Table 8.4 Go data types used in this book

Data Type	Example
array	var a [10]int // define an array consisting of 10 integers
	var primes [3]int = [3]int {2,3,5} // define an array consisting of 2,3,5
	var p0 int = primes[0] // primes[0] is the zero-th element of primes
bool	var a bool = true // define a bool variable named “a”, whose value is true
	b,c := true,false // define 2 bool variables whose values are true, false
byte	var X byte = ‘a’ // define byte variable X, assign ASCII encoding of ‘a’ to it
int	var y int = 1 // define a signed integer variable whose value is 1
slice	var prime_array [3]int = [6]int {2,3,5} // prime_array is an array
	var s []int = primes[0:2] // s is a slice representing the first 2 elements of primes
	var u []int = make([]int,3) // u is a slice representing a nameless array consisting of 3 integers
string	var str1 string = “directly declaration” // define a string
	var n int = len(str1) // len() returns the number of characters in str1
uint	var i uint = 1 // define an unsigned integer variable whose value is 1

In many Go programs of this book, e.g., fib.dp.bu.go in Example 34 of Section 4.3.1, input data are hardwired into the code, to simplify the code. This is bad programming practice. The fib.dp.bu.go code only works for F(5). It is better to change

```
func main() {
    fmt.Println("F(5)=", fibonacci(5))
}
```

in fib.dp.bu.go to the following code, which enables user to enter a number.

```
func main() {
  var n int = 0
  fmt.Printf("Please enter a natural number between 0 and 92: ")
  _,err := fmt.Scanf("%d", &n)              // n = user-entered integer
  if err != nil {
    fmt.Println("Input Error: Not a number")
    return
  }else if n < 0 {
    fmt.Println("Input Error: Please enter a non-negative integer.")
    return
  }else if n >92 {
    fmt.Println("Input Error: The number is too large. The program overflows.")
    return
  }
  fmt.Printf("F(%d) = %d\n", n, fibonacci(n))
}
```

8.3. Pointers to Supplementary Material

The companion website cs101.ucas.edu.cn provides supplementary material for (1) lecture and projects slides, (2) text, graphics, sound, and video files, and (3) sample programs and tools which help provide a teaching and learning platform.

The website also contains solutions to even-numbered homework exercises, as well as source code of all programs in this book. The following are included.

- Go programs:

  binary.search.go
  fib-10.go
  fib-5.go
  fib-50.go
  fib.binet-50.go
  fib.dp-5.go
  fib.dp-50.go
  fib.dp.big.go
  fib.dp.go
  fib.dp.bu.go
  fib.go
  fib.matrix.go
  fib.Uint.go
  hash.search.go
  hello-1.go
  hello.go
  hide-0.go
  linear.search.go
  name_to_number-0.go
  name_to_number-1.go
  name_to_number.go
  null.go
  parity.go
  pi.go
  pointer.go
  replace.go
  symbols.go
  testPoint123.go
  WebServer.go

- Web programs:

myFirstWebPage.html
staticChildrensDay.html
ChildrensDay.html

Index

A

ABC features, 25
absolute file name, 222
abstract computer, 3, 5
Abstraction, 8, 206
Access Point, 268
accumulator, 80
Acu-Exams., 31
adapt, 146
address, 80, 200
Addressing Mode, 216
algorithm, 18, 147
Algorithmic thinking, 31, 146
ALU, 76
Amdahl’s Law, 249
analog values, 57
antivirus software, 295
application layer, 278
application software, 242
array, 68, 69, 213
ASCII, 63
assembly language, 79
assignment, 69
asymptotic notation, 151
Automata, 120
Automatic execution, 8, 26
automaton, 121
average case, 177
average time complexity, 183
axiomatic systems, 119
axioms, 119

B

Babbage’s problem, 42
bandwidth, 288
barber paradox, 136
base, 80
base register, 80
basic verb, 71
Batch, 45
bees metaphor, 8
Bell’s law, 44
benchmarks, 202
beneficial and harmful impact, 299
best case, 177
biased exponent, 211
big endian, 200
Big endian, 200
binary, 58
Binary, 76
binary search, 179
Binet's formula, 27
BIOS, 242
bit, 18, 27
Bit accuracy, 27
bit-shift, 112
BMP, 222
body, 278
Boolean algebra axioms, 101
Boolean expressions, 101
Boolean function, 102
Boolean Logic, 98
bottleneck, 249
broadcasting, 275
Bubble Sort Algorithm, 149
Bugs, 23
bus, 274
Bush’s problem, 42, 44
business computing., 46
byte, 63, 69, 214
byte-addressable, 58

C

cache, 78, 252
Caesar cipher, 295
camel notation, 12
Caryl Rusbult’s investment model, 29
certificate authority, 298
character, 213
Church-Turing Hypothesis, 132
Church-Turing Thesis, 133
Circuit switching, 276
client, 43, 268
clock cycle, 235, 245
clock frequency, 235
cloud computing, 45
code, 18, 21
COG properties, 206
collaboration, 299
combinational circuit, 234
Combinational Circuits, 229
command-line prompt., 65
comments, 22

Compilation, 21
compile-time errors, 23
Completeness, 137
complexity, 5, 8
Complexity class NP, 191
Complexity class P, 190
Computability, 132
computable problem, 132
computational lens, 47
computational process, 18
Computational Thinking, 25, 47
computer application, 3, 5
Computer Science, 18, 34
Computer simulation, 41
Computer System, 21
computer virus, 292
conclusion, 119
conditional, 227
conjunction, 100
conjunctive normal form, 103
Connectivity, 266, 269
Consistency, 137
constant, 68
constrained, 207
Constructive abstraction, 29
constructs, 73
consumer computing, 46
context, 228
contrapositive, 111
control abstractions, 198, 207, 227
control dependency, 254
converse, 110
Coping with complexity, 199
correctness, 97
Correctness, 8
CPU, 76
creative expression, 5
crossbar, 274
cryptography, 295
current directory, 222
curse of exponentiation, 41
Cyberspace, 18

D

D flip-flop, 235
data, 18, 228
data abstractions, 198
Data Abstractions, 207
data cache, 253
Data compression, 289
data dependency, 254
data link layer, 278
data representation, 200
data types, 67, 207
Decidability, 137
decimal, 58
Decryption, 295
define, 146
denial-of-service, 294
dependencies, 254
dereference, 216
design, 146
dictionary, 178
digit set, 209
digital certificate, 298
digital economy, 36
digital infinity, 8
digital symbol, 18
Digital Symbol Manipulation, 57
digits, 208
directory, 221
discretization, 57
disjunction, 100
disjunctive normal form, 102
distributed denial-of-service, 294
Divide-and-Conquer Algorithms, 153
DNS, 270
documentation, 72
domain, 116
domain name, 272
dot notation, 12
Dynamic networks, 274
dynamic programming, 74, 166
dynamic webpage, 282

E

edges, 269
Effectiveness, 8
Efficiency, 254
Eight Understandings Within, 31
Elementary Web Programming, 282
Embedded, 43
encoding, 19
encryption, 295
enterprise computing, 46
Entscheidungsproblem, 133
Evolutionary networks, 274
exceptions, 248
exclusive-or, 100
Explicit and implicit input/output, 131
exponent, 210
expression, 69

F

Fibonacci Computer, 79, 240
Fibonacci numbers, 26
File Abstraction, 221
file system, 221
files, 18, 221
finite automaton,, 123
Finite states, 130
firewall, 295
firmware, 242
flip-flops, 234
floating-point number, 212
floating-point numbers, 210
FLOPS, 202
for loop, 69, 227
formatting verbs, 67
Free flow, 300
full adder, 106, 231
Full disclosure, 301
full-duplex, 275
function, 21, 73, 228
Function, 227

G

gates, 229
gateway, 269
GDPR, 299
generality, 97
generalizable, 207
Gödel’s Incompleteness Theorems, 136
Gödel's first incompleteness theorem, 137
Gödel's second incompleteness theorem, 138
gold metaphor, 38
Goodstein theorem, 139
graph, 269
graph coloring problem, 192
greatest common divisor, 148
Greedy Strategy, 169
GUI, 45

H

halting problem, 133
Hamiltonian path problem, 193
Harvard architecture, 248
hash function, 183
hash index, 183
Hash search, 183
hash table, 183
hashing, 183
header, 278
heap, 228
hexadecimal, 61
high-level formatting verb, 71
high-level language program, 21
Hockney's formula, 288
home directory, 222
homepage, 273
Host, 268
HTTP, 273
HTTPS, 273, 298
Hub, 268
hypertext, 44

I

I/O devices, 76, 78, 201
implication, 99
Incomputable Problems, 133
index, 69, 80, 208
index register, 80
indirect addressing, 216
inference rules, 119
information and communication technology (ICT), 36
information hiding, 198
information society, 37
information technology, 36
information-hiding, 230
infrastructure software, 242
Input, 146, 147
Insertion Sort Algorithm, 153
Instance, 270
instruction cache, 253
instruction cycle, 245
Instruction driven, 76
instruction pipeline, 239
Instruction Register, 250
instruction set, 79, 239
instructions, 76
integer, 213
Integer Multiplication, 162
Interactive, 45
Internet, 266, 271
interpreter, 23
inverse, 110
IP address, 270
IPv4 addresses, 271
IPv6 addresses, 271
IR, 250

ISP, 268
IT industry, 36
IT professionals, 38

K

Keck’s law, 39
keys, 295
Kleene algebra axioms, 107
Kleene expressions, 107
Kleene function, 109
Kleene logic, 107
Knuth's Characterization of Algorithm, 147
Knuth's Test, 10, 34

L

latency, 288
length, 69
liar paradox, 136
linear search, 179
little endian, 200
local area network, 268
locality, 252
logic, 47
Logic thinking, 31, 97
loop, 68
loop body, 69
loopback domain name, 273
loopback IP address, 273
Lossless compression, 289
Lossy compression, 289
low-level language program, 23

M

machine code, 21
Magic numbers, 72
main memory, 76
maintain the code, 72
make, 214
Manipulation, 18
mantissa, 210
MAR, 250
Matrix Multiplication, 163
MDR, 250
measure, 146
Mechanical Theorem Proving, 120
memory, 76
Memory Address Register, 250
memory cell, 234
Memory Data Register, 250
Merge Sort Algorithm, 155
message storage problem, 199
messages, 276
metadata, 222
Metcalfe’s law, 290
metropolitan area network, 268
middleware, 242
Mobile Internet, 45
modeling, 19
Modularity, 8
modules, 198, 205
Modulization, 229
Moore’s law, 39
Morris worm, 301
Motherboard, 78
multicore, 78
Multimedia, 45
multiplexer, 233

N

namespace, 270
Namespace, 270
naming, 270
negation, 111
negation properties, 116
network, 45, 267
Network effect, 290
Network Laws, 288
network layer, 278
Network Thinking, 266
Network Topology, 274
networking devices, 268
No stop in the middle, 131
nodes, 267
Nordhaus’s law, 39
Normal forms, 102
normalized significant, 211
NP-complete, 193

O

object, 285
objective, 207
offset, 80
operations, 18
order, 116
Output, 146, 147
overflow, 33

P

P versus NP problem, 192
Packet switching, 276
packets, 276
Palindromes, 123
Paradoxes, 133
parallel computer, 255
parallel computing, 255
parity, 112
peak speed, 254
Peano Arithmetic, 137
peering interface, 280
PEPS, 18
persistence, 221
Personal, 45
phishing emails, 294
phishing website, 294
physical layer, 278
PIPA, 299
pixel, 222
P-M-I/O, 76
Pointer, 216
Polynomial Church-Turing Thesis, 133
Portable, 45
positional notation, 208
positional number systems, 208
Postel's Robustness Principle, 245
practical, 198
Precedence, 227
predicate, 115
Predicate refinement, 117
Predicative Logic, 115
premise, 99
prime factorization problem, 297
Princeton architecture, 248
Principle of Harmony, 47
Privacy awareness, 298
private key, 295
Problem, 19
Problem Solving, 18
process, 203
processor, 76
professionally sharing of scientific data, 300
program, 18
program counter, 80
program cycle, 245
program status register, 80
Programming, 19
proposition, 99
Propositional Logic, 99
protocol, 276
protocol stack, 266, 276
public key, 295
public-key encryption, 295
Public-key encryption, 295
Pumping Lemma, 144

Q

quantifiers, 115
quicksort, 175
quotation marks, 12

R

Randomization Strategy, 175
reads, 221
real computer, 3, 5
reducible, 133
redundancy, 281
Reed's law, 290
relative file names, 222
Remark, 270
representation, 58
Responsible Computing, 288, 293
responsible disclosure, 301
RFCs, 270
right-moving Turing machine, 144
ripple-carry adder, 231
root directory, 222
rounding error, 30
routers, 268
RSA method, 296
runtime errors, 23
Russell's paradox, 136

S

scientific computing, 45
seamless execution, 198, 205
seamless transition, 243
Seamless transition, 8
search, 178
Search Algorithms, 178
search engine, 275
Selection, 227
Sequencing, 227
sequential circuit, 234
sequential circuits, 236
sequential computers, 255
Serial adder, 236

Serial execution, 76
server, 268
service interface, 280
shell, 23, 66
significant, 210
simple signed integer, 62
Simulation, 41
Single Factor Optimization, 158
slash notation, 12
slice, 73, 75, 213
software stack, 242
Sorting Problem, 149
spams, 294
Spatial locality, 202
Specific Algorithm, 146
speedup, 249
spyware, 295
stable matching problem, 170
stack, 203, 228
standard, 270
Standard Error, 66
Standard Input, 66
Standard Output, 66
state circuit, 234
state transition, 79
statement, 21
state-transition diagram, 121
state-transition table, 122
Static networks, 274
Steps, 146, 147
Stored program, 76
stored program architecture, 76
stores, 221
Strassen's algorithm, 164
string, 68, 69
subtractor, 236
supercomputers, 202
sustained speed, 254
Switch, 268
Symbol-Manipulation System, 76
symmetric-key encryption, 295
synchronous sequential circuits, 239
System, 20
system heterogeneity, 205
system organization, 205
system scale, 205
system software, 242
system variation, 205
systematic, 199
Systems thinking, 31, 198

T

TCP/IP protocol stack, 278
Temporal locality, 202
text, 228
thorough, 199
Three Features Without, 25
Time complexity, 152
TLS, 298
topology, 274
transport layer, 278
Trojan horses, 295
Truth table, 100
Turing computable, 132
Turing machine, 126
Turing Machines, 120
Turing Test, 46
Turing's problem, 45
two's complement, 62
type cast, 70

U

uint8, 69, 214
UKA units, 34
undesirable, 295
Unicode, 207
Universality, 8
unsigned integer, 62
URL, 273
usage mode, 44
User Experience, 288
UTF-8, 213

V

variable, 68
vertex, 269
Viral marketing, 292
virtual things, 41
virtual worlds, 41
von Neumann architecture, 76
von Neumann bottleneck, 252
von Neumann computer, 79
von Neumann's Exhaustiveness Principle, 247
VPNs, 295

W

web address, 273

Web over Internet Stack, 276
web resource, 273
wide area network, 268
WiFi router, 302
Wonder of Cyberspace, 41
Wonder of exponentiation, 39
Wonder of Simulation, 41
Word, 58
word length, 34, 58
World Wide Web, 273
worst case, 177

Y

Yang's Cycle Principle, 244