
Pemrograman Go

Dr. Bambang Purnomosidi D. P.



Daftar Isi

1	Tentang Buku Ini	3
2	Pengenalan Go	4
	Apa itu Go?	4
	Lisensi Go	4
	Instalasi Go	4
	Distribusi Go	4
	Download dan Install Go	5
	Konfigurasi Variabel Lingkungan Sistem Operasi, Compiler Go, dan Workspace	6
	Menguji Instalasi Go	12
	Memahami Lingkungan Peranti Pengembangan Go	13
	go	13
	godoc	14
	gofmt	16
3	IDE Untuk Go	18
	Menggunakan Vim	18
	Instalasi dan Konfigurasi Pathogen	18
	Instalasi dan Konfigurasi Plugin Golang dan Plugin Pendukung	19
	Autocompletion	20
	Menggunakan LiteIDE	22
	Software IDE Lain	23
4	Dasar-dasar Pemrograman Go	24
	Struktur Program Go	24
	Program Aplikasi Sederhana - 1 File binary executable Utama	24
	Tanpa Proses Kompilasi	25
	Mengkompilasi Menjadi <i>Binary Executable</i>	25
	Pustaka / Library / Package	26
	Mengatur Workspace	26
	Membuat Pustaka	27

Membuat Aplikasi yang Memanfaatkan Pustaka	27
Konstruksi Dasar Bahasa Pemrograman Go	28
Komentar	28
Tipe Data Angka / Numerik	28
String	30
Boolean	31
Variabel	31
Konstanta	34
Pointer	34
Struktur Kendali	34
Perulangan dengan for	34
Seleksi Kondisi	34
Defer	34
5 Fungsi / Function	35
6 Penanganan Kesalahan	36
Penggunaan Kode Error	36
Panic dan Recover	36
7 Struktur Data Lanjut	37
Arrays	37
Slices	37
Maps	37
Struct	37
8 Method	38
9 Testing	39
10 Konkurensi	40

1 Tentang Buku Ini



Buku ini berisi materi pemrograman Go dengan lisensi Creative Commons Attribution-ShareAlike 4.0 International License - CC-BY-SA 4.0. Secara umum, penggunaan lisensi ini mempunyai implikasi bahwa pengguna materi:

1. Harus memberikan atribusi ke penulis (* Dr. Bambang Purnomosidi D. P.*).
2. Boleh menggunakan produk yang ada disini untuk keperluan apapun jika point 1 di atas terpenuhi.
3. Boleh membuat produk derivatif dari produk yang ada disini sepanjang perubahan-perubahan yang dilakukan diberitahukan ke kami dan di-share dengan menggunakan lisensi yang sama.

Untuk penggunaan selain ketentuan tersebut, silahkan menghubungi:

- 1 Dr. Bambang Purnomosidi D. P.
- 2 Magister Teknologi Informasi
- 3 STMIK AKAKOM
- 4 Jl. Raya Janti no 143 Yogyakarta
- 5 bdp@akakom.ac.id
- 6 Phone: 0274 486664

2 Pengenalan Go

Apa itu Go?

Go adalah nama bahasa pemrograman sekaligus nama implementasi dalam bentuk kompilator (`compiler`). Untuk pembahasan berikutnya, istilah `Go` akan mengacu juga pada spesifikasi bahasa pemrograman serta peranti pengembangannya.

Lisensi Go

Go didistribusikan dengan menggunakan lisensi modifikasi dari BSD. Lisensi lengkap dari Go bisa diakses di URL Lisensi Go. Secara umum, penggunaan lisensi ini mempunyai implikasi sebagai berikut:

- boleh digunakan untuk keperluan komersial maupun non-komersial tanpa batasan
- boleh memodifikasi sesuai keperluan
- boleh mendistribusikan
- boleh memberikan sublisensi ke pihak lain
- boleh memberikan garansi
- tidak boleh menggunakan merk dagang Go
- tanpa jaminan dan jika terjadi kerusakan terkait penggunaan software ini maka pemberi lisensi tidak bisa dituntut
- jika mendistribusikan harus mengikutsertakan pemberitahuan hak cipta.

Instalasi Go

Distribusi Go

Go tersedia pada berbagai platform. Proyek Go sendiri secara resmi mendukung platform Linux, FreeBSD, MacOSX, dan Windows. Dukungan tersebut merupakan dukungan resmi dan distribusi `binary executable` dari berbagai platform tersebut tersedia di repository downloads Go seperti bisa dilihat di gambar berikut:

The Go Programming Language

Documents Packages The Project Help Blog Search

Downloads

After downloading a binary release suitable for your system, please follow the [installation instructions](#).

If you are building from source, follow the [source installation instructions](#).

See the [release history](#) for more information about Go releases.

Featured downloads

Microsoft Windows
Windows XP or later, Intel 64-bit processor
go1.7.1.windows-amd64.msi (72MB)
SHA256: c043ee8053eb48ac259f25021e000f8c80645bea02c55659a21d0c70756c

Apple OS X
OS X 10.8 or later, Intel 64-bit processor
go1.7.1.darwin-amd64.pkg (78MB)
SHA256: 62544484135df790c2069416ca58ab79023611bcb096f7ae1c863cd056867b7e

Linux
Linux 2.6.23 or later, Intel 64-bit processor
go1.7.1.linux-amd64.tar.gz (78MB)
SHA256: 43ba021c9014c0e8b17393ac108378d37bd853a351a6c740e5432c1be4182

Source
go1.7.1.src.tar.gz (13MB)
SHA256: 2b843f133b81b7995f26d0cb64bbdbb9d0704b90c44df45f844d28881ad442d3

Stable versions

go1.7.1 ▼

File name	Kind	OS	Arch	Size	SHA256 Checksum
go1.7.1.src.tar.gz	Source			13MB	2b843f133b81b7995f26d0cb64bbdbb9d0704b90c44df45f844d28881ad442d3

Gambar 2.1: Repository downloads Go

Dengan dukungan tersebut, Proyek Go akan menerima laporan [bugs](#) terkait dengan distribusi pada berbagai platform tersebut. Meski demikian, bukan berarti platform-platform lain tidak bisa menggunakan Go karena distribusi dalam bentuk kode sumber tersedia dan telah berhasil dikompilasi ke berbagai platform: NetBSD, OpenBSD, DragonFlyBSD, dan lain-lain. Informasi mengenai platform-platform yang mungkin bisa digunakan oleh Go bisa diperoleh di Wiki.

Download dan Install Go

Download dan instalasi Go pada tulisan ini adalah download dan instalasi untuk lebih dari satu versi Go dan masing-masing menggunakan workspace sendiri-sendiri. Hal ini disebabkan karena seringkali software yang dibangun ditargetkan untuk lebih dari satu versi, misalnya Go 1.5 ke atas (Go 1.5.x, 1.6.x, dan 1.7.x). Kondisi ini menjadi tidak sederhana karena penulis tidak ingin mencampurkan source code yang dibuat menggunakan masing-masing versi. Go sendiri menyarankan untuk menggunakan satu workspace untuk semua proyek Go yang kita buat. Satu workspace saja tidak masalah jika hanya menargetkan satu versi. Di bagian ini penulis akan menjelaskan konfigurasi yang saya gunakan untuk menangani masalah tersebut.

```
1 Catatan:
2
3 Go akan diinstall di direktori /opt/software/go-dev-tools/go/goVERSI
4
5 VERSI = versi dari Go yang akan diinstall, misalnya go1.7.3
6 Lokasi instalasi tersebut saya gunakan karena saya mempunyai lebih dari
   1 versi Go, jika nanti ada versi lainnya, versi lain tersebut akan
   saya install (misal versi 1.6.3) di /opt/software/go-dev-tools/go/
   go1.6.3
```

Meski mendukung banyak platform, di buku ini hanya akan dibahas penggunaan Go di platform Linux. Pada dasarnya peranti pengembang yang disediakan sama. Silahkan menyesuaikan dengan platform yang anda gunakan. Untuk instalasi berikut ini, ambil distribusi yang sesuai dengan platform di komputer anda. Untuk pembahasan ini, digunakan `go1.7.3.linux-amd64.tar.gz`. Setelah itu, ikuti langkah-langkah berikut:

```
1 $ ls -la
2 total 80632
3 drwxr-xr-x 1 bdpd users      52 Oct 20 21:38 .
4 drwxr-xr-x 1 bdpd users     198 Jun 13 13:17 ..
5 -rw-r--r-- 1 bdpd users 82565628 Oct 20 02:02 go1.7.3.linux-amd64.tar.
   gz
6 $ mkdir -p /opt/software/go-dev-tools/go
7 $ cd /opt/software/go-dev-tools/go
8 $ tar -xvf ~/master/go/1.7/go1.7.3.linux-amd64.tar.gz
9 $ mv go go1.7.3
```

Setelah menjalankan langkah-langkah di atas, Go sudah terinstall di direktori `/opt/software/go-dev-tools/go/go1.7.3`

Konfigurasi Variabel Lingkungan Sistem Operasi, Compiler Go, dan Workspace

Untuk konfigurasi kompiler, ada tiga langkah yang perlu dilakukan: download, ekstrak pada lokasi tertentu, dan terakhir setting environment variables. Pada konfigurasi ini, compiler dan workspace berada pada `/opt/software/go-dev-tools/`. Lokasi ini selanjutnya akan kita sebut dengan `GODEVTOOLS_HOME`. Setelah download dan install compiler Go seperti langkah di atas, buat struktur direktori sebagai berikut (untuk go1.7.3 sudah dibuat dengan cara di atas):

```
10:30:53-bpdp@archera:/opt/software/go-dev-tools$  
.  
├── go  
│   ├── go1.2.2  
│   ├── go1.3.3  
│   ├── go1.4.3  
│   ├── go1.5.4  
│   ├── go1.6.3  
│   └── go1.7.0  
└── workspace  
    ├── go12  
    ├── go13  
    ├── go14  
    ├── go15  
    ├── go16  
    └── go17  
  
10:32:04-bpdp@archera:/opt/software/go-dev-tools$
```

Gambar 2.2: Struktur direktori

Direktori go digunakan untuk lokasi semua versi compiler Go. Direktori workspace digunakan untuk menyimpan source code yang kita buat sesuai dengan versi Go yang kita targetkan. Untuk setiap direktori di workspace, buat struktur dan 1 file env.sh sebagai berikut:

```
10:43:48-bpdp@archera:/opt/software/go-dev-tools$ tree -L 2 workspace/go16  
workspace/go16  
├── bin  
├── env.sh  
├── pkg  
└── src  
  
3 directories, 1 file  
10:43:56-bpdp@archera:/opt/software/go-dev-tools$
```

Gambar 2.3: Struktur workspace

Isi dari file env.sh adalah sebagai berikut:


```
1 export GOPATH=`pwd`  
2 export PATH=$PATH:$GOPATH/bin
```

Go menggunakan beberapa variabel lingkungan sistem operasi. Supaya berfungsi dengan baik, tetapkan nilai-nilai variabel lingkungan tersebut di file inisialisasi shell (penulis menggunakan Bash, sehingga file-file inisialisasi diletakkan di `$HOME/.bashrc`). Meski bisa diletakkan pada file tersebut, penulis menyarankan untuk meletakkan pada suatu file text biasa dan kemudian di - *source*. Pada bagian ini, penulis akan meletakkan di file `%HOME/env/go/go1.7.3`.

```
1 GODEVTOOLS_HOME=/opt/software/go-dev-tools  
2  
3 GO_HOME=$GODEVTOOLS_HOME/go/go1.7.3  
4 LITEIDE_HOME=$GODEVTOOLS_HOME/liteide  
5 GOTTOOLS=$GODEVTOOLS_HOME/workspace/go17/bin  
6 GO3RDPARTYTOOLS=$GODEVTOOLS_HOME/go-3rd-party-tools  
7  
8 export GOROOT=$GO_HOME  
9 export GOOS=linux  
10 export GOARCH=amd64  
11 export GOHOSTOS=linux  
12 export GOHOSTARCH=amd64  
13  
14 # https://golang.org/cmd/go/#hdr-GOPATH_environment_variable  
15 # a) If you don't set your GOBIN env variable,  
16 # you get the Go compiler binaries going in  
17 # GOROOT/bin whereas the your binaries are going  
18 # in GOPATH/bin. (I personally like this separation of  
19 # binaries.)  
20 # (b) If you set your GOBIN to anything, then both the Go  
21 # binaries and your binaries are going to GOBIN.  
22 #export GOBIN=$GOROOT/bin  
23  
24 export PATH=$PATH:$GO_HOME/bin:$LITEIDE_HOME/bin:$GOTTOOLS:  
    $GO3RDPARTYTOOLS/bin  
25  
26 alias godev='cd $GODEVTOOLS_HOME/workspace/go17'
```

Dengan memasukkan beberapa variabel lingkungan tersebut ke file, saat kita ingin menggunakan Go, tinggal di - *source* sebagai berikut:

```
1 $ source ~/env/go/go1.7.3
```

Setelah itu, Go bisa digunakan. Untuk melihat hasil, eksekusi perintah `go env`, hasilnya seharusnya adalah sebagai berikut:

```
1 $ go env
2 GOARCH="amd64"
3 GOBIN=""
4 GOEXE=""
5 GOHOSTARCH="amd64"
6 GOHOSTOS="linux"
7 GOOS="linux"
8 GOPATH=""
9 GORACE=""
10 GOROOT="/opt/software/go-dev-tools/go/go1.7.3"
11 GOTOOLDIR="/opt/software/go-dev-tools/go/go1.7.3/pkg/tool/linux_amd64"
12 CC="gcc"
13 GOGCCFLAGS="-fPIC -m64 -pthread -fmessage-length=0 -fdebug-prefix-map=/
    tmp/go-build947764430=/tmp/go-build -gno-record-gcc-switches"
14 CXX="g++"
15 CGO_ENABLED="1"
16 $
```

Saat bekerja menggunakan Go, pada dasarnya kita akan menemukan berbagai macam proyek yang bisa dikategorikan menjadi 2 berdasarkan output dari proyek tersebut: (1) ready-to-use application—aplikasi yang siap dipakai, biasanya didistribusikan dalam bentuk binary executable(s) atau kode sumber seperti nsq, Hugo, dan lain-lain. (2) pustaka / library. Untuk dua kategori ini, ada dua perlakuan.

Ready-to-use application

Untuk kategori ini, siapkan lokasi khusus di media penyimpanan untuk menyimpan hasil binary executable, setelah itu set PATH, GOPATH dan go get -u -v repo. Berikut adalah setting pada komputer penulis:

```

21:03:01-bpdp@archera: /opt/software/go-dev-tools/go-3rd-party-tools$ tree . -L 1
.
├── bin
├── env.sh
├── go-pkg-needed.sh
├── pkg
└── src

3 directories, 2 files
21:03:11-bpdp@archera: /opt/software/go-dev-tools/go-3rd-party-tools$ cat env.sh
export GOPATH=`pwd`
export PATH=$PATH:$GOPATH/bin
21:03:50-bpdp@archera: /opt/software/go-dev-tools/go-3rd-party-tools$ cat go-pkg-needed.sh
#!/usr/bin/bash
go get -u -v github.com/nsf/gocode
go get -u -v github.com/rogppe/godef
go get -u -v github.com/golang/lint/golint
go get -u -v github.com/lukehoban/go-outline
go get -u -v github.com/sqs/goreturns

```

Gambar 2.4: Struktur direktori untuk 3rd party tools

Isi dari file `go-pkg-needed.sh` adalah sebagai berikut, anda bisa menambah atau mengurangi sesuai kebutuhan:

```

1  #!/usr/bin/bash
2  go get -u -v github.com/nsf/gocode
3  go get -u -v github.com/rogppe/godef
4  go get -u -v github.com/golang/lint/golint
5  go get -u -v github.com/lukehoban/go-outline
6  go get -u -v github.com/sqs/goreturns
7  go get -u -v golang.org/x/tools/cmd/gorename
8  go get -u -v github.com/tpng/gopkgs
9  go get -u -v github.com/newhook/go-symbols
10 go get -u -v golang.org/x/tools/cmd/guru
11 go get -u -v github.com/derekparker/delve/cmd/dlv
12 go get -u -v github.com/pointlander/peg
13 go get -u -v github.com/songgao/colargo
14 go get -u -v github.com/constabulary/gb/...
15 # glide now taken from release page
16 go get -u -v github.com/Masterminds/glide
17 go get -u -v github.com/motemen/gore
18 go get -u -v github.com/onsi/ginkgo/ginkgo
19 go get -u -v github.com/onsi/gomega
20 go get -u -v github.com/smartystreets/goconvey
21 go get -u -v github.com/blynn/nex
22 go get -u -v golang.org/x/tools/cmd/goimports
23 go get -u -v golang.org/x/tools/cmd/gotype

```

```
24 go get -u -v github.com/zmb3/gogetdoc
25 go get -u -v github.com/govend/govend
```

Dengan konfigurasi seperti itu, kerjakan berikut ini untuk install:

```
1 $ source env/go/go1.7.3
2 $ cd /opt/software/go-dev-tools/go-3rd-party-tools
3 $ source env.sh
4 $ ./go-pkg-needed.sh
```

Setelah proses sebentar, hasil binary executables akan diletakkan pada \$GO3RDPARTYTOOLS/bin dan bisa kita jalankan langsung.

Catatan: jangan meletakkan paket-paket executables ini dalam skema vendoring karena bisa mencampuradukkan dan mengacaukan letak dari hasil kompilasi.

Pustaka / Library

Untuk pustaka, sebaiknya menggunakan vendoring (aktif default mulai Go 1.6+). Pustaka ini biasanya kita gunakan pada proyek kita. Untuk kasus seperti ini, gunakan vendoring. Dengan struktur workspace seperti yang telah dijelaskan di atas, gunakan tools yang mendukung vendoring (penulis menggunakan govend). Direktori vendor diletakkan pada src/ dan pustaka di src/vendor dikelola menggunakan govend. Berikut ini contoh struktur direktori yang penulis gunakan:

```
21:23:54-bdp@archera:/opt/software/go-dev-tools/workspace/go17/src$ tree . -L 4
.
├── github.com
│   ├── bdp
│   │   └── hayy
│   │       ├── examples
│   │       ├── hayy.go
│   │       ├── README.md
│   │       └── utils
│   └── vendor
│       ├── github.com
│       │   ├── inconstreveable
│       │   │   └── mousetrap
│       │   ├── spf13
│       │   │   ├── cobra
│       │   │   └── pflag
└──
```

12 directories, 2 files
21:26:32-bdp@archera:/opt/software/go-dev-tools/workspace/go17/src\$

Gambar 2.5: Struktur direktori untuk vendoring

Pada skema di atas, proyek saya adalah hayy dan bisa di akses menggunakan skema import github.com/bdp/hayy. Proyek saya tersebut menggunakan pustaka github.com/spf13/cobra

(github.com/inconshreveable/mousetrap dan github.com/spf13/pflag adalah dependency dari github.com/spf13/cobra). Dengan demikian, vendoring ini memungkinkan kita mengelola dependency dengan lebih baik dan membuat source code proyek kita “bersih” dari berbagai pustaka dependencies.

Menguji Instalasi Go

Kode sumber Go yang kita buat bisa dijalankan / dieksekusi tanpa harus dikompilasi (jadi seperti script Python atau Ruby) atau bisa juga dikompilasi lebih dulu untuk menghasilkan **binary executable**. Selain menghasilkan **binary executable**, sebenarnya ada paket pustaka yang dimaksudkan untuk digunakan dalam program (disebut sebagai **package**). Package akan dibahas lebih lanjut pada bab-bab berikutnya.

Untuk menguji, buat program sederhana seperti listing `hello.go`. Setelah itu, gunakan `go run namafile.go` untuk menjalankan secara langsung atau dikompilasi lebih dulu dengan `go build namafile.go`.

```
1 // hello.go
2 package main
3
4 import "fmt"
5
6 func main() {
7     fmt.Printf("hello, world\n")
8 }
```

Berikut ini adalah langkah-langkah untuk mengeksekusi `hello.go`:

```
1 $ go run hello.go
2 hello, world
3 $ go build hello.go
4 $ ls -la
5 total 1624
6 drwxr-xr-x 1 bdpd users 102 Sep 12 21:20 .
7 drwxr-xr-x 1 bdpd users 288 Aug 15 11:24 ..
8 -rwxr-xr-x 1 bdpd users 1650350 Sep 12 21:20 hello
9 -rw-r--r-- 1 bdpd users 162 Aug 15 11:28 hello-formatted.go
10 -rw-r--r-- 1 bdpd users 86 Aug 15 11:26 hello.go
11 -rw-r--r-- 1 bdpd users 170 Aug 15 11:28 hello-unformatted.go
12 $ ./hello
13 hello, world
14 $ file hello
```

```
15 hello: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically
    linked, not stripped
16 $
```

Memahami Lingkungan Peranti Pengembangan Go

Saat menginstall Go, kita akan memperoleh 3 buah file `binary executable`:

```
1 $ ls /opt/software/go-dev-tools/go/go1.7.3/bin/
2 total 27496
3 drwxr-xr-x 1 bdpdp users      24 Sep  8 02:38 .
4 drwxr-xr-x 1 bdpdp users    230 Sep  8 02:38 ..
5 -rwxr-xr-x 1 bdpdp users 9954032 Sep  8 02:35 go
6 -rwxr-xr-x 1 bdpdp users 15156929 Sep  8 02:37 godoc
7 -rwxr-xr-x 1 bdpdp users  3036195 Sep  8 02:35 gofmt
8 $
```

Penjelasan untuk masing-masing akan diuraikan di sub-sub bab berikut.

go

`go` merupakan peranti untuk mengelola kode sumber Go yang kita buat. Beberapa argumen dari `go` adalah:

```
1 $ go
2 Go is a tool for managing Go source code.
3
4 Usage:
5
6     go command [arguments]
7
8 The commands are:
9
10    build      compile packages and dependencies
11    clean      remove object files
12    doc        show documentation for package or symbol
13    env       print Go environment information
14    fix        run go tool fix on packages
15    fmt        run gofmt on package sources
16    generate    generate Go files by processing source
17    get        download and install packages and dependencies
```

```
18  install    compile and install packages and dependencies
19  list       list packages
20  run        compile and run Go program
21  test      test packages
22  tool       run specified go tool
23  version    print Go version
24  vet        run go tool vet on packages
25
26  Use "go help [command]" for more information about a command.
27
28  Additional help topics:
29
30  c          calling between Go and C
31  buildmode  description of build modes
32  filetype   file types
33  gopath     GOPATH environment variable
34  environment environment variables
35  importpath import path syntax
36  packages   description of package lists
37  testflag   description of testing flags
38  testfunc   description of testing functions
39
40  Use "go help [topic]" for more information about that topic.
```

godoc

`godoc` merupakan peranti untuk menampilkan dokumentasi paket pustaka standar Go atau menampilkan server untuk dokumentasi Go (mirip seperti yang terdapat pada website dokumentasi Go).

```
1  $ godoc
2  usage: godoc package [name ...]
3      godoc -http=:6060
4  -analysis string
5      comma-separated list of analyses to perform (supported: type,
6      pointer). See http://golang.org/lib/godoc/analysis/help.html
7  -ex
8      show examples in command line mode
9  -goroot string
10     Go root directory (default "/opt/software/go-dev-tools/go/go1.7.3")
11  -html
```

```
11     print HTML in command-line mode
12 -http string
13     HTTP service address (e.g., ':6060')
14 -httptest.serve string
15     if non-empty, httptest.NewServer serves on this address and
        blocks
16 -index
17     enable search index
18 -index_files string
19     glob pattern specifying index files; if not empty, the index is
        read from these files in sorted order
20 -index_interval duration
21     interval of indexing; 0 for default (5m), negative to only
        index once at startup
22 -index_throttle float
23     index throttle value; 0.0 = no time allocated, 1.0 = full
        throttle (default 0.75)
24 -links
25     link identifiers to their declarations (default true)
26 -maxresults int
27     maximum number of full text search results shown (default
        10000)
28 -notes string
29     regular expression matching note markers to show (default "BUG"
        )
30 -play
31     enable playground in web interface
32 -q      arguments are considered search queries
33 -server string
34     webserver address for command line searches
35 -src
36     print (exported) source in command-line mode
37 -tabwidth int
38     tab width (default 4)
39 -templates string
40     directory containing alternate template files
41 -timestamps
42     show timestamps with directory listings
43 -url string
44     print HTML for named URL
45 -v      verbose mode
46 -write_index
47     write index to a file; the file name must be specified with -
```



```
        index_files
48     -zip string
49         zip file providing the file system to serve; disabled if empty
```

gofmt

`gofmt` merupakan peranti untuk mem-format kode sumber dalam bahasa pemrograman Go.

```
1 $ gofmt --help
2 usage: gofmt [flags] [path ...]
3     -cpuprofile string
4         write cpu profile to this file
5     -d      display diffs instead of rewriting files
6     -e      report all errors (not just the first 10 on different lines)
7     -l      list files whose formatting differs from gofmt's
8     -r string
9         rewrite rule (e.g., 'a[b:len(a)] -> a[b:]')
10    -s      simplify code
11    -w      write result to (source) file instead of stdout
```

Untuk melihat bagaimana `gofmt` bisa digunakan untuk membantu memformat kode sumber, buat kode sumber sederhana berikut ini:

```
1 // hello-unformatted.go
2 package main
3 import "fmt"
4 func main() {
5     fmt.Printf("halo\n") // menampilkan tulisan
6     fmt.Printf("dunia")  // ini tulisan baris kedua
7 }
```

Format file kode sumber di atas sebagai berikut:

```
1 $ gofmt hello-unformatted.go > hello-formatted.go
```

Hasilnya adalah sebagai berikut:

```
1 // hello-formatted.go
2 package main
3
4 import "fmt"
5
```

```
6 func main() {  
7     fmt.Printf("halo\n") // menampilkan tulisan  
8     fmt.Printf("dunia")  // ini tulisan baris kedua  
9 }
```

3 IDE Untuk Go

IDE ([Integrated Development Environment](#)) merupakan software yang digunakan oleh pemrogram dan pengembang software untuk membangun software. IDE berisi berbagai fasilitas komprehensif yang diperlukan para pemrogram untuk membantu mereka dalam membangun software aplikasi. Secara minimal, biasanya IDE terdiri atas editor teks (untuk mengetikkan kode sumber), debugger (pencari bugs), 'syntax highlighting, code completion, serta dokumentasi / help. Bab ini akan membahas beberapa software yang bisa digunakan. Sebenarnya menggunakan editor teks yang menghasilkan file text / ASCII murni sudah cukup untuk bisa menuliskan dan kemudian mengkompilasi kode sumber. Pada bab ini akan dibahas Vim sebagai editor teks dan LiteIDE sebagai software IDE yang lebih lengkap untuk Go, tidak sekedar hanya untuk menuliskan kode sumber.

Menggunakan Vim

Untuk menggunakan Vim, ada plugin utama serta berbagai plugin pendukung yang bisa digunakan. Sebaiknya, menggunakan [pathogen](#) untuk mempermudah pengelolaan berbagai plugin tersebut. Bagian ini akan menjelaskan berbagai konfigurasi serta instalasi yang diperlukan sehingga Vim bisa menjadi peranti untuk pengembangan aplikasi menggunakan Go.

Instalasi dan Konfigurasi Pathogen

Pathogen adalah plugin dari Tim Pope yang digunakan untuk mempermudah pengelolaan plugin. Kode sumber dari Pathogen bisa diperoleh di repository Github. Untuk instalasi, ikuti langkah berikut:

```
1 $ cd
2 $ mkdir .vim/autoload
3 $ mkdir .vim/bundle
4 $ cd .vim/autoload
5 $ wget https://raw.githubusercontent.com/tpope/vim-pathogen/master/
  autoload/pathogen.vim
6 --2016-08-15 11:16:50-- https://raw.githubusercontent.com/tpope/vim-
  pathogen/master/autoload/pathogen.vim
```

```

7  Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
   151.101.100.133
8  Connecting to raw.githubusercontent.com (raw.githubusercontent.com)
   |151.101.100.133|:443... connected.
9  HTTP request sent, awaiting response... 200 OK
10 Length: 12474 (12K) [text/plain]
11 Saving to: 'pathogen.vim'
12
13 pathogen.vim
   100%[=====>] 12.18K
   --.-KB/s   in 0.02s
14
15 2016-08-15 11:16:56 (512 KB/s) - 'pathogen.vim' saved [12474/12474]
16
17 $

```

Setelah itu, untuk menggunakan Pathogen, letakkan aktivasinya di `$HOME/.vimrc` atau di `$HOME/.vim/vimrc` (saya pilih lokasi yang kedua) sebagai berikut:

```
1 execute pathogen#infect()
```

Setelah itu, semua plugin tinggal kita ambil dari repository (bisa dari github, bitbucket, dan lain-lain) langsung di-copy satu direktori ke direktori `$HOME/.vim/bundle`.

Instalasi dan Konfigurasi Plugin Golang dan Plugin Pendukung

Setelah selesai melakukan instalasi Pathogen, berbagai plugin yang diperlukan bisa diambil langsung dari Internet. Berikut ini adalah daftar yang digunakan penulis:

- Colorschemes: untuk tema warna dari Vim.
- Nerdtree: untuk menampilkan file-file dalam struktur pohon di sebelah kiri sehingga memudahkan navigasi.
- vim-go: plugin utama agar Vim mengenali kode sumber Go.

Cara instalasi:

```

1 $ cd
2 $ cd .vim/bundle
3 $ git clone <masing-masing lokasi plugin>

```

Hasil dari menjalankan `vim 'ataugvim` melalui shell untuk menulis kode sumber Go bisa dilihat pada gambar berikut ini:

```

1 /*
2  aplikasi.go
3
4  Contoh program sederhana untuk menjelaskan
5  struktur program Go untuk aplikasi executable
6
7  (c) bdp.name
8
9  */
10
11 // Program Go diawali dengan nama pake.
12 // Paket untuk aplikasi executable selalu berada
13 // pada paket main.
14 package main
15
16 // pustaka standar yang diperlukan
17 // Jika hanya satu:
18 // import "fmt"
19 // Jika lebih dari satu:
20 import (
21     "fmt"
22     "os"
23 )
24
25 // "Fungsi" merupakan satuan terintegrasi dari
26 // program Go, selalu diberi nama "main" untuk
27 // aplikasi executable.
28 func main() {
29
30     // ini adalah kode sumber / program Go
31     // akan dijelaskan lebih lanjut, no need to
32     // care a lot right now.
33     var (
34         user string
35         homeDir string
36         goHome string
37     )
38
39     user = os.Getenv("USER")
40     homeDir = os.Getenv("HOME")
41 }

```

vim-go/src/go/bab-03 aplikasi.go
[1-aplikasi.go]

Gambar 3.1: vim-go

Autocompletion

Vim menyediakan fasilitas `autocompletion` melalui `Omniautocompletion`. Fasilitas ini sudah terinstall saat kita menginstall Vim. Untuk mengaktifkan fasilitas ini untuk keperluan Go, kita harus menginstall dan mengaktifkan Gocode. Gocode sudah terinstall setelah selesai mengerjakan instalasi di bab 1. Hasil dari instalasi Gocode adalah file `executable binary $GOPATH/bin/gocode` (sesuai letak GOPATH di env.sh). Sebelum menggunakan Vim, aktifkan dulu gocode dengan mengeksekusi `gocode` melalui shell. Setelah itu, tambahkan satu baris di `$HOME/.vim/vimrc`: `set ofu=syntaxcomplete#Complete` di bawah baris `filetype plugin indent on`.

Kode sumber lengkap dari `$HOME/.vim/vimrc` yang penulis gunakan bisa dilihat pada listing berikut ini:

```
1 execute pathogen#infect()
2
3 syntax on
4 filetype plugin indent on
5 set ofu=syntaxcomplete#Complete
6
7 if has("gui_running")
8     colorscheme asmanian_blood
9 else
10    colorscheme slate
11 endif
12
13 set smartindent
14 set tabstop=2
15 set shiftwidth=2
16 set expandtab
17
18 autocmd vimenter * NERDTree
19 autocmd vimenter * if !argc() | NERDTree | endif
20 autocmd bufenter * if (winnr("$") == 1 && exists("b:NERDTreeType") && b
    :NERDTreeType == "primary") | q | endif
21
22 let g:NERDTreeDirArrows=0
23
24 let g:cssColorVimDoNotMessMyUpdatetime = 1
25 set guifont=Liberation\ Mono\ 11
26
27 set number
28 set numberwidth=4
29 set coptions+=n
30 highlight LineNr term=bold cterm=NONE ctermfg=DarkGrey ctermbg=NONE gui
    =NONE guifg=DarkGrey guibg=NONE
31
32 set grepprg=grep\ -nH\ $*
```

Untuk mengaktifkan completion, kita harus masuk ke mode `Insert` dari Vim, setelah itu tekan `Ctrl-X`, `Ctrl-O` secara cepat. Hasil `autocomplete` bisa dilihat di gambar berikut ini:

The screenshot shows the LiteIDE Go IDE interface. The menu bar includes 'Edit', 'Tools', 'Syntax', 'Buffers', 'Window', 'Plugin', 'Themes', and 'Help'. The toolbar contains various icons for file operations, editing, and running. The editor window displays a Go source file named 'pointer.go' with the following code:

```

1 func Scanln(a ...interface{}) (n int, err error)
2
3     func Errorf(format string, a ...interface{}) error
4     func Fprintf(w io.Writer, format string, a ...interface{}) (n int, err error)
5     func Fprintln(w io.Writer, a ...interface{}) (n int, err error)
6     var func Fscanf(r io.Reader, a ...interface{}) (n int, err error)
7     fmt func Fscanf(r io.Reader, format string, a ...interface{}) (n int, err error)
8     fmt func Fscanln(r io.Reader, a ...interface{}) (n int, err error)
9     fmt func Print(a ...interface{}) (n int, err error)
10    fmt func Printf(format string, a ...interface{}) (n int, err error)
11    var func Println(a ...interface{}) (n int, err error)
12    fmt func Scan(a ...interface{}) (n int, err error)
13    fmt func Scanf(format string, a ...interface{}) (n int, err error)
14    fmt func Scanln(a ...interface{}) (n int, err error)
15    fmt func Sprintf(format string, a ...interface{}) string
16    var func Sprintf(format string, a ...interface{}) string
17    fmt func Sprintln(a ...interface{}) string
18    fmt func Sscan(str string, a ...interface{}) (n int, err error)
19    fmt func Sscanf(str string, format string, a ...interface{}) (n int, err error)
20    fmt func Sscanln(str string, a ...interface{}) (n int, err error)
21    var type Formatter interface
22    fmt type GoStringer interface
23    ptr type ScanState interface
24    fmt type Scanner interface
25    fmt type State interface
26    type Stringer interface
27    fmt.Errorf()
28
29 }

```

The status bar at the bottom shows the file path 'e/bdpd/src/go/pointer/pointer.go' and indicates 'mini completion (AOANAP) match 13 of 25'.

Gambar 3.2: Go completion

Menggunakan LiteIDE



Gambar 3.3: LiteIDE

LiteIDE dibuat oleh visualfc dan tersedia dalam bentuk kode sumber maupun binary. Kode sumber bisa diperoleh di repo GitHub. Installer executable bisa diperoleh di Sourceforge

Instalasi di Linux sangat mudah, hanya tinggal mengekstrak file yang kita download pada suatu direktori dan jika ingin menjalankan cukup dengan mengeksekusi file `$LITEIDE_HOME/bin/liteide` (`cd $LITEIDE_HOME/bin; ./liteide &`)

Software IDE Lain

Vim dan LiteIDE hanyalah beberapa peranti yang bisa digunakan oleh pengembang. Distribusi Go juga menyediakan dukungan untuk berbagai peranti lunak lain:

- Emacs. Dukungan untuk Go diwujudkan dalam fasilitas `add-on`. Untuk Emacs 24 ke atas, bisa diinstall melalui manajer paket (M-x `package-list-packages`), cari dan install `go-mode`. Emacs juga mendukung `gocode` untuk `completion`.
- Eclipse. Dukungan untuk Go diwujudkan melalui plugin `goclipse`, bisa diperoleh di <https://code.google.com/p/goclipse/>.
- Selain software-software yang telah disebutkan, rata-rata IDE / Editor sudah mempunyai dukungan terhadap bahasa pemrograman Go (JEdit, Sublime-text, Notepad++, dan lain-lain).

4 Dasar-dasar Pemrograman Go

Struktur Program Go

Program Aplikasi Sederhana - 1 File `binary executable` Utama

Suatu aplikasi `executable` (artinya bisa dijalankan secara langsung oleh sistem operasi) mempunyai struktur seperti yang terlihat pada listing berikut ini:

```
1  /*
2      aplikasi.go
3
4      Contoh program sederhana untuk menjelaskan
5      struktur program Go untuk aplikasi executable
6
7      (c) bdpd.xyz
8
9  */
10
11 // Program Go diawali dengan nama paket.
12 // Paket untuk aplikasi executable selalu berada
13 // pada paket main.
14 package main
15
16 // pustaka standar yang diperlukan
17 // Jika hanya satu:
18 // import "fmt"
19 // Jika lebih dari satu:
20 import (
21     "fmt"
22     "os"
23 )
24
25 // "Fungsi" merupakan satuan terintegrasi dari
26 // program Go, selalu diberi nama "main" untuk
27 // aplikasi executable.
```

```
28 func main() {
29
30     // ini adalah kode sumber / program Go
31     // akan dijelaskan lebih lanjut, abaikan
32     // jika belum paham
33     var (
34         user    string
35         homeDir string
36         goHome  string
37     )
38
39     user = os.Getenv("USER")
40     homeDir = os.Getenv("HOME")
41     goHome = os.Getenv("GOROOT")
42
43     fmt.Printf("Halo %s", user)
44     fmt.Printf("\nHome anda di %s", homeDir)
45     fmt.Printf("\nAnda menggunakan Go di %s", goHome)
46     fmt.Printf("\n")
47
48 }
```

Untuk menjalankan kode sumber di atas, ikuti langkah-langkah berikut:

Tanpa Proses Kompilasi

```
1 $ go run aplikasi.go
2 Halo bdpd
3 Home anda di /home/bdpd
4 Anda menggunakan Go di /home/bdpd/software/go-dev-tools/go/go1.7.3
```

Mengkompilasi Menjadi *Binary Executable*

```
1 $ go build aplikasi.go
2 $ ls -la
3 total 1620
4 drwxr-xr-x 1 bdpd users      38 Sep 12 22:49 .
5 drwxr-xr-x 1 bdpd users    288 Aug 15 11:24 ..
6 -rwxr-xr-x 1 bdpd users 1654480 Sep 12 22:49 aplikasi
7 -rw-r--r-- 1 bdpd users    900 Sep 12 22:49 aplikasi.go
8 $ ./aplikasi
```

```
9  Halo bdpd
10 Home anda di /home/bdpd
11 Anda menggunakan Go di /opt/software/go-dev-tools/go/go1.7.3
12 $ file aplikasi
13 aplikasi: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
    statically linked, not stripped
14 $
```

Pustaka / Library / Package

Ada kalanya, para software developer membangun pustaka yang berisi berbagai fungsionalitas yang bisa digunakan kembali suatu saat nanti. Untuk keperluan ini, Go menyediakan fasilitas untuk membangun library dalam bentuk kumpulan fungsi. Kumpulan fungsi ini nantinya akan diletakkan pada suatu repo tertentu sehingga bisa langsung di `go get <lokasi repo pustaka>`. Pada penjelasan berikut ini, kita akan membangun suatu aplikasi kecil (hello) yang menggunakan suatu pustaka yang sebelumnya sudah kita bangun (stringutil/Reverse - untuk membalik kata). Kode sumber diambil dari How to write Go code.

Mengatur Workspace

Ada 3 direktori di workspace yang disiapkan: bin, pkg, dan src: * bin: berisi hasil kompilasi aplikasi (hello) * pkg: berisi hasil kompilasi pustaka (stringutil) * src: kode sumber untuk pustaka serta aplikasi. Pada direktori tersebut, juga dibuat `env.sh`.

```
1  $ ls
2  total 4
3  drwxr-xr-x 1 bdpd users 30 Sep 12 23:16 .
4  drwxr-xr-x 1 bdpd users 56 Sep 12 23:05 ..
5  drwxr-xr-x 1 bdpd users 10 Sep 12 23:16 bin
6  -rw-r--r-- 1 bdpd users 50 Sep 12 23:05 env.sh
7  drwxr-xr-x 1 bdpd users 22 Sep 12 23:16 pkg
8  drwxr-xr-x 1 bdpd users 32 Sep 12 23:07 src
9  $ cat env.sh
10 export GOPATH=`pwd`
11 export PATH=$PATH:$GOPATH/bin
12 $ source ~/env/go/go1.7.3
13 $ source env.sh
14 $
```

Semua kode sumber, baik untuk pustaka ataupun aplikasi akan diletakkan pada pola direktori tertentu. Go menggunakan pola repo untuk penamaan / pengelompokan aplikasi atau pustaka meskipun belum dimasukkan ke repo di Internet. Sebaiknya membiasakan diri sejak awal menggunakan pola tersebut meskipun belum akan dimasukkan ke repositori di Internet.

Membuat Pustaka

Kode sumber untuk pustaka ini akan diletakkan di `src/github.com/bdp/stringutil`. Paket yang dibuat dengan penamaan ini, nantinya akan diacu dalam `import` sebagai `github.com/bdp/stringutil`.

```
1  /*
2      src/github.com/bdp/stringutil/reverse.go
3      diambil dari https://golang.org/doc/code.html
4  */
5  package stringutil
6
7  // Reverse returns its argument string reversed rune-wise left to right
8  .
9  func Reverse(s string) string {
10     r := []rune(s)
11     for i, j := 0, len(r)-1; i < len(r)/2; i, j = i+1, j-1 {
12         r[i], r[j] = r[j], r[i]
13     }
14     return string(r)
15 }
```

Untuk mengkompilasi:

```
1  $ go build github.com/bdp/stringutil
2  $
```

Jika tidak ada kesalahan, maka akan langsung kembali ke prompt shell.

Membuat Aplikasi yang Memanfaatkan Pustaka

Sama halnya dengan pustaka, aplikasi juga menggunakan pola penamaan yang sama.

```
1  /*
2      src/github.com/bdp/hello/hello.go
3      hello.go
4      diambil dari https://golang.org/doc/code.html
```

```
5 */
6 package main
7
8 import (
9     "fmt"
10
11     "github.com/bdpd/stringutil"
12 )
13
14 func main() {
15     fmt.Printf(stringutil.Reverse("!oG ,olleH"))
16 }
```

Untuk mengkompilasi:

```
1 $ go install github.com/bdpd/hello
2 $ hello
3 Hello, Go!
4 $ ls bin/
5 total 1612
6 drwxr-xr-x 1 bdpd users      10 Sep 12 23:16 .
7 drwxr-xr-x 1 bdpd users     30 Sep 12 23:16 ..
8 -rwxr-xr-x 1 bdpd users 1650409 Sep 12 23:16 hello
9 $
```

Konstruksi Dasar Bahasa Pemrograman Go

Komentar

Bagian komentar dimaksudkan untuk dokumentasi dari *source code*. Ada dua cara untuk memberikan komentar: * Menggunakan `/* ... */` untuk komentar yang meliputi lebih dari satu baris * Menggunakan `//` di awal baris untuk komentar yang meliputi satu baris saja. Komentar ini sejak awal sebaiknya sudah dibiasakan harus ada karena Go menyediakan fasilitas `godoc` untuk menghasilkan dokumentasi dari *source code*. Bagian yang sebaiknya diberikan komentar / dokumentasi adalah bagian diatas `package` dan di atas setiap definisi fungsi (lihat contoh dari `stringutil` di atas).

Tipe Data Angka / Numerik

Untuk tipe numerik, pada dasarnya kita bisa menggunakan bilangan bulat (*integer*) dan bilangan pecahan (*floating-point*). Bilangan bulat terdiri atas bilangan bertanda (*signed* - int) dan bilangan tak-

bertanda (*unsigned* - uint). Berikut ini adalah daftar lengkap dari tipe data numerik tersebut:

Tipe	Arti	Jangkauan
uint8	unsigned 8-bit integer	0 sampai 255
uint16	unsigned 16-bit integer	0 sampai 65535
uint32	unsigned 32-bit integer	0 sampai 4294967295
uint64	unsigned 64-bit integer	0 sampai 18446744073709551615
int8	signed 8-bit integer	-128 sampai 127
int16	signed 16-bit integer	-32768 sampai 32767
int32	signed 32-bit integer	-2147483648 sampai 2147483647
int64	signed 64-bit integer	-9223372036854775808 sampai 9223372036854775807
float32	IEEE-754 32-bit floating-point	
float64	IEEE-754 64-bit floating-point	
complex64	bilangan kompleks dengan float32 riil dan imajiner	~
complex128	bilangan kompleks dengan float64 riil dan imajiner	~
byte	alias dari uint8	

Tipe	Arti	Jangkauan
rune	alias dari int32	

Selain definisi di atas, Go juga mempunyai alias penyebutan yang implementasinya tergantung pada arsitektur komputer yang digunakan:

Tipe	Arti
uint	arsitektur 32 atau 64 bit
int	mempunyai ukuran yang sama dengan uint
uintptr	bilangan bulat tak bertanda untuk menyimpan nilai pointer

String

String digunakan untuk men

```

1 package main
2
3 import (
4     "fmt"
5     "reflect"
6     s "strings"
7 )
8
9 // Definisi string
10 var str1 string = "STMIK AKAKOM"
11 var str2 = "Yogyakarta"
12 var str3 = "stmik akakom"
13
14 func main() {
15
16     // Lihat https://golang.org/pkg/strings/
17     fmt.Println(str1)
18     fmt.Println(len(str1))
19     fmt.Println(s.Contains(str1, "AKAKOM"))
20     fmt.Println(s.Title(str3))
21     fmt.Println(str1[2])
22     fmt.Println(s.Join([]string{str1, str2}, " "))

```

```
23     fmt.Println(reflect.TypeOf(str1))
24     fmt.Println(reflect.TypeOf(str2))
25     fmt.Println()
26
27 }
```

Hasil:

```
1 $ go run contoh-string.go
2 STMIK AKAKOM
3 12
4 true
5 Stmik Akakom
6 77
7 STMIK AKAKOM Yogyakarta
8 string
9 string
```

Boolean

Tipe data Boolean berisi nilai benar (**true**) atau salah (**false**).

```
1 package main
2
3 import "fmt"
4
5 var hasilPerbandingan bool
6 var angka1 uint8 = 21
7 var angka2 uint8 = 17
8
9 func main() {
10     hasilPerbandingan = angka1 < angka2
11     fmt.Println(hasilPerbandingan)
12 }
```

Hasil:

```
1 $ go run boolean.go
2 false
```

Variabel


```
1 // nilai-default-variabel.go
2 package main
3
4 import "fmt"
5
6 func main() {
7
8     // unsigned-integer
9     var defUint8 uint8
10    var defUint16 uint16
11    var defUint32 uint32
12    var defUint64 uint64
13    var defUint uint
14
15    // signed-integer
16    var defInt8 int8
17    var defInt16 int16
18    var defInt32 int32
19    var defInt64 int64
20    var defInt int
21
22    // string
23    var defString string
24
25    // floating-point
26    var defFloat32 float32
27    var defFloat64 float64
28
29    // complex
30    var defComplex64 complex64
31    var defComplex128 complex128
32
33    // alias
34    var defByte byte
35    var defRune rune
36
37    fmt.Println("\nNilai default untuk uint8 = ", defUint8)
38    fmt.Println("Nilai default untuk uint16 = ", defUint16)
39    fmt.Println("Nilai default untuk uint32 = ", defUint32)
40    fmt.Println("Nilai default untuk uint64 = ", defUint64)
41    fmt.Println("Nilai default untuk uint = ", defUint)
42
```

```
43     fmt.Println("\nNilai default untuk int8 = ", defInt8)
44     fmt.Println("Nilai default untuk int16 = ", defInt16)
45     fmt.Println("Nilai default untuk int32 = ", defInt32)
46     fmt.Println("Nilai default untuk int63 = ", defInt64)
47     fmt.Println("Nilai default untuk int = ", defInt)
48
49     fmt.Println("\nNilai default untuk string = ", defString)
50
51     fmt.Println("\nNilai default untuk float32 = ", defFloat32)
52     fmt.Println("Nilai default untuk float64 = ", defFloat64)
53
54     fmt.Println("\nNilai default untuk complex64 = ", defComplex64)
55     fmt.Println("Nilai default untuk complex128 = ", defComplex128)
56
57     fmt.Println("\nNilai default untuk byte = ", defByte)
58     fmt.Println("Nilai default untuk rune = ", defRune)
59
60 }
```

Hasil eksekusi:

```
1  $ go run nilai-default-variabel.go
2
3  Nilai default untuk uint8 =  0
4  Nilai default untuk uint16 = 0
5  Nilai default untuk uint32 = 0
6  Nilai default untuk uint64 = 0
7  Nilai default untuk uint =  0
8
9  Nilai default untuk int8 =  0
10 Nilai default untuk int16 = 0
11 Nilai default untuk int32 = 0
12 Nilai default untuk int63 = 0
13 Nilai default untuk int =  0
14
15 Nilai default untuk string =
16
17 Nilai default untuk float32 = 0
18 Nilai default untuk float64 = 0
19
20 Nilai default untuk complex64 = (0+0i)
21 Nilai default untuk complex128 = (0+0i)
22
```

```
23 Nilai default untuk byte = 0
24 Nilai default untuk rune = 0
25 $
```

Konstanta

Konstanta dimaksudkan untuk menampung data yang tidak akan berubah-ubah. Konstanta dideklarasikan menggunakan kata kunci *const*. Konstanta bisa bertipe *character*, *string*, *boolean*, atau numerik.

Pointer

Konsep *pointer* sebenarnya sudah ada pada bahasa pemrograman lain, khususnya C/C++ (dengan kompleksitas yang lebih tinggi). Suatu *pointer* merupakan suatu nilai yang menunjuk ke suatu nilai lain.

Struktur Kendali

Perulangan dengan **for**

Seleksi Kondisi

Pernyataan **if**

Pernyataan **switch**

Defer

Defer digunakan untuk mengeksekusi suatu perintah sebelum suatu fungsi berakhir. Jika berada pada suatu fungsi, baris kode sumber yang di-defer akan dikerjakan sebelum menemui akhir (*return*). Kegunaan utama dari *defer* ini adalah untuk keperluan pembersihan (*cleanup*). Saat kita membuat kode sumber Go, sering kali dalam setiap operasi terdapat beberapa hal yang harus kita akhiri dengan kondisi tertentu, misalnya jika kita membuka file maka kita harus menutup file jika kita sudah selesai melakukan operasi dengan file tersebut. *Defer* mempermudah kita untuk memastikan bahwa pekerjaan-pekerjaan pembersihan tersebut selalu bisa dilakukan.

5 Fungsi / Function

Fungsi merupakan bagian dari kode sumber yang dimaksudkan untuk mengerjakan sesuatu hal. Hal yang dikerjakan tersebut biasanya merupakan suatu hal yang sifatnya cukup umum sehingga terdapat kemungkinan dalam kode sumber bisa digunakan berkali-kali. Fungsi dibuat supaya tidak perlu mengkode ulang pekerjaan tersebut. Jika diperlukan pada suatu kode, bagian tersebut tinggal memanggil fungsi. Untuk mengerjakan pekerjaan tersebut, fungsi biasanya memerlukan data masukan (sering disebut dengan *argumen* atau *parameter*). Setelah mengerjakan fungsi tersebut, fungsi biasanya menghasilkan suatu nilai (sering disebut dengan istilah *return value* / nilai kembalian). Kode sumber Go yang dimaksudkan untuk menghasilkan *binary executable* mempunyai satu fungsi yang akan dikerjakan saat kode tersebut dikompilasi dan dieksekusi, yaitu fungsi `main()` (lihat bab 2).

6 Penanganan Kesalahan

Penggunaan Kode Error

Panic dan Recover

Go menyediakan konstruksi *panic* dan *recover* untuk menangani kesalahan yang tidak bisa “ditolerir”. Sebagai contoh, jika aplikasi kita mutlak memerlukan suatu file konfigurasi dan file konfigurasi tersebut tidak ada, maka kita bisa menggunakan *panic* untuk menghentikan eksekusi aplikasi dan kemudian memberikan pesan kesalahan. Jika kita hanya menggunakan *panic*, maka program akan berakhir dengan pesan error serta dump dari instruksi biner yang menyebabkan error tersebut. Hal ini seringkali tidak dikehendaki sehingga diperlukan *recover* untuk mengakhiri program dengan baik.

7 Struktur Data Lanjut

Arrays

Slices

Maps

Struct

8 Method

9 Testing

10 Konkurensi