

MODUL PRAKTIKUM

ALGORITMA PEMROGRAMAN

S1 INFORMATIKA



LEMBAR PENGESAHAN

Saya yang bertanda tangan di bawah ini:

Nama : Prasti Eko Yunanto, S.T.,M.Kom.

NIK : 19890017

Koordinator Mata Kuliah : Algoritma Pemrograman

Prodi : S1 Informatika

KK : Intelligent System

Menerangkan dengan sesungguhnya bahwa modul ini digunakan untuk pelaksanaan praktikum di Semester Genap Tahun Ajaran 2020/2021 di Laboratorium Informatika Fakultas Informatika Universitas Telkom.



Bandung, 12 Februari 2021



Mengesahkan,

Koordinator Mata Kuliah

Algoritma Pemrograman

Mengetahui,

Kaprodi S1 Informatika


Prasti Eko Yunanto, S.T.,M.Kom.


Niken Dwi Wahyu Cahyani, Ph.D.

Peraturan Praktikum Laboratorium Informatika 2020 / 2021

1. Praktikum diampu oleh dosen kelas dan dibantu oleh asisten laboratorium dan asisten praktikum.
2. Praktikum dilaksanakan di Gedung Kultubai Selatan (IFLAB 1 s/d IFLAB 5) dan Gedung Kultubai Utara (IFLAB 6 s/d IFLAB 7) sesuai jadwal yang ditentukan.
3. Praktikan wajib membawa modul praktikum, kartu praktikum, dan alat tulis.
4. Praktikan wajib mengisi daftar hadir dan BAP praktikum sesuai dengan ketentuan yang berlaku.
5. Durasi kegiatan praktikum S-1 adalah 2 jam perkuliahan atau 100 menit.
6. Praktikan wajib hadir minimal 75% dari seluruh pertemuan praktikum di lab. Jika total kehadiran kurang dari 75% maka nilai praktikum dan/atau tugas besar dianggap nol.
7. Praktikan yang datang terlambat lebih dari 30 menit tidak diperkenankan mengikuti kegiatan praktikum, kecuali secara jelas diijinkan oleh penanggung jawab praktikum saat itu.
8. Saat praktikum berlangsung, asisten praktikum dan praktikan:
 - Wajib berpakaian rapih sesuai dengan ketentuan institusi.
 - Wajib mematikan/ mengkondisikan semua alat komunikasi dan/atau menyimpannya di *locker*.
 - Dilarang membuka aplikasi yang tidak berhubungan dengan praktikum yang berlangsung.
 - Dilarang mengubah pengaturan *software* maupun *hardware* komputer tanpa ijin.
 - Dilarang membawa makanan maupun minuman di ruang praktikum.
 - Dilarang menyebarkan soal praktikum kepada diluar peserta praktikum saat itu.
 - Dilarang memberikan jawaban ke praktikan lain.
 - Dilarang membuang sampah di ruangan praktikum.
 - Wajib meletakkan alas kaki dengan rapi pada tempat yang telah disediakan.
9. Praktikan dapat mengikuti praktikum susulan maksimal dua modul per satu mata kuliah praktikum.
 - Praktikan yang dapat mengikuti praktikum susulan hanyalah praktikan yang memenuhi syarat sesuai ketentuan institusi, yaitu: sakit dengan surat keterangan medis, tugas institusi yang dibuktikan dengan surat dinas atau dispensasi, atau mendapat musibah atau kedukaan lainnya dengan menunjukkan surat keterangan dari orangtua/wali mahasiswa.
 - Persyaratan untuk praktikum susulan diserahkan sesegera mungkin kepada asisten laboratorium untuk keperluan administrasi.
 - Praktikan yang diijinkan menjadi peserta praktikum susulan ditetapkan oleh Asman Lab dan Bengkel Informatika dan penetapan tidak dapat diganggu gugat.
10. Pelanggaran terhadap peraturan praktikum akan ditindak secara tegas sesuai dengan tingkat pelanggaran yang dilakukan.

Penyampaian Keluhan Praktikum IFLab Melalui IGracias

1. Login IGracias
2. Pilih Menu **Masukan dan Komplain**, pilih **Input Tiket**



3. Pilih Fakultas/Bagian: **Bidang Akademik (FIF)**
4. Pilih Program Studi/Urusan: **Urusan Laboratorium/Bengkel/Studio (FIF)**
5. Pilih Layanan: **Praktikum**
6. Pilih Kategori: **Pelaksanaan Praktikum**, lalu pilih **Sub Kategori**.
7. Isi **Deskripsi** sesuai komplain yang ingin disampaikan.

The screenshot shows a form titled "Input Keluhan". The fields filled in are:

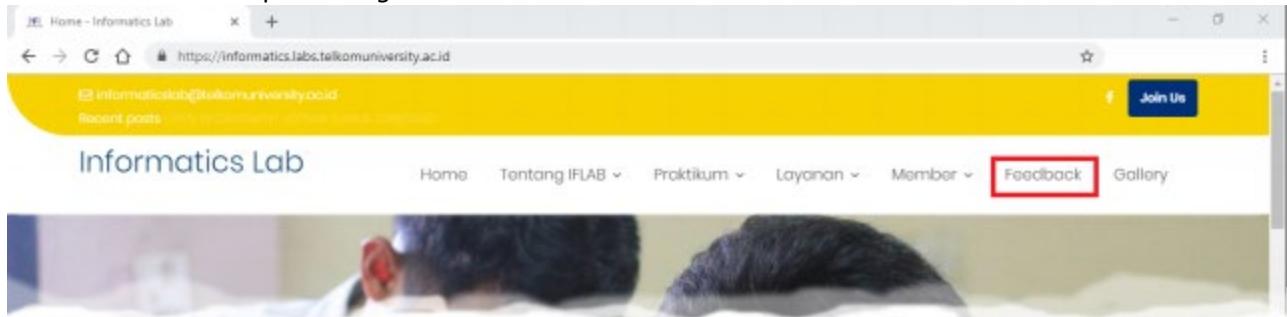
- Fakultas / Bagian : BIDANG AKADEMIK (FIF)
- Program Studi / Urusan : URUSAN LABORATORIUM/BENGKEL/STUDIO (FIF)
- Pelapor : RIZQILLAH ZAHRA LESTARI
- Layanan : PRAKTIKUM
- Kategori : Pelaksanaan Praktikum
- Sub Kategori : Please Select...
- Tipe Masukan : Komplain Masukan

Below the form is a rich text editor toolbar with various icons for bold, italic, underline, font size, alignment, and other document functions.

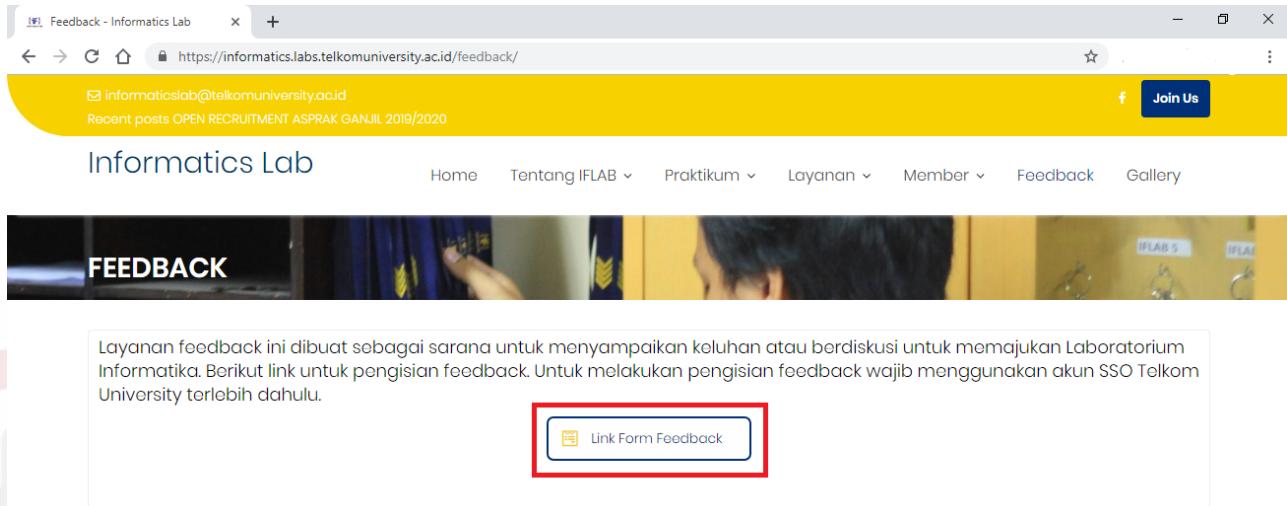
8. Lampirkan file jika perlu. Lalu klik Kirim.

Penyampaian Keluhan Praktikum IFLAB Melalui Situs Web

1. Gunakan browser untuk membuka situs <https://informatics.labs.telkomuniversity.ac.id/>
2. Pilih menu **Feedback** pada *navigation bar website*.



3. Pilih tombol **Link Form Feedback**.



4. Lakukan *login* menggunakan akun **SSO Telkom University** untuk mengakses *form feedback*.
5. Isi *form* sesuai dengan *feedback* yang ingin diberikan.

Aturan Penulisan dan Submisi Tugas

Penulisan Program Tugas

1. Setiap program harus disimpan dalam file dengan ekstensi/akhiran .go.
2. Program harus dibuat rapi dan terstruktur
 - a) Buat indentasi yang benar;
 - Instruksi pada level yang sama dimulai dari kolom yang sama
 - Instruksi di dalam suatu blok, misal didalam struktur while-loop, harus masuk 4 spasi kedalam dibandingkan dengan struktur kontrolnya.
 - Gunakan perintah **go fmt** sebelum submisi untuk memastikan program anda mempunyai struktur yang konsisten. Pada contoh dibawah spasi digambarkan dengan simbol | agar lebih jelas.
 - Gunakan perintah **go clean** sebelum submisi untuk membuang semua file yang tidak diperlukan didalam folder, seperti executable code, sehingga hanya menyisakan file program sumbernya/source code saja.

```
i := 1
for i < 100 {
    if i%10 == 0 {
        fmt.Println( i*3 )
    }
    i = i + 1
}
fmt.Println( "Selesai" )
```

- b) Setiap baris hanya boleh berisi satu instruksi saja, walaupun instruksi tersebut sangat pendek.

3. Pemilihan nama variabel harus sesuai dengan kebutuhan
 - a) **JANGAN** menggunakan satu huruf (**a, b, c, ...**) dari alfabet untuk nama variabel bagi semua kebutuhan
 - b) **JANGAN** menggunakan kata kunci sebagai bagian dari nama variabel, seperti: **_true, _false, _for, forloop, ...**
 - c) Pilih nama variabel yang berhubungan dengan nilai yang disimpan dalam variabel tersebut, seperti: **jumlah, rerata, terbesar, ganjil, ketemu, mhs, skor, min, max, ...**
 - d) Gunakan variabel satu huruf atau akronim untuk manfaat variabel yang sudah umum untuk huruf/akronim tersebut, seperti:
 - **i, j, k, ...** untuk indeks, iterator loop, dsb.
 - **t, t1, temp, ...** untuk variabel temporer
 - **p, q, r, ...** untuk pointer
 - **n, m, ...** untuk jumlah data, ukuran array, dsb.
 - **u, v, w, x, y, z, ...** untuk nilai dari suatu koleksi data
 - e) Gunakan penulisan nama variabel yang konsisten.

- **CamelCase**: pada nama berbentuk kata majemuk, setiap kata dimulai dengan huruf besar, kecuali kata pertama. Kata pertama (atau hanya satu kata) maka tetap dimulai dengan huruf kecil. Dalam bahasa Go, nama dimulai dengan huruf besar merupakan nama yang dieksport oleh paket.
Contoh: **nilaiTerkecil**, **mhsBaru**, **mhsLama**, ...
- **Dengan_underscore**: pada nama berbentuk kata majemuk, kata-kata digabungkan dengan dengan “_”.
Contoh **nilai_terkecil**, **mhs_baru**, **mhs_lama**, ...
- **HURUF BESAR/KAPITAL** digunakan untuk konstanta simbolik, yaitu nilainya tidak pernah, tidak mungkin, dan tidak boleh diubah dalam program.
Contoh: **MAX_MHS**, **NMAX**, ...
- Pada tugas pemrograman DAP **HINDARI** pemilihan nama yang dimulai dengan underscore.
Contoh yang dilarang: **_mhs**, **_temp**, ...

4. Setiap algoritma dan program sumbernya selalu berlaku prinsip satu pintu masuk satu pintu keluar.
 - a) Sekuens instruksi dimulai dari instruksi pertama, eksekusi secara berurutan, dan berakhir pada satu instruksi terakhir.
 - b) Bentuk pengulangan dimulai dari instruksi pertama, masuk ke dalam dan mengulang iterasi badan loop, dan kemudian keluar dari **SATU** lokasi saja, yaitu dari kondisi iterasi tersebut.
 - Jika bentuk struktur **while-loop**, maka loop berakhir dan pintu keluar dari instruksi **while** (bukan dari instruksi **endwhile**).
 - Jika bentuk struktur **repeat-until**, maka loop berakhir dan pintu keluar dari instruksi **until** (bukan dari instruksi **repeat**).
 - Bentuk lain juga menggunakan prinsip yang sama, satu pintu keluar saja, sehingga tidak diperkenankan bentuk **while** dicampur dengan adanya instruksi **break**, atau menggunakan beberapa perintah **break** dalam satu loop, atau menggunakan perintah **continue** didalam suatu loop.
 - c) Bentuk percabangan dimulai dari instruksi pertama, kemudian mengeksekusi blok instruksi sesuai dengan kondisinya, dan berakhir pada instruksi penutup (**endif** atau **endcase**).
 - Tidak diperkenankan merenteng beberapa blok instruksi, misalkan menggunakan instruksi **fallthrough**.
 - Tidak diperkenankan keluar dari tengah blok **if** atau **switch** menggunakan perintah **break**, kecuali untuk kebutuhan keluar dari loop yang memenuhi syarat bentuk pengulangan diatas.
5. Kelas ini mempelajari konsep pemrograman konvensional. Dalam tugas yang diberikan, tidak diperlukan instruksi yang bersifat konkurensi dan/atau paralel. Ini termasuk untuk **JANGAN** menggunakan instruksi assignment paralel
6. Algoritma dan program sumbernya harus jelas dan menggunakan tipe data yang sesuai peruntukannya. Beberapa **contoh penyalahgunaan tipe data**:

- a) Tipe **integer**, tetapi nilai yang dipakai hanya 0 dan 1 (atau sejenisnya), dimana manfaatnya adalah untuk menyatakan suatu keadaan (ada/tidak ada, ketemu/tidak ketemu), dlsb. Tipe yang tepat yang sebaiknya digunakan sudah jelas adalah tipe **Boolean**.
- b) Tipe **real**, tetapi untuk semua operasi yang diterapkan pada variabel tersebut tidak pernah berkaitan dengan pecahan. Tipe yang sebaiknya digunakan adalah tipe **integer**.
7. Algoritma dan program sumber implementasinya harus selalu bersifat efektif, tidak menggunakan instruksi yang pada dasarnya tidak memberikan efek neto yang penting. Beberapa contoh bentuk ekspresi yang berlebihan:

Bentuk yang TIDAK TEPAT	Bentuk yang SEHARUSNYA
<code>if found == true then ..</code> <code>if found != false then ..</code>	<code>if found then ..</code>
<code>if found == false then ..</code> <code>if found != true then ..</code>	<code>if not found then ..</code>
<code>if suatu_kondisi then</code> <code>abc ← true</code> <code>else</code> <code>abc ← false</code> <code>endif</code>	<code>abc ← suatu_kondisi</code>
<code>abc ← def * 1</code>	<code>abc ← def</code>
<code>abc ← def - 0</code>	<code>abc ← def</code>

8. Proses debugging adalah bagian dari kegiatan untuk membuat program sumber yang berjalan dengan baik dan sesuai dengan algoritma yang dirancang. Dalam proses debugging mungkin ditambahkan perintah yang input dan output untuk mencari kesalahan yang ada dalam program.
- Perintah seperti **fmt.Scanln()**, misalnya sebagai instruksi terakhir dalam program, digunakan untuk menghentikan eksekusi agar keluaran dapat diverifikasi.
 - Perintah **fmt.Println("...")** digunakan diberbagai lokasi program untuk mencetak isi variabel dan/atau ekspresi tertentu untuk memastikan kebenaran proses yang sedang terjadi.
 - Akan tetapi pada saat submisi**, semua perintah tambahan tersebut harus dihapus dari program yang akan diserahkan!
 - Pastikan semua perintah **fmt.Print*** sudah memberikan **keluaran yang sesuai dalam format, bentuk dan hasil seperti yang diminta** dalam tugas.
 - Pastikan semua perintah **fmt.Scan*** memang membaca yang harus dibaca, tidak kurang dan tidak lebih.
9. Patuhi etika pendidikan

Jika diperintahkan sebagai tugas mandiri, maka selayaknya untuk **tidak mengutip**, sebagian atau pun seluruhnya, apa adanya atau dengan modifikasi, hasil karya orang lain, rekan lain, atau tim lain.

DAFTAR ISI

LEMBAR PENGESAHAN.....	I
PERATURAN PRAKTIKUM LABORATORIUM INFORMATIKA 2020 / 2021.....	II
PENYAMPAIAN KELUHAN PRAKTIKUM IFLAB MELALUI IGRACIAS	III
ATURAN PENULISAN DAN SUBMISI TUGAS	V
PENULISAN PROGRAM TUGAS	V
DAFTAR ISI	VIII
DAFTAR GAMBAR.....	X
MODUL 1 BAHASA PEMROGRAMAN GO	1
1.1. STRUKTUR PROGRAM Go.....	1
1.2. KODING, KOMPILASI, DAN EKSEKUSI Go	2
1.3. INFORMASI LANJUTAN	3
1.4. LATIHAN	3
MODUL 2 TIPE DATA DAN INSTRUKSI DASAR	4
2.1. DATA DAN VARIABEL.....	4
2.2. INSTRUKSI DASAR	7
2.3. LATIHAN	7
MODUL 3 STRUKTUR KONTROL PERULANGAN	10
3.1. BENTUK WHILE-LOOP	10
3.2. BENTUK REPEAT-UNTIL	11
3.3. LATIHAN	12
MODUL 4 STRUKTUR KONTROL PERCABANGAN.....	15
4.1. BENTUK IF-ELSE	15
4.2. BENTUK SWITCH-CASE	15
4.3. LATIHAN	17
MODUL 5 LATIHAN 1 : TIPE DATA DAN STRUKTUR KONTROL	20
MODUL 6 SUBPROGRAM.....	28
6.1. PROSEDUR	28
6.2. FUNGSI.....	29
6.3. PARAMETER DAN ARGUMEN.....	30
6.4. RUANG LINGKUP VARIABEL	31
6.5. LATIHAN	32
MODUL 7 LATIHAN 2 : SUBPROGRAM	34
MODUL 8 LATIHAN 3 : REVIEW MATERI.....	38
MODUL 9 ARRAY DAN TIPE BENTUKAN.....	42
9.1. RECORD ATAU STRUCTURE	42
9.2. ARRAY.....	44
9.3. KONSTANTA SIMBOLIK.....	46
9.4. TIPE BENTUKAN (TYPE).....	46
9.5. PROSES DASAR ARRAY.....	46
9.6. LATIHAN	48

MODUL 10	
PENCARIAN	50
10.1. PENCARIAN	50
10.2. PENCARIAN SEKUENSIAL.....	50
10.3. PENCARIAN BINER/BELAH TENGAH.....	51
MODUL 11 LATIHAN 4 : ARRAY DAN PENCARIAN.....	53
MODUL 12 PENGURUTAN	64
12.1. METODA PENGURUTAN SELEKSI	64
12.2. METODA PENGURUTAN INSERSI	64
12.3. METODA PENGURUTAN DENGAN DIHITUNG.....	65
12.4. TEKNIK PERINGKAT	66
12.5. LATIHAN	66
MODUL 13 LATIHAN 5 : PENGURUTAN.....	69
MODUL 14 SKEMA PEMROSESAN SEKUENSIAL (PENGAYAAN)	71
14.1. PEMBACAAN DATA TANPA MARKER PADA AKHIR RANGKAIAN DATA	71
14.2. PEMBACAAN DATA DENGAN MARKER PADA AKHIR RANGKAIAN DATA.....	71
14.3. KEMUNGKINAN RANGKAIAN DATA KOSONG, HANYA ADA MARKER	72
14.4. ELEMEN PERTAMA PERLU DIPROSES TERSENDIRI / KASUS KHUSUS	72
14.5. LATIHAN	73
MODUL 15 SUBPROGRAM LANJUTAN (PENGAYAAN)	76
15.1. REKURSIF	76
15.2. INDUKSI MATEMATIKA.....	77
15.3. REKURSIF DAN PEMROGRAMAN	77
15.4. LATIHAN	78
MODUL 16 MESIN ABSTRAK (PENGAYAAN)	79
16.1. CONTOH KASUS MESIN DOMINO	79
16.2. LATIHAN	80
DAFTAR PUSTAKA	82
LAMPIRAN.....	83
A. PAKET-PAKET PENTING	83
B. PERINTAH Go	96
C. INSTALASI GO PERKAKAS	96
D. INSTALASI PAKET PIHAK KE-3 (CONTOH PAKET GUI).....	98
E. CONTOH TUBES: PERMAINAN SOLITAIRE GAPLEH	99

Daftar Gambar

GAMBAR - 6. 1.....	73
GAMBAR - 6. 2.....	74
GAMBAR - 6. 3.....	99



Modul 1 Bahasa Pemrograman Go

1.1. Struktur Program Go

Dalam kerangka program yang ditulis dalam bahasa pemrograman Go, program utama selalu mempunyai dua komponen berikut:

- **package main** merupakan penanda bahwa file ini berisi program utama.
- **func main()** berisi kode utama dari sebuah program Go.

Komentar, bukan bagian dari kode program, dan dapat ditulis dimana saja didalam program:

- Satu baris teks yang diawali dengan garis miring ganda ('//') s.d. akhir baris, atau
- Beberapa baris teks yang dimulai dengan pasangan karakter '/*' dan diakhiri dengan '*/'

```
// Setiap program utama dimulai dengan "package main"
package main

// Impor paket yang dibutuhkan, "fmt" berisi proses I/O standar
import "fmt"

// Kode program utama dalam "fungsi main"
func main() {
    ..
}
```

Contoh sebuah program dalam bahasa pemrograman Go.

```
package main
import "fmt"
func main() {
    var greetings = "Selamat datang di dunia DAP"
    var a, b int

    fmt.Println(greetings)
    fmt.Scanln(&a, &b)
    fmt.Printf("%v + %v = %v\n", a, b, a+b)
}
```

```
C:\users\go\src\hello>dir
Directory of C:\users\jimmyt\go\src\hello
6/29/2019 7:15 PM 1,727 hello.go
C:\users\go\src\hello>go build hello.go
C:\users\go\src\hello>dir
Directory of C:\users\jimmyt\go\src\hello
6/29/2019 7:15 PM 1,727 hello.go
6/29/2019 7:18 PM 2,198,528 hello.exe
C:\users\go\src\hello>hello
Selamat datang di dunia DAP
7 5
7 + 5 = 12
C:\users\go\src\hello>
```

1.2. Koding, Kompilasi, dan Eksekusi Go

Koding

- Tidak berbeda dengan penulisan program sumber dalam bahasa lain, program Go harus dibuat menggunakan penyunting teks dan disimpan dalam format teks, bukan dalam format dokumen (doc, docx, atau lainnya).
- Setiap program go disimpan dalam file teks dengan ekstensi **.go**, dengan nama bebas. Sebaiknya nama file adalah nama untuk program tersebut.
- Setiap satu program lengkap Go disimpan dalam satu folder tersendiri. Nama folder merupakan nama program tersebut. Karena itu secara prinsip, satu program Go dapat dipecah dalam beberapa file dengan ekstensi **.go** selama disimpan dalam folder yang sama.

Kompilasi

Beberapa bahasa pemrograman dirancang untuk diimplementasikan sebagai interpreter dan lainnya sebagai kompilator. **Interpreter** akan membaca setiap baris instruksi dan kemudian langsung mengeksekusinya, dengan hanya sedikit pemeriksaan apakah penulisan keseluruhan program sudah benar atau belum. **Kompilator** akan memeriksa keseluruhan program sumber dan kemudian mengubahnya menjadi program eksekutabel, sehingga konsistensi penulisan (seperti penggunaan tipe data) sudah diperiksa sebelum eksekusi. Selain itu karena program dibuat menjadi eksekutabel lebih dahulu, proses optimasi dapat dilakukan sehingga program menjadi sangat efisien.

Go diimplementasikan sebagai kompilator. Berikut adalah contoh sesi yang biasa dilakukan saat mengkompilasi dan mengeksekusi program dalam bahasa Go:

- Panggil shell atau terminal (program/utiliti **cmd.exe** di Windows)
- Masuk kedalam (**cd**) folder program (normalnya ada di **C:\Users\go\src** atau yang sejenis)
- Kemudian panggil perintah **go build** atau **go build file.go** untuk mengkompilasi file.go
- Jika gagal, akan muncul pesan error yang sesuai, pelajari dengan baik pesan tersebut, perbaiki teks program sumber, kemudian ulangi proses **build**-nya.
- Jika berhasil maka pada folder tersebut akan dibuat program dengan nama yang sama dan diakhiri dengan **.exe** (untuk Windows)
- Panggil program eksekutabel tersebut dari terminal yang sama. **Jangan** memanggil program tersebut dengan meng-klik eksekutabel tersebut dari folder karena program kalian hanya berbasis teks, bukan/belum dirancang dengan tampilan windows.

Catatan

Semua proses terkait bahasa Go dilakukan melalui utilitas go. Beberapa opsi dengan utilitas go:

- **go build**: mengkompilasi program sumber yang ada dalam folder menjadi sebuah program.
- **go build file.go**: mengkompilasi program sumber file.go saja.
- **go fmt**: membaca semua program sumber dalam folder dan mereformat penulisannya agar sesuai dengan standar penulisan program sumber Go.
- **go clean**: membersihkan file-file dalam folder sehingga tersisa program sumber nya saja.

1.3. Informasi Lanjutan

Lihat rujukan dalam daftar pustaka untuk lebih memahami pemrograman Go.

1.4. Latihan

1. Selidiki bahasa-bahasa pemrograman berikut, apakah termasuk diinterpretasi, dikompilasi, dikompilasi (ke instruksi perantara) kemudian diinterpretasi:
 - a. Pascal
 - b. C dan C++
 - c. Java
 - d. Python
2. Instal kompilator Go di komputer yang anda gunakan. kemudian salin contoh program diatas ke dalam folder C:\Users\userid\Go\hello\hello.go, yaitu buat folder Go dalam direktori home anda, kemudian buat subfolder hello dan taruh file hello.go didalamnya. Hidupkan terminal (cmd.exe), dan panggil go build di dalam folder hello tersebut. Periksa apakah hello.exe muncul di folder tersebut? Jika ya, coba eksekusi program tersebut, juga melalui terminal cmd.exe tersebut. (Jangan di klik melalui browser folder).



Modul 2 Tipe Data dan Instruksi Dasar

2.1. Data dan Variabel

Variabel adalah **nama** dari suatu **lokasi** di memori dimana **data** dengan **tipe** tertentu dapat disimpan.

1. Nama variabel dimulai dengan huruf dan dapat dikuti dengan sejumlah huruf, angka, atau garisbawah.

Contoh: **ketemu, found, rerata, mhs1, data_2, ...**

2. Tipe data yang umum tersedia adalah integer, real, Boolean, karakter, dan string. Lihat tabel berikut ini untuk variasi tipe data yang disediakan dalam bahasa Go.

Notasi tipe dasar	Tipe dalam Go	Keterangan
integer	int int8 int32 (rune) int64 uint uint8 (byte) uint32 uint64	bergantung platform 8 bit: -128..127 32 bit: -10^9..10^9 64 bit: -10^19..10^19 bergantung platform 0..255 0.4294967295 0..(2^64-1)
real	float32 float64	32bit: -3.4E+38 .. 3.4E+38 64bit: -1.7E+308 .. 1.7E+308
Boolean (atau logikal)	bool	false dan true
karakter	byte (uint8) rune (int32)	tabel ASCII/UTF-8 tabel UTF-16
string	string	

3. **Nilai data** yang tersimpan dalam variabel dapat diperoleh dengan menyebutkan langsung nama variabelnya.

Contoh: Menyebutkan nama **found** akan mengambil nilai tersimpan dalam memori untuk variabel **found**, pastinya.

4. **Informasi alamat** atau lokasi dari variabel dapat diperoleh dengan menambahkan prefiks **&** didepan nama variabel tersebut.

Contoh: **&found** akan mendapatkan alamat memori dimana data untuk **found** disimpan

5. Jika variabel berisi alamat memori, prefiks ***** pada variabel tersebut akan memberikan nilai yang tersimpan dalam memori dimana lokasinya disimpan dalam variabel tersebut.

Contoh: ***mem** akan mendapatkan data di memori yang alamatnya tersimpan di **mem**.

Karenanya ***(&found)** akan mendapatkan data dari lokasi memori dimana variabel **found** berada, alias sama saja dengan menyebutkan langsung **found 8=**

6. Operasi yang dapat dilakukan terhadap tipe data diatas adalah

Operator dalam Go	Tipe data terkait	Keterangan
+	string integer dan real	konkatenasi 2 string operasi penjumlahan
- * /	integer dan real	operasi pengurangan, perkalian, dan pembagian
%	integer	operasi sisa pembagian integer (modulo)
& ^ &^	integer	operasi per-bit AND, OR, XOR, AND-NOT
<< >>	integer dan unsigned integer	operasi geser bit kiri/kanan sebanyak unsigned integer yang diberikan
< <= >= > == !=	selain Boolean	komparasi menghasilkan nilai Boolean komparasi karakter sesuai dengan posisi karakter tersebut dalam tabel ASCII/UTF-16 komparasi string sesuai dengan operasi karakter per karakter, dimulai dari karakter paling kiri (awal)
&& !	Boolean	operasi Boolean AND, OR, dan NOT
* &	variabel apasaja	mendapatkan data dari lokasi memori dan mendapatkan lokasi dari variabel

Contoh:

"non suffi" + "cit mundo" → "non sufficit mundo"

2019.01 + 1.0102 → 2020.0202

2020 / 20 → 101

20.2 * 1.1 → 22.22

2020 % 1999 → 21

2020 & 1111 → 2104

2020 ^ 1111 → 1663

2020 >> 2 → 505

"minutus" < "magnus" → false

2020 >= 1234 → true

! false && true → true

7. Bahasa Go menganut kesesuaian tipe data yang ketat. Tipe data yang berbeda tidak boleh dicampur dalam satu ekspresi, bahkan tipe datanya masih yang sejenis, misalnya masih sama-sama integer (**int** dan **int32**). Untuk menyesuaikan tipe data, ada beberapa cara yang dapat dilakukan:
 - a) Casting, **tipe(data)**, mengubah tipe dari data yang diberikan ke tipe yang diinginkan.
 - b) Memanfaatkan fungsi **Sprint** dan **Sscan** dari paket **fmt**.
 - c) Memanfaatkan fungsi-fungsi dalam paket **strconv**, seperti **Atoi**, **Itoa**, dan **ParseBool**. Lihat lampiran untuk contoh penggunaan.

Contoh:

`2020.0 % 19 → will be an illegal expression error`

`int(2020.0) % 19 → 6`

Konversi tipe	Data	Tipe baru	Keterangan
<code>tipe(data)</code>	integer	integer	format data tidak berubah, hanya penyesuaian jumlah bit. Kekurangan bit diisi bit 0 disebelah kiri (MSB)
	real	real	format data tidak berubah, hanya penyesuaian jumlah bit. Kekurangan bit, maka bit mantissa diisi bit 0.
	real	integer	format data disesuaikan dg. tipe data tujuan
	integer	real	format data disesuaikan dg. tipe data tujuan
<code>fmt.Sprintf("%v", v)</code>	any type	string	tulis output ke string
<code>fmt.Sprintf("%c", v)</code>	karakter	string	tulis karakter ke string
<code>fmt.Sscanf(s, "%v", &v)</code>	string	any type	baca string ke variabel dengan tipe tertentu
<code>fmt.Sscanf(s, "%c", &v)</code>	string	karakter	baca string ke variabel bertipe karakter

8. Variabel harus dideklarasikan dulu sebelum digunakan. Variabel juga harus diinisialisasi dulu (diisi data) agar nilai yang tersimpan diketahui dengan jelas dan eksekusi algoritma menjadi terprediksi. Dalam bahasa Go, variabel yang tidak diinisialisasi lebih dahulu otomatis diisi dengan nilai default yang ekivalen dengan bit 0.
- Nilai 0 untuk bilangan integer
 - `0.0E+0` untuk bilangan real
 - `false` untuk Boolean
 - Karakter NUL (lihat tabel ASCII) untuk karakter
 - `""` (string kosong, string dengan panjang 0) untuk string
 - `nil` untuk alamat memori

Notasi deklarasi variabel	Penulisan dalam Go	Keterangan
<code>kamus a : tipe</code>	<code>var a tipe</code>	a diinisialisasi dengan nilai default
<code>kamus a : tipe</code>	<code>var a tipe = nilai_awal</code> <code>var a = (tipe)nilai_awal</code>	a diinisialisasi dengan nilai_awal
<code>algoritma a ← nilai_awal</code>	<code>a := nilai_awal</code> <code>a := (tipe)nilai_awal</code>	secara implisit , tipe variabel a ditentukan dari nilai inisialisasinya

Contoh:

```
func main() {
    var a int
    a = 2019
    r := 2019.0707
    b := false
    c := 'x'
    s := "a string is a string"
}
```

2.2. Instruksi Dasar

Notasi instruksi dasar	Penulisan dalam bahasa Go	Keterangan
v1 ← e1 { v1:=e1 }	v1 = e1	operasi assignment, mengisi data ke lokasi memori (variabel)
v1 ← v1 + e1	v1 += e1 // atau v1 = v1 + e1	
v1 ← v1 - e1	v1 -= e1 // atau v1 = v1 - e1	
v1 ← v1 + 1	v1++ // atau v1 = v1 + 1	
v1 ← v1 - 1	v1-- // atau v1 = v1 - 1	
input v1, v2	fmt.Scan(&v1, &v2) fmt.Scanln(&v1, &v2) fmt.Scanf("%v %v", &v1, &v2)	Pembacaan data memerlukan alamat memori kemana data akan disimpan.
output e1, e2	fmt.Print(e1, e2) fmt.Println(e1, e2) fmt.Printf("%v %v\n", e1, e2)	Penulisan data memerlukan nilai data yang akan dituliskan.

Contoh:

```
package main
import "fmt"

func main() {
    var a, b, c float64
    var hipotenusa bool

    fmt.Scanln( &a, &b, &c )
    hipotenusa = (c*c) == (a*a + b*b)
    fmt.Println( "Sisi c adalah hipotenusa segitiga a,b,c: ", hipotenusa )
}
```

2.3. Latihan

1. Telusuri program berikut dengan cara mengkompilasi dan mengeksekusi program. Silakan masukan data yang sesuai sebanyak yang diminta program. Perhatikan keluaran yang diperoleh. Coba terangkan apa sebenarnya yang dilakukan program tersebut?

```

package main
import "fmt"

func main() {
    var (
        satu, dua, tiga string
        temp             string
    )
    fmt.Print("Masukan input string: ")
    fmt.Scanln(&satu)
    fmt.Print("Masukan input string: ")
    fmt.Scanln(&dua)
    fmt.Print("Masukan input string: ")
    fmt.Scanln(&tiga)
    fmt.Println("Output awal = " + satu + " " + dua + " " + tiga)
    temp = satu
    satu = dua
    dua = tiga
    tiga = temp
    fmt.Println("Output akhir = " + satu + " " + dua + " " + tiga)
}

```

2. Tahun kabisat adalah tahun yang habis dibagi 400 atau habis dibagi 4 tetapi tidak habis dibagi 100. Buatlah sebuah program yang menerima input sebuah bilangan bulat dan memeriksa apakah bilangan tersebut merupakan tahun kabisat (**true**) atau bukan (**false**).

(Contoh input/output, **Teks bergaris bawah** adalah input dari user):

Tahun: 2016
Kabisat: true

Tahun: 2000
Kabisat: true

Tahun: 2018
Kabisat: false

3. Buat program **Bola** yang menerima input jari-jari suatu bola (bilangan bulat). Tampilkan Volume dan Luas kulit bola. $volume_{bola} = \frac{4}{3}\pi r^3$ dan $luas_{bola} = 4\pi r^2$. Dimana $\pi \approx 3.1415926535$.

(Contoh input/output, **Teks bergaris bawah** adalah input dari user):

Jejari = 5
Bola dengan jejari 5 memiliki volume 523.5988 dan luas kulit 314.1593

4. Dibaca nilai temperatur dalam derajat Celcius. Nyatakan temperatur tersebut dalam Fahrenheit

$$Celsius = (Fahrenheit - 32) \times \frac{5}{9}$$

$$Reamur = Celsius \times \frac{4}{5}$$

$$Kelvin = (Fahrenheit + 459.67) \times \frac{5}{9}$$

(Contoh input/output, **Teks bergaris bawah** adalah input dari user):

Temperatur Celcius: 50
Derajat Fahrenheit: 122

Lanjutkan program diatas, sehingga temperatur dinyatakan juga dalam derajat Reamur dan Kelvin.

(Contoh input/output, **Teks bergaris bawah** adalah input dari user):

Temperatur Celcius: 50
Derajat Reamur: 40
Derajat Fahrenheit: 122
Derajat Kelvin: 323

5. Tipe karakter sebenarnya hanya apa yang tampak dalam tampilan. Didalamnya tersimpan dalam bentuk biner 8 bit (byte) atau 32 bit (rune) saja.

Buat program ascii yang akan membaca 5 buah data integer dan mencetaknya dalam format karakter. Dan kemudian membaca 3 buah data karakter dan mencetak 3 buah karakter setelah karakter tersebut (menurut tabel ASCII)

Masukan terdiri dari dua baris. Baris pertama berisi 5 buah data integer. Data integer mempunyai nilai antara 32 s.d. 127. Baris kedua berisi 3 buah karakter yang berdampingan satu dengan yang lain (tanpa dipisahkan spasi).

Keluaran juga terdiri dari dua baris. Baris pertama berisi 5 buah representasi karakter dari data yang diberikan, yang berdampingan satu dengan lain, tanpa dipisahkan spasi. Baris kedua berisi 3 buah karakter (juga tidak dipisahkan oleh spasi).

No.	Masukan	Keluaran
1	66 97 103 117 115 SNO	Bagus TOP

Catatan: Gunakan fmt.Scanf("%c", &var) untuk pembacaan satu karakter dan fmt.Printf("%c", var) untuk penulisan satu karakter.

Modul 3 Struktur Kontrol Perulangan

Go hanya mempunyai kata kunci **for** untuk semua jenis pengulangan yang kita pelajari dalam notasi algoritma. Dua bentuk yang kita gunakan disini adalah struktur **while-loop** dan **repeat-until**.

Bentuk pengulangan dalam bahasa Go
<pre>for inisialisasi; kondisi; update { // .. for-loop ala C // .. ke-3 bagian opsional, tetapi ";" tetap harus ada }</pre>
<pre>for kondisi { // .. ulangi kode disini selama kondisi terpenuhi // .. sama seperti "for ; kondisi; {"} }</pre>
<pre>for { // .. tanpa kondisi, berarti loop tanpa henti (perlu if-break) }</pre>
<pre>for ndx, var := range slice/array { // .. iterator mengunjungi seluruh isi slice/array // .. pada setiap iterasi ndx diset indeks dan var diisi nilainya }</pre>

Dalam konsep pemrograman terstruktur, setiap rancangan algoritma harus memenuhi syarat satu pintu masuk dan satu pintu keluar. Karena itu **tidaklah diperkenankan** untuk membuat program yang sumber yang mempunyai struktur loop dimana pintu keluar lebih dari satu, seperti:

- Satu pintu keluar dari kondisi **for** dan satu lagi dari instruksi **if-break**
- Atau mempunyai instruksi **if-break** yang lebih dari satu.

3.1. Bentuk While-Loop

Bentuk **while-loop** memastikan setiap kali memasuki loop, ada kondisi yang harus terpenuhi (benar/**true**). Dan ini juga berarti saat keluar dari loop, maka nilai kondisi tersebut pasti salah/**false**!

Notasi algoritma	Penulisan dalam bahasa Go
<pre>while (kondisi) do .. kode yang diulang endwhile</pre>	<pre>for kondisi { .. kode yang diulang }</pre>

Contoh penggunaan bentuk **while-loop** untuk menghitung $y=\sqrt{x}$:

```
e := 0.0000001
x := 2.0
y := 0.0
y1 := x
for y1-y > e || y1-y < -e {
    y = y1
    y1 = 0.5*y + 0.5*(x/y)
}
fmt.Printf( "sqrt(%v)=%v\n", x, y )
```

3.2. Bentuk Repeat-Until

Bentuk **repeat-until** dimana pengulangan dilakukan terus menerus sampai kondisi keluar terpenuhi. Artinya selama kondisi belum terpenuhi (salah/false) maka pengulangan akan terus dilakukan. Pada saat keluar dari loop maka nilai kondisi pasti benar/true!

Notasi Algoritma	Penulisan dalam bahasa Go
<pre>repeat .. kode yang diulang until (kondisi)</pre>	<pre>for selesai:=false; !selesai; { .. kode yang diulang selesai = kondisi } for selesai:=false; !selesai; selesai=kondisi { .. kode yang diulang }</pre>

Contoh penggunaan bentuk **repeat-until** untuk mencetak deret bilangan Fibonacci:

```
maxF := 100
f0 := 0
f1 := 1
f2 := 1
fmt.Println( "Bilangan fibonacci pertama:", f1 )
for selesai:=false; !selesai; {
    f0 = f1
    f1 = f2
    f2 = f1 + f0
    fmt.Println( "Bilangan fibonacci berikutnya:", f1 )
    selesai = f2 > maxF
}
```

Perhatian: Karena pernyataan kondisi ada dibawah pada bentuk repeat-until, **apapun kondisinya**, badan loop **pasti akan pernah dieksekusi** minimum satu kali!

Kode Go dibawah menggunakan algoritma yang sangat mirip dengan algoritma diatas, dengan perbedaan pada digunkannya bentuk while-loop. Umumnya keluaran kedua algoritma sama, **kecuali** saat **maxF** diinisialisasi dengan nilai 0 atau lebih kecil!

```
maxF := 100
f0 := 0
f1 := 1
f2 := 1
fmt.Println( "Bilangan fibonacci pertama:", f1 )
for f2 <= maxF {
    f0 = f1
    f1 = f2
    f2 = f1 + f0
    fmt.Println( "Bilangan fibonacci berikutnya:", f1 )
}
```

3.3. Latihan

1. Siswa kelas IPA di salah satu sekolah menengah atas di Indonesia sedang mengadakan praktikum kimia. Di setiap percobaan akan menggunakan 4 tabung reaksi, yang mana susunan warna cairan di setiap tabung akan menentukan hasil percobaan. Siswa diminta untuk mencatat hasil percobaan tersebut. Percobaan dikatakan berhasil apabila susunan warna zat cair pada gelas 1 hingga gelas 4 secara berturutan adalah ‘merah’, ‘kuning’, ‘hijau’, dan ‘ungu’ selama 5 kali percobaan berulang.

Buatlah sebuah program yang menerima input berupa warna dari ke 4 gelas reaksi sebanyak 5 kali percobaan. Kemudian program akan menampilkan **true** apabila urutan warna sesuai dengan informasi yang diberikan pada paragraf sebelumnya, dan **false** untuk urutan warna lainnya.

Perhatikan contoh sesi interaksi program seperti dibawah ini (**teks bergaris bawah** adalah input/read):

Percobaan 1: <u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 2: <u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 3: <u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 4: <u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 5: <u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
BERHASIL: true			
Percobaan 1: <u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 2: <u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 3: <u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 4: <u>ungu</u>	<u>kuning</u>	<u>hijau</u>	<u>merah</u>
Percobaan 5: <u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
BERHASIL: false			

2. Suatu pita (string) berisi kumpulan nama-nama bunga yang dipisahkan oleh spasi dan ‘-’, contoh pita diilustrasikan seperti berikut ini.

Pita : mawar – melati – tulip – teratai – kamboja – anggrek

Buatlah sebuah program yang menerima input sebuah bilangan bulat positif (dan tidak nol) N, kemudian program akan meminta input berupa nama bunga secara berulang sebanyak N kali dan nama tersebut disimpan kedalam pita.

(Petunjuk: gunakan operasi penggabungan string dengan operator “+”).

Tampilkan isi pita setelah proses input selesai.

Perhatikan contoh sesi interaksi program seperti dibawah ini (**teks bergaris bawah** adalah input/read):

N : <u>3</u>	N : <u>0</u>
Bunga 1: <u>Kertas</u>	Pita :
Bunga 2: <u>Mawar</u>	
Bunga 3: <u>Tulip</u>	
Pita : Kertas – Mawar – Tulip –	

Modifikasi program sebelumnya dimana proses input akan berhenti apabila user mengetikkan ‘SELESAI’. Kemudian tampilkan isi pita beserta banyaknya bunga yang ada didalam pita

Perhatikan contoh sesi interaksi program seperti dibawah ini (**teks bergaris bawah** adalah input/read):

Bunga 1: <u>Kertas</u> Bunga 2: <u>Mawar</u> Bunga 3: <u>Tulip</u> Bunga 4: <u>SELESAI</u> Pita : Kertas - Mawar - Tulip - Bunga: 3	Bunga 1: <u>SELESAI</u> Pita : Bunga: 0
--	---

3. Setiap hari Pak Andi membawa banyak barang belanjaan dari pasar dengan mengendarai sepeda motor. Barang belanjaan tersebut dibawa dalam kantong terpal dikiri-kanan motor. Sepeda motor tidak akan oleng jika selisih berat barang di kedua kantong sisi tidak lebih dari 9 kg.

Buatlah program PakAndi yang menerima input dua buah bilangan real positif yang menyatakan berat total masing-masing isi kantong terpal. Program akan terus meminta input bilangan tersebut hingga salah satu kantong terpal berisi 9 kg atau lebih.

Perhatikan contoh sesi interaksi program seperti dibawah ini (teks bergaris bawah adalah input/read):

Masukan berat belanjaan di kedua kantong: <u>5.5</u> <u>1.0</u>
Masukan berat belanjaan di kedua kantong: <u>7.1</u> <u>8.5</u>
Masukan berat belanjaan di kedua kantong: <u>2</u> <u>6</u>
Masukan berat belanjaan di kedua kantong: <u>9</u> <u>5.8</u>
Proses selesai.

Modifikasi program tersebut, dimana program akan menampilkan **true** jika selisih kedua isi kantong lebih dari atau sama dengan 9 kg. Program berhenti memproses apabila total berat isi kedua kantong melebihi 150 kg atau salah satu kantong beratnya negatif.

Perhatikan contoh sesi interaksi program seperti dibawah ini (teks bergaris bawah adalah input/read):

Masukan berat belanjaan di kedua kantong: <u>5</u> <u>10</u>
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: <u>55.6</u> <u>70.2</u>
Sepeda motor pak Andi akan oleng: true
Masukan berat belanjaan di kedua kantong: <u>72.3</u> <u>66.9</u>
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: <u>59.5</u> <u>98.7</u>
Proses selesai.

4. Diberikan sebuah persamaan sebagai berikut ini.

$$f(k) = \frac{(4k + 2)^2}{(4k + 1)(4k + 3)}$$

Buatlah sebuah program yang menerima input sebuah bilangan sebagai K, kemudian menghitung dan menampilkan nilai f(K) sesuai persamaan diatas.

Perhatikan contoh sesi interaksi program seperti dibawah ini (teks bergaris bawah adalah input/read):

Nilai K = <u>100</u>
Nilai f(K) = 1.0000061880

$\sqrt{2}$ merupakan bilangan irasional. Meskipun demikian, nilai tersebut dapat dihampiri dengan rumus berikut:

$$\sqrt{2} = \prod_{k=0}^{\infty} \frac{(4k+2)^2}{(4k+1)(4k+3)}$$

Modifikasi program sebelumnya yang menerima input integer K dan menghitung $\sqrt{2}$ untuk K tersebut. Hampiran $\sqrt{2}$ dituliskan dalam ketelitian 10 angka di belakang koma.

Perhatikan contoh sesi interaksi program seperti dibawah ini (**teks bergaris bawah** adalah input/read):

```
#Contoh 1
Nilai K = 10
Nilai akar 2 = 1.4062058441

#Contoh 2
Nilai K = 100
Nilai akar 2 = 1.4133387072

#Contoh 3
Nilai K = 1000
Nilai akar 2 = 1.4141252651
```

Modul 4 Struktur Kontrol Percabangan

Untuk analisa kasus, bahasa Go mendukung dua bentuk percabangan, yaitu **if-else** dan **switch-case**.

4.1. Bentuk If-Else

Berikut ini bentuk-bentuk **if-else** yang mungkin dilakukan dalam bahasa Go. Semua bentuk dibawah merupakan satu instruksi **if-else-endif** saja (hanya satu **endif**). Bentuk **if-else** yang bersarang (dengan beberapa **endif**) dapat dibentuk dengan komposisi beberapa **if-else-endif** tersebut.

Notasi algoritma	Penulisan dalam bahasa Go
<pre>if (kondisi) then .. kode untuk kondisi true endif</pre>	<pre>if kondisi { .. kode untuk kondisi true }</pre>
<pre>if (kondisi) then .. kode untuk kondisi true else .. kode untuk kondisi false endif</pre>	<pre>if kondisi { .. kode untuk kondisi true } else { .. kode untuk kondisi false }</pre>
<pre>if (kondisi-1) then .. kode untuk kondisi-1 true else if (kondisi-2) then .. kode untuk kondisi-2 true .. dst. dst. else .. kode jika semua kondisi .. diatas false endif</pre>	<pre>if kondisi_1 { .. kode untuk kondisi_1 true } else if kondisi_2 { .. kode untuk kondisi_2 true .. dst. dst. } else { .. kode jika semua kondisi .. diatas false }</pre>

Contoh konversi (nilai, tubes, kehadiran) menjadi indeks nilai.

```
if nilai > 75 && adaTubes {
    indeks = 'A'
} else if nilai > 65 {
    indeks = 'B'
} else if nilai > 50 && pctHadir > 0.7 {
    indeks = 'C'
} else {
    indeks = 'F'
}
fmt.Printf( "Nilai %v dengan kehadiran %v% dan buat tubes=%v, mendapat
s %c\n", nilai, pctHadir, adaTubes, indeks )
```

4.2. Bentuk Switch-Case

Dalam bahasa Go ada dua variasi bentuk switch-case. Bentuk yang biasa digunakan adalah ekspresi ditulis pada perintah **switch** dan nilai ditulis dalam setiap label **case**-nya. Bentuk yang kedua mempunyai **switch** tanpa ekspresi, tetapi setiap **case** boleh berisi ekspresi Boolean. Tentunya bentuk yang kedua lebih bersifat umum, dan merupakan penyederhanaan bentuk (atau alias dari) susunan suatu **if-elseif...-else-endif**.

Notasi algoritma	Penulisan dalam bahasa Go
<pre>depend on expresi nilai_1: .. kode jika ekspresi bernilai_1 nilai_2: .. kode jika ekspresi bernilai_2 .. dst. dst. }</pre>	<pre>switch ekspresi { case nilai_1: .. kode jika ekspresi bernilai_1 case nilai_2: .. kode jika ekspresi bernilai_2 .. dst. dst. default: .. kode jika tidak ada nilai .. yang cocok dengan ekspresi }</pre>
<pre>depend on (daftar variabel) kondisi_1: .. kode jika ekspresi_1 true kondisi_2: .. kode jika ekspresi_2 true .. dst. dst. }</pre>	<pre>switch { case kondisi_1: .. kode jika ekspresi_1 true case kondisi_2: .. kode jika ekspresi_2 true .. dst. dst. default: .. jika tidak ada ekspresi .. yang bernilai true }</pre>

Contoh menentukan batas nilai untuk suatu indeks:

```
switch indeks {
case 'A':
    batasA = 100
    batasB = 75
case 'B':
    batasA = 75
    batasB = 65
case 'C':
    batasA = 65
    batasB = 50
default:
    batasA = 50
    batasB = 0
}
fmt.Printf( "Rentang nilai %v adalah: %v..%v\n", indeks, batasB, batasA )

switch {
case nilai > 75 && adaTubes:
    indeks = 'A'
case nilai > 65:
    indeks = 'B'
case nilai > 50 && pctHadir > 0.7:
    indeks = 'C'
default:
    indeks = 'F'
}
fmt.Printf( "Nilai %v dengan kehadiran %v%% dan buat tubes=%v, mendapat
s %c\n", nilai, pctHadir, adaTubes, indeks )
```

4.3. Latihan

- PT POS membutuhkan aplikasi perhitungan biaya kirim berdasarkan berat parcel. Maka, **buatlah program BiayaPos untuk menghitung biaya pengiriman tersebut dengan ketentuan sebagai berikut!**

Dari berat parcel (dalam gram), harus dihitung total berat dalam kg dan sisanya (dalam gram). Biaya jasa pengiriman adalah Rp. 10.000,- per kg. Jika sisa berat tidak kurang dari 500 gram, maka tambahan biaya kirim hanya Rp. 5,- per gram saja.

Tetapi jika kurang dari 500 gram, maka tambahan biaya akan dibebankan sebesar Rp. 15,- per gram.

Sisa berat (yang kurang dari 1kg) digratiskan biayanya apabila total berat ternyata lebih dari 10kg.

Perhatikan contoh sesi interaksi program seperti dibawah ini (teks bergaris bawah adalah input/read):

Contoh #1 Berat parcel (gram) : <u>8500</u> Detail berat: 8 kg + <u>500</u> gr Detail biaya: Rp. 80000 + Rp. 2500 Total biaya: Rp. 82500	Contoh #3 Berat parcel (gram) : <u>11750</u> Detail berat: 11 kg + <u>750</u> gr Detail biaya: Rp. 110000 + Rp. 3750 Total biaya: Rp. 110000
Contoh #2 Berat parcel (gram) : <u>9250</u> Detail berat: 9 kg + <u>250</u> gr Detail biaya: Rp. 90000 + Rp. 3750 Total biaya: Rp. 93750	

- Diberikan sebuah nilai akhir mata kuliah (NAM) [0..100] dan standar penilaian nilai mata kuliah (NMK) sebagai berikut:

NAM	NMK
NAM>80	A
72.5 < NAM <= 80	AB
65 < NAM <= 72.5	B
57.5 < NAM <= 65	BC
50 < NAM <= 57.5	C
40 < NAM <= 50	D
NAM <= 40	E

Program berikut menerima input sebuah bilangan riil yang menyatakan NAM. Program menghitung NMK dan menampilkannya.

```

01 package main
02 import "fmt"
03 func main() {
04     var nam float64
05     var nmk string
06     fmt.Println("Nilai akhir mata kuliah: ")
07     fmt.Scanln(&nam)
08     if nam > 80 {
09         nam = "A"
10     }
11     if nam > 72.5 {
12         nam = "AB"
13     }
14     if nam > 65 {
15         nam = "B"
16     }
17     if nam > 57.5 {
18         nam = "BC"
19     }
20     if nam > 50 {
21         nam = "C"
22     }
23     if nam > 40 {
24         nam = "D"
25     } else if nam <= 40 {
26         nam = "E"
27     }
28     fmt.Println("Nilai mata kuliah: ", nmk)
29 }
```

Jawablah pertanyaan-pertanyaan berikut:

- Jika **nam** diberikan adalah 80.1, apa keluaran dari program tersebut? Apakah eksekusi program tersebut sesuai spesifikasi soal?
 - Apa sajakah kesalahan dari program tersebut? Mengapa demikian? Jelaskan alur program seharusnya!
 - Perbaiki program tersebut! Ujilah dengan masukan: 93.5; 70.6; dan 49.5. Seharusnya keluaran yang diperoleh adalah 'A', 'B', dan 'D'
3. Sebuah bilangan bulat **b** memiliki faktor bilangan **f** > 0 jika **f** habis membagi **b**. Contoh: 2 merupakan faktor dari bilangan 6 karena 6 habis dibagi 2.

Buatlah program yang menerima input sebuah bilangan bulat **b**, dimana **b** > 1 . Program harus dapat mencari dan menampilkan semua faktor dari bilangan tersebut!

Perhatikan contoh sesi interaksi program seperti dibawah ini (**teks bergaris bawah** adalah input/read):

Bilangan: <u>12</u> Faktor: 1 2 3 4 6 12	Bilangan: <u>7</u> Faktor: 1 7
---	-----------------------------------

Bilangan bulat **b** > 0 merupakan bilangan prima **p** jika dan hanya jika memiliki persis dua faktor bilangan saja, yaitu 1 dan dirinya sendiri.

Lanjutkan program sebelumnya. Setelah menerima masukan sebuah bilangan bulat $b > 0$. Program tersebut mencari dan menampilkan semua faktor bilangan tersebut. Kemudian, program menentukan apakah b merupakan bilangan prima.

Perhatikan contoh sesi interaksi program seperti dibawah ini (teks bergaris bawah adalah input/read):

Bilangan: <u>12</u> Faktor: 1 2 3 4 6 12 Prima: false	Bilangan <u>11</u> Faktor: 1 11 Prima: true
---	---

Modul 5 Latihan 1 : Tipe Data dan Struktur Kontrol

1. [manytomany_1]

Keberhasilan Ahmad dan Badrun dalam kompetisi pemrograman bahasa Go, membuat mereka ketagihan untuk mengikuti kompetisi selanjutnya.

Kali ini kompetisi juga lebih menantang karena jumlah soal yang harus diselesaikan lebih dari satu.

Penilaian masing-masing solusi program masih sama, setiap program dinilai berdasarkan: struktur program, efisiensi, dan kelengkapan solusi. Nilai setiap faktor adalah antara 1 s.d. 100.

Buat program *m2m_1* untuk memeriksa siapa yang berhasil menang berdasarkan total rata-rata dari nilai semua program yang berhasil dikirimkan.

Masukan terdiri dari dua baris. Baris pertama adalah nilai untuk solusi program yang dibuat Ahmad. Dimulai dengan sebuah integer **p**, yaitu banyaknya solusi yang dibuat, diikuti dengan **3p** bilangan real nilai dari setiap faktor untuk semua solusi.

Baris kedua adalah nilai untuk solusi program yang dibuat Badrun. Struktur penempatan data sama seperti baris pertama.

Keluaran terdiri dari dua baris. Baris pertama berisi informasi kemenangan Ahmad dan rata-rata yang dia peroleh. Baris kedua informasi yang mirip tetapi untuk Badrun.

Masukan	2 99 16 8 55.5 60.5 30 3 60 60 60 60 62 90 60 70
Keluaran	Rata-rata Ahmad adalah 44.833. Menang? false Rata-rata Badrun adalah 64.667. Menang? True

Keterangan:

Rata-rata Ahmad adalah $(99+16+8+55.5+60.5+30)/6 = 44.833$.

Rata-rata Badrun adalah $(60+60+60+60+60+62+90+60+70)/9=64.667$.

2. [manytomany_2]

Ternyata rekan-rekan Ahmad dan Badrun juga berminat untuk serta kompetisi.

Karena ada banyak peserta yang hadir di kompetisi, maka pemenang medali tidak harus hanya satu orang, tetapi semua peserta yang nilainya telah mencapai batas minimum.

Kembangkan program diatas, menjadi *m2m_2*, agar jumlah peserta dapat lebih dari dua orang dan penentuan kemenangan berdasarkan aturan baru ini.

Masukan terdiri dari lebih dari 1 baris. Baris pertama berisi hanya satu buah bilangan riil, yaitu skor minimum untuk menang.

Baris-baris berikutnya berisi satu buah bilangan integer **p** diikuti dengan **3p** buah bilangan riil yang menyatakan nilai tiap faktor untuk semua soal dari seorang peserta untuk semua program.

Baris terakhir selalu berisi satu buah bilangan integer **0** (nol)

Keluaran adalah sebagai berikut: Setiap baris berisi rata-rata perolehan nilai dan pernyataan apakah peserta tersebut memenangkan medali atau tidak.

Masukan	60.5 1 99 16 8 3 17 28 30 75 80 85 55.5 60.5 30 1 73 55 60 2 60 61 62 59 50 71 3 60 60 60 60 62 90 60 70 0
Keluaran	Rata-rata peserta adalah 41. Menang? false Rata-rata peserta adalah 51.222. Menang? false Rata-rata peserta adalah 62.667. Menang? true Rata-rata peserta adalah 60.5. Menang? true Rata-rata peserta adalah 64.667. Menang? true

Keterangan:

Batas minimum rata-rata tiap peserta untuk mendapatkan medali adalah 60.5.

Peserta pertama mengerjakan satu soal saja, dan rata-ratanya adalah $(99+16+8)/3 = 41$.

Yang berhasil memenangkan medali adalah peserta ke-3, ke-4, dan ke-5.

3. [manytomany_3 - BONUS TANTANGAN]

Mengingat nilai batas minimum hanya berdasarkan perkiraan panitia. Nilai tersebut perlu dievaluasi terhadap rata-rata nilai semua peserta agar di kompetisi berikutnya nilai batas minimum ini makin representatif.

Buat program *m2m_3* yang juga mencari rata-rata keseluruhan nilai peserta. Tapi perlu diperhatikan, input yang ini sedikit berbeda dari yang sebelumnya.

Masukan adalah **n+1** baris. Baris pertama berisi tiga buah bilangan, **n** (integer), **p** (integer) dan **score** (real), yaitu jumlah peserta, jumlah soal dalam kompetisi, dan skor minimum untuk menang.

Sebanyak **n** baris berikutnya berisi **3p** buah bilangan riil yang menyatakan nilai tiap faktor untuk semua soal dari seorang peserta untuk semua program.

Keluaran terdiri dari **n+1** baris. Setiap baris berisi pernyataan apakah konstestan ke-i tersebut memenangkan medali atau tidak. Baris terakhir berisi rata-rata nilai seluruh peserta, dan apakah rata-rata ini lebih rendah atau tidak dibandingkan batas minimum yang ditetapkan panitia.

Masukan	5 2 60.5 99 16 8 55.5 60.5 30 17 28 30 75 80 85 73 55 60 90 60 70 60 61 62 59 50 71 60 60 60 60 62
Keluaran	Peserta 1 menang? false Peserta 2 menang? false Peserta 3 menang? true Peserta 4 menang? true Peserta 5 menang? false Apakah rata-rata lebih rendah dari batas minimum? true (rata-rata = 57.233)

Keterangan:

Ada 5 peserta, 2 soal yang harus dijawab tiap peserta, dan batas minimum rata-rata tiap peserta untuk mendapatkan medali adalah 60.5.

Rata-rata peserta pertama adalah $(99+16+8+55.5+60.5+30)/6 = 44.833$.

Yang berhasil memenangkan medali adalah peserta ke-3 (68) dan ke-4 (60.5).

4. [moneygame_1]

Karena Indonesia sering mengalami tingkat inflasi yang cukup tinggi, nilai nominal sangat kecil jarang sekali kita temui, dan sebaliknya nominal besar makin banyak beredar. Koin dibawah 100 rupiah pun jarang diperoleh, walaupun koin 50 rupiah dan 25 rupiah sebenarnya masih berlaku, sedangkan uang kertas 50000 rupiah dan 100000 rupiah makin sering kita lihat.

Buatlah program *money_1* yang menerima input berupa bilangan bulat yang menyatakan nilai uang. Program tersebut akan mengeluarkan pernyataan bahwa nilai uang yang diinputkan adalah nilai uang Indonesia yang valid atau tidak. Nilai uang disebut valid jika mengandung pecahan uang yang masih berlaku (pecahan terkecil Rp. 25,-). Jika nilai uang yang diinputkan bukan nilai yang valid, program akan terus meminta user untuk meminta nilai uang sampai nilai yang diinputkan adalah nilai yang valid.

Contoh input dan output (**teks bergaris bawah** adalah input) sebagai berikut

```
Nilai uang : 18525
18525 nilai yang valid
{program berhenti}

Nilai uang : 1376793 {input nilai uang pertama}
1376793 bukan nilai yang valid
Nilai uang : 25623 {input nilai uang kedua}
25623 bukan nilai yang valid
Nilai uang : 1234567825 {input nilai uang ketiga bisa diterima}
1234567825 nilai yang valid
{program berhenti}
```

5. [moneygame_2]

Buatlah program *money_2* yang membaca sebuah nilai uang yang berlaku di Indonesia. Program akan menampilkan jumlah uang sepuluh ribuan paling banyak yang dapat ditukar dari nilai uang tersebut. Dalam membuat program ini, diperkenankan hanya, dan cukup dengan, menggunakan operator aritmatika pengurangan.

Contoh input dan output sebagai berikut (teks bergaris bawah adalah input yang dimasukkan user).

```
Nilai uang : 20550
Banyaknya Rp.10.000,- : 2

Nilai uang : 500000
Banyaknya Rp.10.000,- : 50
```

6. [dadu]

Siapa yang tidak kenal Dadu? Benda yang berbentuk kubus dan setiap sisi terdapat titik-titik yang mengindikasikan nomor dari satu hingga enam. Pada soal kali ini, kita akan mencoba mensimulasikan sebuah permainan Dadu ke dalam sebuah program.

Buatlah program **tebakdadu.go**, untuk bermain tebak dadu antara anda dan Dadang. Program meminta tebakan anda untuk lemparan dadu (rentang 1..6). Dadang (pribadi ganda dari komputer) juga akan menampilkan angka tebakannya. Program akan menampilkan nilai dadu yang sebenarnya, dan menentukan siapa pemenangnya.

Petunjuk: Gunakan **rand.Intn(6) + 1** untuk menghasilkan bilangan acak antara 1 sampai dengan 6. Perhatikan pola program berikut untuk menghasilkan bilangan acak:

```
package main
import "fmt"
import "math/rand"
func main() {
    var seed int64

    fmt.Println("Masukan satu bilangan rahasia anda ")
    fmt.Scanln(&seed)
    rand.Seed(seed)
    fmt.Println("Satu nilai 1...6:", rand.Intn(6)+1)
}
```

Beberapa contoh input dan output **tebakdadu** (teks bergaris bawah adalah input):

#Contoh 1

Angka rahasia: 2019

Anda: 5

Dadang: 3

Nilai dadu 1, Tidak ada pemenang

#Contoh 2

Angka rahasia: 9

Anda: 5

Dadang: 3

Nilai dadu 5, Pemenang adalah anda

#Contoh 3

Angka rahasia: 11

Anda: 5

Dadang: 3

Nilai dadu 3, Pemenang adalah Dadang

#Contoh 4

Angka rahasia: 123456789

Anda: 4

Dadang: 4

Nilai dadu 4, Seri

7. Buat program **lempardadu.go** dengan memodifikasi program sebelumnya, dimana tebakan lebih bervariasi, yaitu menebak ganjil/genap, besar(≥ 4)/kecil. Kali ini anda bermain sendiri, tanpa bersaing dengan Dadang.

Beberapa contoh input dan output (teks **bergaris bawah** adalah input):

#Contoh 1

Angka rahasia: 2019

Anda: genap kecil

Nilai dadu 1, Anda kalah

#Contoh 2

Angka rahasia: 19

Anda: genap kecil

Nilai dadu 5, Anda kalah

#Contoh 3

Angka rahasia: 10

Anda: genap kecil

Nilai dadu 4, Anda kalah

#Contoh 4

Angka rahasia: 1

Anda: genap kecil

Nilai dadu 2, Anda menang

8. [BONUS TANTANGAN]

Buat program **dudadu.go** dengan memodifikasi program sebelumnya, dengan melakukan iterasi sampai kalah berturut-turut 3 kali atau mampu menang 3 kali (tetapi tidak harus berturutan)

Beberapa contoh input dan output (teks **bergaris bawah** adalah input):

#Contoh 1

Angka rahasia: 2019

Anda: genap besar

Nilai dadu 1, Anda kalah

Anda: ganjil kecil

Nilai dadu 6, Anda kalah

Anda: genap kecil

Nilai dadu 2, Anda menang

Anda: genap kecil

Nilai dadu 1, Anda kalah

Anda: ganjil kecil

Nilai dadu 2, Anda kalah

Anda: genap kecil

Nilai dadu 2, Anda menang
Anda: **genap besar**
Nilai dadu 4, Anda menang
Skor anda 3 dari 7 lemparan

#Contoh 2

Angka rahasia: **2019**
Anda: **genap kecil**
Nilai dadu 2, Anda menang
Anda: **genap kecil**
Nilai dadu 5, Anda kalah
Anda: **genap kecil**
Nilai dadu 1, Anda kalah
Anda: **genap kecil**
Nilai dadu 3, Anda kalah
Skor anda 1 dari 4 lemparan

9. [kompetisi_1]

Kali ini, Ahmad dan Badrun ikut kompetisi pemrograman yang lebih bergengsi. Terdapat predikat *honorable mention*. Untuk memperoleh predikat tersebut, sebuah tim harus memperoleh rata-rata skor di atas nilai standar yang ditentukan.

Penilaian masing-masing program masih sama, setiap program dinilai berdasarkan: struktur program, efisiensi, dan kelengkapan solusi. Nilai setiap faktor adalah antara 1 s.d. 100. Di akhir penilaian, diumumkan berapa banyak tim yang memperoleh predikat tersebut.

Buatlah program **honormention.go** untuk mencacah berapa banyak tim yang mendapat predikat terpuji berdasarkan batas lolos atas skor total.

Masukan baris pertama terdiri dari sebuah integer **n** dan sebuah bilangan riil **std**, yang menyatakan secara berturut-turut banyaknya tim dan batas lolos. **n** baris berikutnya, masing-masing terdiri dari 3 bilangan riil; **s1**, **s2**, dan **s3**, yang secara berturut-turut menyatakan skor untuk faktor 1, faktor 2, dan faktor 3.

Keluaran terdiri dari sebuah baris. Baris ini menyatakan banyaknya tim dengan predikat *honorable mention*.

Masukan	5 62.5 100 22.5 30 40 57.5 60 70 100.0 90 90 42.5 100 50 80 40
Keluaran	Peserta dengan predikat honorable mention ada 2 tim

Keterangan:

Ada 5 tim dengan batas lolos 62.5

Tim 1 dengan rerata $(100+22.5+30)/3 = 50.8333$ tidak mendapat honorable mention

Tim 2 dengan rerata $(40+57.5+60) / 3 = 52.5$ tidak mendapat honorable mention

Tim 3 dengan rerata $(70+100.0+90)/3 = 86.6667$ mendapat honorable mention

Tim 4 dengan rerata $(90+42.5+100)/3 = 77.5$ mendapat honorable mention

Tim 5 dengan rerata $(50+80+40) / 3 = 56.6667$ tidak mendapat honorable mention

10 . [kompetisi_2]

Buatlah program **jenius.go** untuk mencacah berapa banyak tim yang mendapat predikat luar biasa berdasarkan kesempurnaan skor yang diperoleh.

Masukan terdiri dari sejumlah baris, masing-masing terdiri dari 3 bilangan riil; **s1**, **s2**, dan **s3**, yang secara berturut-turut menyatakan skor untuk faktor 1, faktor 2, dan faktor 3. Baris terakhir, sebagai penanda berisi $<0, 0, 0>$

Keluaran terdiri dari sebuah baris. Baris ini menyatakan banyaknya tim dengan predikat **luar biasa**.

Masukan	100 22.5 30 40 57.5 60 70 100.0 90 100 100 100 50 80 40 0 0 0
Keluaran	Tim dengan predikat luar biasa ada 1 tim dari 5 tim

Keterangan:

Hanya tim 4 dengan semua skor 100

Semuanya ada 5 tim yang ikut serta.

11 . [kompetisi_3]

Untuk evaluasi kompetisi yang sama, panitia ingin mencari rerata faktor tertinggi dan terendah dari semua tim yang berpartisipasi.

Bantulah dengan membuat program **hiloaverage.go** yang mencari dan menampilkan tim dengan rerata faktor tertinggi dan terendah berdasarkan data faktor yang diberikan.

Masukan terdiri dari sejumlah baris. Setiap baris terdiri dari sebuah string **nama**, dan tiga buah bilangan riil **s1**, **s2**, dan **s3** yang secara berturut-turut menyatakan nama tim, dan skor ketiga faktor. Input diakhiri dengan entri $<"AKHIR",0,0,0>$.

Keluaran terdiri dari dua baris. Baris pertama menampilkan nama dan rerata tim dengan rerata tertinggi. Baris kedua menampilkan nama dan rerata tim dengan rerata terendah.

Masukan	Ahmad 57.5 40.4 70.8 Badrun 70.3 50.2 55.2 Candra 60.1 55.7 20.4 Dono 30.4 39.2 65.2 Edric 59.5 70.2 68.5 AKHIR 0 0 0
Keluaran	Tim dengan rerata faktor tertinggi adalah tim Edric dengan nilai 66.067 Tim dengan rerata faktor terendah adalah tim Dono dengan nilai 44.933

Keterangan:

Tim Ahmad dengan rerata $(57.5+40.4+70.8)/3 = 56.233$

Tim Badrun dengan rerata $(70.3+50.2+55.2)/3 = 58.567$

Tim Candra dengan rerata $(60.1+55.7+20.4)/3 = 45.400$

Tim Dono dengan rerata $(30.4+39.2+65.2)/3 = 44.933$

Tim Edric dengan rerata $(59.5+70.2+68.5)/3 = 66.067$

Modul 6 Subprogram

Subprogram merupakan implementasi suatu algoritma (kecil) tersendiri, yaitu satu kesatuan rangkaian instruksi yang memberikan manfaat. Subprogram juga dapat mempunyai data yang bersifat lokal untuk dirinya (kamus lokal). Argumen yang dikirimkan kesubprogram merupakan data awal agar subprogram tersebut dapat bekerja.

Go hanya mempunyai kata kunci **func** untuk deklarasi fungsi dan prosedur. Perbedaan keduanya ada pada:

- Ada/tidak ada deklarasi tipe nilai yang dikembalikan, dan
- Digunakan / tidak digunakan kata kunci **return** dalam badan subprogram tersebut.

6.1. Prosedur

Prosedur dapat dianggap sebagai pembentukan suatu **instruksi baru** yang dibuat untuk mempermudah pemahaman algoritma program yang lebih besar. Kedudukannya sama seperti instruksi dasar yang sudah ada sebelumnya (**assignment**) dan/atau instruksi yang berasal dari paket (**fmt**), seperti **fmt.Scan** dan **fmt.Println**. Karena itu selalu pilih nama prosedur yang berbentuk **kata kerja** atau sesuatu yang merepresentasikan proses sebagai nama dari prosedur.

Contoh nama-nama prosedur yang baik: **cetak**, **hitungRerata**, **cariNilai**, **belok**, **mulai**, ..

Notasi algoritma	Penulisan dalam bahasa Go
<pre>procedure sub(params) kamus .. deklarasi variabel lokal sub algoritma .. badan prosedur sub</pre>	<pre>func sub(params) { .. deklarasi variabel lokal su .. badan prosedur sub }</pre>

Contoh deklarasi prosedur mencetak 5 nilai pertama dari deret Fibonacci.

Notasi algoritma	Penulisan dalam bahasa Go
<pre>procedure cetak5Fibo kamus constant n = 5 f1, f2, f3 : integer i : integer algoritma f2 ← 0 f3 ← 1 i ← 1 while i ≤ n do output f3 f1 ← f2 f2 ← f3 f3 ← f1 + f2 i ← i + 1 endwhile</pre>	<pre>func cetak5Fibo() { const n = 5 var f1, f2, f3 int f2 = 0 f3 = 1 for i := 1; i ≤ n; i++ { fmt.Println(f3) f1 = f2 f2 = f3 f3 = f1 + f2 } }</pre>

6.2. Fungsi

Variasi lain dari suatu subprogram adalah fungsi. Seperti prosedur, fungsi juga merupakan satu kesatuan rangkaian instruksi yang memberikan manfaat. Bedanya, fungsi selalu menghasilkan/mengembalikan nilai. Maka fungsi digunakan dimana suatu nilai biasanya diperlukan; assignment nilai ke suatu variabel, bagian dari ekspresi, bagian dari argumen suatu subprogram, dsb. Karena itu selalu pilih nama fungsi yang menggambarkan nilai, seperti **kata benda** dan **kata sifat**.

Contoh nama-nama fungsi yang baik: **median**, **rerata**, **nilaiTerbesar**, **ketemu**, **selesai**, ..

Ada dua variasi deklarasi fungsi dalam bahasa Go:

- Deklarasi mirip bahasa C, dimana nilai yang dikembalikan menggunakan perintah **return** nilai, atau
- Deklarasi mirip bahasa Pascal, dimana nilai yang dikembalikan menggunakan assignment ke suatu variabel khusus. Instruksi **return** harus tetap ada (hanya saja polos, tanpa nilai), pada akhir dari kode fungsi, sebagai penanda bahwa subprogram ini adalah fungsi bukan prosedur.

Notasi algoritma	Penulisan dalam bahasa Go
<pre>function alaC(params) → type kamus .. deklarasi variabel lokal alaC algoritma .. badan prosedure alaC return nilai_untuk_aluC</pre>	<pre>func alaC(params) type { .. deklarasi variabel lokal alaC .. badan fungsi alaC .. harus ada "return nilai" }</pre>
<pre>function alaPas(params) → type kamus .. deklarasi var_lokal alaPas .. badan fungsi alaPas .. ada "alaPas = nilai"</pre>	<pre>func alaPas(params) (retval type) { .. deklarasi var_lokal alaPas .. badan fungsi alaPas .. ada "retval = nilai" return }</pre>

Contoh deklarasi fungsi yang mengembalikan nilai ke-5 dari deret Fibonacci.

Notasi algoritma	Penulisan dalam bahasa Go
<pre>function fiboKe5() → integer kamus constant n = 5 i : integer f1, f2, f3 : integer algoritma f2 ← 0 f3 ← 1 i ← 1 while i ≤ n do f1 ← f2 f2 ← f3 f3 ← f1 + f2 i ← i + 1 endwhile return f3</pre>	<pre>func fiboKe5() int { const n = 5 var f1, f2, f3 int f2 = 0 f3 = 1 for i:=1; i ≤ n; i++ { f1 = f2 f2 = f3 f3 = f1 + f2 } return f3 }</pre>

Notasi algoritma	Penulisan dalam bahasa Go
	<pre>func fiboKe5() (ke5 int) { const n = 5 var f1, f2, f3 int f2 = 0 f3 = 1 for i:=1; i <= n; i++ { f1 = f2 f2 = f3 f3 = f1 + f2 } ke5 = f3 return }</pre>

6.3. Parameter dan Argumen

Antar pemanggil subprogram dan subprogramnya dapat saling berkomunikasi melalui argumen yang diberikan melalui parameter yang dideklarasikan pada subprogram. Ada dua jenis pengiriman informasi:

- Pengiriman nilai ke subprogram. Nilai akan disalin dari pemanggil ke suatu variabel lokal di lokasi subprogram. Subprogram dapat menggunakan nilai tersebut untuk apapun, tetapi tidak dapat mengembalikan informasinya ke pemanggil karena pemanggil tidak dapat mengakses memori yang digunakan subprogram.
- Pengiriman alamat dari suatu variabel yang dimiliki pemanggil ke subprogram. Subprogram mengambil data langsung dari lokasi tersebut. Karena lokasinya di area pemanggil, setiap perubahan data dilokasi tersebut yang dilakukan oleh subprogram, nilai terakhirnya akan dapat diketahui oleh si pemanggil setelah subprogram tersebut selesai eksekusi.

Contoh prosedur, parameter dan argumen	Contoh fungsi , parameter dan argumen
<pre>func hitungFaktorial(f *int, n int) { *f = n for n > 1 { n = n - 1 *f = *f * n } } func main() { f := 10 n := 5 hitungFaktorial(&f, n) fmt.Println(f, n) }</pre>	<pre>func faktorial(n int) int { f := n for n > 1 { n = n - 1 f = f * n } } func main() { n := 5 fmt.Println(faktorial(n), n) }</pre>

Catatan

- Karena sebuah fungsi akan mengembalikan nilai melalui nama fungsinya, sangat tidak disarankan sebuah fungsi menerima parameter alamat dengan tujuan mengubah/mengirimkan nilai ke pemanggil fungsi melalui parameter. (Biasa disebut efek samping).
- Pengiriman alamat melalui parameter menghindari seluruh data dari argumen disalin ke area memori untuk subprogram. Jika data yang dikirimkan besar dan pemanggilan subprogram sering sekali, pengiriman alamat dapat mengefisienkan proses pemanggilan subprogram.

6.4. Ruang Lingkup Variabel

- Suatu variabel harus dideklarasikan lebih dahulu sebelum dapat digunakan. Artinya variabel hanya dikenal dimulai dari kapan/dimana dia dideklarasikan s.d. akhir dari blok dimana variabel tersebut dideklarasikan.
- Setiap deklarasi variabel dalam program Go dikenal dalam rentang pasangan kurawal { dan }. Variabel tersebut tidak akan dikenal diluar area pasangan kurawal tersebut.
- Variabel yang dideklarasikan secara implisit pada instruksi **for**, **if**, dan **switch**, berlaku dalam rentang instruksi terkait tersebut.
- Variabel juga tetap dikenal dalam pasangan kurawal yang bersarang didalam suatu pasangan kurawal dimana variabel tersebut dideklarasikan.
- Variabel yang dideklarasikan diluar fungsi manapun, berarti diluar suatu pasangan kurawal akan bersifat global, artinya variabel tersebut akan dikenal oleh dan dapat diakses oleh semua fungsi dan prosedur yang ada setelah deklarasi dari variabel tersebut..

Contoh segmen kode Go	Keterangan
<pre>c = 10 var c int c = c + 1</pre>	Assignment salah karena c belum dideklarasikan pada assignment yang pertama
<pre>for kondisi{ var i } fmt.Println(i)</pre>	Saat Println , variabel i dari dalam for sudah tidak dikenal
<pre>if v:=f(); v > c { fmt.Scanln(&v) } else { v = v * 3 } fmt.Println(v)</pre>	Variabel v dikenal pada instruksi Scanln dan instruksi v=v*3 (karena didalam instruksi if), tetapi tidak dikenal pada instruksi Println (karena diluar instruksi if)
<pre>for kondisi{ var w int if v > w { fmt.Println(w) } else { var w int fmt.Println(w) } }</pre>	Kedua instruksi Println berhasil mencetak nilai dalam variabel w, tetapi berasal dari hasil deklarasi w yang berbeda!

Contoh segmen kode Go	Keterangan
<pre>func h() { fmt.Scanln(&g) } var g int func f() { fmt.Println(g) } func e() { var g int g = 9 } func main() { g = 10 h() f() e() }</pre>	<p>Prosedur h tidak mengenal variabel global g, sehingga instruksi Scanln eror. Println pada prosedur f dan assignmeng g=10 pada program utama berhasil mengakses variabel global g. Assignment g=9 pada prosedur e berhasil mengakses variabel lokal di prosedur tersebut.</p>

6.5. Latihan

- Buatlah sebuah program untuk menghitung/menampilkan hasil dari fungsi komposisi **(fogoh)(x)** untuk x bilangan real, dan diketahui $f(x) = x^2$, $g(x) = x - 2$, dan $h(x) = x + 1$.

Fungsi komposisi **(fog)(x)** didefinisikan sebagai $f(g(x))$.

Berikut ini adalah spesifikasi subprogram terkait.

```
function fungsif(x : real) → real
{menerima masukan x bertipe riil dan mengembalikan x kuadrat}
```

```
function fungsig(x : real) → real
{menerima masukan x bertipe riil dan mengembalikan x-2}
```

```
function fungsih(x : real) → real
{menerima masukan x bertipe riil dan mengembalikan x+1}
```

```
function fungsifoGoH(x: real) → real
{menerima masukan x bertipe riil dan mengembalikan hasil sesuai fungsi komposisi (fogoh)(x)}
```

Simak sesi interaksi program berikut. (teks bergaris bawah adalah input/read)

```
Masukan nilai x = 7.5
f(7.50) = 56.25
g(7.50) = 5.50
h(7.50) = 8.50
(fogoh)(7.50) = 42.25
```

- Sebuah taman berbentuk lingkaran dan dikelilingi oleh suatu pagar tanaman, Radius atau jari-jari dari pusat taman diketahui. Anda berada didalam taman apabila jarak anda ke pusat taman lebih dekat daripada jarak pagar ke pusat taman.

Buatlah fungsi untuk menghitung jarak titik (x_1, y_1) dan (x_2, y_2) dimana rumus jarak adalah:
$$\text{jarak} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Catatan: Lihat paket math dalam lampiran untuk menggunakan fungsi **math.Sqrt()** untuk menghitung akar kuadrat.

```
program Taman
kamus
    function jarak(x1,y1,x2,y2 : real) → real
        { mengembalikan jarak antara titik (x1,y1) dan (x2,y2) }
algoritma
    { menerima masukan kedua titik }
    { menghitung jarak antara kedua titik }
    { menampikan jarak kedua titik }
```

Contoh interaksi program (font **bold** untuk output dan font underlined untuk input):

```
(Titik1X, Titik1Y): 3.0 0.0 { koordinat titik1 (3.0, 0.0) }
(Titik2X, Titik2Y): 0.0 4.0 { koordinat titik2 (0.0, 4.0) }
Jarak titik1 ke titik2 adalah: 5.00
```

Modifikasi program diatas dengan manambahkan fungsi untuk mengetahui suatu titik berada didalam atau diluar suatu lingkaran. Asumsi jari-jari lingkaran adalah r .

```
program Taman
kamus
    x, y, r : real      { titik pusat dan jejari lingkaran }
    xa, ya : real       { data sebarang titik }

    function jarak(x1,y1,x2,y2 : real) → real
        { mengembalikan jarak antara titik (x1,y1) dan (x2,y2) }

    function dalamLingkaran(x,y,xc,yc,r : real) → Boolean
        { mengembalikan true jika titik (x,y) dalam lingkaran dengan
          titik pusat (xc,yc) dan jejari r, dan false jika diluar }

algoritma
    { menerima masukan titik pusat dan jejari lingkaran }
    { menerima masukan satu sebarang titik }
    { periksa apakah sebarang titik tersebut berada
      didalam/diluar lingkaran yang diberikan }
    { menampikan hasil pemeriksaan }
```

Contoh sesi interaksi program (font **bold** untuk output dan font underlined untuk input):

```
(PusatX, PusatY),R: 0.0 0.0 5.0 { pusat=(0.0, 0.0) jejari=5.0 }
(TitikX, TitikY): 2.2 3.3
Titik (2.2, 3.3) berada di DALAM lingkaran
```

Contoh lain sesi interaksi program:

```
(PusatX, PusatY),R: 0.0 0.0 5.0 { pusat=(0.0, 0.0) jejari=5.0 }
(TitikX, TitikY): 3.3 4.4
Titik (3.3, 4.4) berada di LUAR lingkaran
```

Modul 7 Latihan 2 : Subprogram

1. [pertandingan_1]

Kompetisi pemrograman tingkat nasional berlangsung ketat. Setiap peserta diberikan 8 soal yang harus dapat diselesaikan dalam waktu 5 jam saja. Peserta yang berhasil menyelesaikan soal paling banyak dalam waktu paling singkat adalah pemenangnya.

Buat program **gema** yang mencari pemenang dari daftar peserta yang diberikan. Program harus dibuat modular, yaitu dengan membuat prosedur **hitungSkor** yang mengembalikan total soal dan total skor yang dikerjakan oleh seorang peserta, melalui parameter formal. Pembacaan nama peserta dilakukan di program utama, sedangkan waktu pengerjaan dibaca didalam prosedur.

prosedure hitungSkor(**in/out** soal, skor : **integer**)

Setiap baris **masukan** dimulai dengan satu string nama peserta tersebut diikuti dengan adalah 8 integer yang menyatakan berapa lama (dalam menit) peserta tersebut menyelesaikan soal. Jika tidak berhasil atau tidak mengirimkan jawaban maka otomatis dianggap menyelesaikan dalam waktu 5 jam 1 menit (301 menit).

Satu baris **keluaran** berisi nama pemenang, jumlah soal yang diselesaikan, dan nilai yang diperoleh. Nilai adalah total waktu yang dibutuhkan untuk menyelesaikan soal yang berhasil diselesaikan.

Masukan	Astuti 20 50 301 301 61 71 75 10 Bertha 25 47 301 26 50 60 65 21 Selesai
Keluaran	Bertha 7 294

Keterangan

Astuti menyelesaikan 6 soal dalam waktu 287 menit, sedangkan Bertha 7 soal dalam waktu 294 menit. Karena Bertha menyelesaikan lebih banyak, maka Bertha menang. Jika keduanya menyelesaikan sama banyak, maka pemenang adalah yang menyelesaikan dengan total waktu paling kecil.

2. [pertandingan_2]

Perbaiki program diatas menjadi **gema2** dengan memperbaiki kekurangan yang ada, yaitu:

- Menguji jika jumlah soal yang diselesaikan sama dan total pengerjaan juga sama, maka harus dinyatakan semua peserta tersebut adalah juara bersama.
Petunjuk: Gunakan operasi string untuk menyimpan semua nama pemenang.
- Memastikan bahwa total waktu pengerjaan tidak lebih dari 5 jam. Jika ada peserta dimana total waktunya lebih dari 5 jam, maka akan otomatis gugur. Buat fungsi **disq** yang mengembalikan nilai Boolean untuk kebutuhan ini.
- Memastikan bahwa data waktu adalah valid, yaitu tidak data negatif atau lebih dari 5 jam 1 menit. Buat fungsi **valid** mengembalikan nilai Boolean untuk kebutuhan ini.

3. [permutasi_1]

Permutasi adalah variasi susunan sejumlah obyek dengan memperhatikan urutan/posisi obyek dalam susunan. Contohnya jika diberikan kumpulan obyek {K,P,K}, maka permutasi yang dapat dibuat dari ketiga obyek tersebut ada 3, yaitu {KPK, KKP, PKK}.

Jika diketahui kumpulan n obyek terdiri dari k obyek yang berbeda, yang masing-masing mempunyai ada n_1, n_2, \dots, n_k buah, maka banyaknya cara obyek-obyek tersebut disusun dapat dirumuskan sebagai:

$$P(n; n_1, n_2, \dots, n_k) = \frac{n!}{n_1! \times n_2! \times \dots \times n_k!}$$

Untuk contoh diatas, $n=3, k=2, n_1=2, n_2=1$, sehingga $P(3;2,1)=3!/(2!.1!)=6/2=3$.

Buat program **faktorial** yang akan menghitung dan mencetak nilai faktorial dari nilai-nilai yang diberikan. Faktorial dari suatu nilai n adalah $f(n)=n.(n-1).(n-2)\dots.1$. Contohnya $f(5) = 5.4.3.2.1 = 120$. Perhitungan faktorial harus dibuat dalam sebuah fungsi **fact** yang mempunyai 1 buah parameter integer, yaitu nilai yang akan dicari faktorialnya.

function fact(n : **integer**) → **integer**

Masukan terdiri dari sejumlah baris. Setiap baris berupa integer positif. Proses harus berhenti setelah jumlah semua data integer dimulai dari baris kedua sama atau lebih besar dari data integer pada baris pertama!

Keluaran adalah nilai-nilai faktorial dari semua integer yang diberikan.

Masukan	7 3 2 1 1
Keluaran	5040 6 2 1 1

Keterangan:

Faktorial dari 7 adalah $7!=7.6.5.4.3.2.1=5040$, faktorial dari 3 adalah $3!=3.2.1=6$, dst. Proses berhenti karena $7 <= 3+2+1+1$.

4. [permutasi_2]

Buat program **permutasi** yang akan menghitung dan mencetak nilai permutasi berdasarkan formula yang diberikan diatas. Perhitungan permutasi harus dibuat dalam sebuah fungsi **perm** yang mempunyai 1 buah parameter integer, yaitu nilai yang akan dicari permutasinya dan fungsi **perm** tersebut harus menggunakan fungsi **fact** yang sudah dibuat sebelumnya.

function perm(n : **integer**) → **integer**

Masukan terdiri dari sejumlah baris. Setiap baris berupa integer positif. Proses harus berhenti setelah jumlah semua data integer dimulai dari baris kedua sama atau lebih besar dari data integer pada baris pertama!

Keluaran adalah nilai permutasi yang dicari.

Masukan	7 3 2 1 1
Keluaran	Permutasi = 420

Keterangan:

$n=7, n1=3, n2=2, n3=1, n4=1$, maka sesuai formula permutasi yang diberikan, nilainya adalah 420.

Masukan	7 4 4
Keluaran	Permutasi = 35

Keterangan:

$n=7, n1=4, n2=3$. Karena jumlah sampai dengan data terakhir melebihi n, maka n2 diubah menjadi 3. Output yang diperoleh berasal dari $7!/(4!.3!) = (7.6.5)/(3.2)=35$.

5. [misteri 3n+1]

Skiena dan Revilla dalam *Programming Challenges* mendefinisikan sebuah deret bilangan. Deret dimulai dengan sebuah bilangan bulat n. Jika bilangan n saat itu genap, maka suku berikutnya adalah $\frac{1}{2}n$, tetapi jika ganjil maka suku berikutnya bernilai $3n+1$. Rumus yang sama digunakan terus menerus untuk mencari suku berikutnya. Deret berakhir ketika suku terakhir bernilai 1. Sebagai contoh jika dimulai dengan n=22, maka deret bilangan yang diperoleh adalah:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Untuk suku awal sampai dengan 1000000, diketahui deret ini selalu mencapai suku dengan nilai 1.

Buat program **skiena** yang akan mencetak setiap suku dari deret yang dijelaskan diatas untuk nilai suku awal yang diberikan. Pencetakan deret harus dibuat dalam prosedur **cetakDeret** yang mempunyai 1 parameter formal, yaitu nilai dari suku awal.

prosedure cetakDeret(in n : integer)

Masukan berupa satu bilangan integer positif yang lebih kecil dari 1000000.

Keluaran terdiri dari satu baris saja. Setiap suku dari deret tersebut dicetak dalam baris yang dan dipisahkan oleh sebuah spasi.

Masukan	22
Keluaran	22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

6. [misteri_2]

Buat program **revilla** yang akan mencetak jumlah suku dari deret yang dijelaskan diatas untuk nilai suku awal yang diberikan dan suku terbesar yang muncul dalam deret tersebut. Penghitungan jumlah suku dan pencarian suku terbesar dalam deret harus dilakukan dalam prosedur bongkarDeret yang mempunyai 1 parameter input, yaitu nilai dari suku awal, dan 2 parameter output yaitu jumlah suku dan suku terbesar. Pembacaan data dan pencetak output harus dalam program utama.

prosedure bongkarDeret(in n:integer, in/out tot,max: integer)

Masukan terdiri dari sejumlah baris, masing-masing baris berisi satu bilangan integer positif yang lebih kecil dari 1000000, yaitu nilai suku awal. Input diakhiri dengan bilangan 0.

Keluaran terdiri dari sejumlah baris yang sama. Setiap baris berisi nilai suku awal, nilai suku maksimum, dan jumlah deret untuk suku awal tersebut.

Masukan	22 3 5 0
Keluaran	22 52 16 3 16 8 5 16 6

7. [misteri_3]

Diberikan sebuah angka dengan nilai lebih dari 1000000, periksa apakah deret yang dimulai dengan angka tersebut akan berakhir di suku 1. Buat program **misteri** yang dapat menyelesaikan problem tersebut secara efisien. Anda harus membuat program secara modular, yaitu logik program harus dibuat dalam fungsi dan/atau prosedur.

Masukan adalah sebuah angka lebih dari 1000000.

Keluaran adalah pesan apakah angka tersebut menghasilkan deret yang berakhir di suku 1.

Masukan	1000001
Keluaran	Deret yang diawali 1000001 pasti akan berhenti.

Modul 8 Latihan 3 : Review Materi

1. [deretrahasia_1]

Ada informasi rahasia disimpan dalam suatu kumpulan data. Tugas anda adalah mengeluarkan data rahasia tersebut.

[TERBIMBING]

Buat program **kunci2** ini. Ada sebanyak n pasang data yang tersedia dalam input. jika jumlah dari sepasang data adalah genap, data rahasia adalah data yang pertama, sedangkan jika ganjil maka data rahasia adalah data yang kedua.

Masukan terdiri dari $n+1$ baris. Baris pertama berisi satu integer positif (n) yang mungkin 0, menyatakan jumlah pasangan data. n baris berikutnya selalu berisi dua buah data integer.

Keluaran adalah n data integer yang merupakan data rahasia tersebut.

Masukan	5 75 72 101 75 108 202 123 108 111 123
Keluaran	72 101 108 108 111

Keterangan:

Ada 5 buah pasangan data integer. Pasangan pertama <75,72> mempunyai total nilai ganjil (147), jadi informasinya adalah data kedua, 72. Pasangan kedua <101,75> mempunyai nilai total genap (176), jadi informasinya adalah data pertama, 101. Cara yang sama untuk sisanya, sehingga diperoleh keluaran 72, 101, 108, 108, dan 111.

Kebetulan dalam karakter ASCII, informasi yang tersimpan adalah "Hello"

2. [deretrahasia_2]

Seperti diatas kita bermaksud mengeluarkan data rahasia. Disini jumlah data tidak diketahui sebelumnya, tetapi diketahui data diakhiri dengan pasangan nilai <-1,-1>. Data rahasia adalah data yang pasangannya habis dibagi oleh kunci rahasia.

Buat program **kuncin** ini.

Masukan terdiri dari sejumlah baris. Baris pertama berisi satu integer saja, yaitu kunci rahasia. Sejumlah baris berikutnya berisi pasangan data integer. Baris terakhir berisi pasangan data dengan nilai <-1,-1>.

Keluaran adalah informasi rahasia yang diterima, berupa sejumlah data integer.

Masukan	5 100 106 101 125 109 135 130 112 90 111 275 108 -1 -1
Keluaran	106 101 109 112 111 108

Keterangan:

Kunci rahasia adalah 5, yaitu data dari baris pertama.

Pada baris kedua, angka yang habis dibagi kunci adalah 100, sehingga angka rahasianya adalah 106. Baris berikutnya angka rahasia adalah 101, karena 125 habis dibagi kunci. dst.

3. [deretrahasia_3]

Jika dalam dua soal diatas diminta untuk mencari data rahasia yang tersimpan, sekarang cobalah buat program **rahasia** untuk membuat serangkaian data yang dapat menyimpan data rahasia.

Masukan terdiri dari 1 baris saja, yaitu jumlah data yang mau dirahasiakan, dan datanya sendiri. Mungkin saja tidak ada data yang mau dirahasiakan.

Keluaran terdiri dari (lihat soal pertama) **n+1** baris. Baris pertama berisi satu integer positif (**n**) yang mungkin 0, menyatakan jumlah pasangan data. **n** baris berikutnya selalu berisi dua buah data integer.

Masukan	5 72 101 108 108 111
Keluaran	5 75 72 101 75 108 202 123 108 111 123

Keterangan:

- a) Baris pertama berisi jumlah data yang hendak dirahasiakan, sesuai contoh input, ada 5 data. Baris-baris berikutnya berisi pasangan data. Data rahasia disimpan di posisi pertama atau kedua, dan apa data tambahan yang sesuai, dapat menggunakan fungsi random seperti yang digunakan minggu lalu.

- b) Bisa saja tidak menggunakan data acak dan data rahasia selalu di yang pertama (atau selalu di yang kedua), akan tetapi hasilnya akan kurang bagus karena gampang ditebak yang mana datanya.
- c) Jika berhasil, ujicobalah eksekusi kedua program tersebut secara bersamaan seperti contoh berikut:

```
C:\> rahasia | kunci2
```

4. [duel_1]

Persaingan dalam kompetisi makin tinggi, ada peserta yang hanya menyelesaikan sebagian saja dari soal yang diberikan. Karena itu keberhasilan dihitung dari berapa banyak soal yang berhasil diselesaikan peserta, tidak bergantung nilai.

Penilaian masing-masing program masih sama, setiap program dinilai berdasarkan: struktur program, efisiensi, dan kelengkapan solusi. Nilai setiap faktor adalah antara 0 s.d. 100. Triplet data <0,0,0> menyatakan soal terkait tidak dia selesaikan.

Buatlah program **bestof5**, untuk mengetahui siapa yang menang antara dua peserta. Pemenang adalah siapa yang paling banyak menyelesaikan soal.

Input terdiri dari **3** baris saja. Baris pertama berisi jumlah soal (integer) dan nama kedua peserta (kedua peserta tidak harus Ahmad dan Badrun). Format kedua baris berikutnya sama saja; triplet data integer. Triplet <0,0,0> jika yang bersangkutan tidak membuat soal tersebut.

Output terdiri satu baris saja, yaitu siapa pemenangnya dan berapa soal yang berhasil diselesaikan. Jika kedua pemain menyelesaikan sama banyak soal, maka cukup disebutkan "Tidak ada pemenang"

Masukan	5 Astuti Bertha 0 0 0 99 17 16 28 8 30 55 75 60 0 0 80 60 60 60 0 0 0 60 20 0 0 0 0 30 85 0
Keluaran	Astuti pemenang dengan menyelesaikan 4 dari 5 soal.

Keterangan:

Ada 5 soal, pemain adalah Astuti dan Bertha. Walau ada beberapa yang bernilai nol, tapi untuk Astuti hanya satu soal (yang pertama) dimana semua komponen penilaian nol, sedangkan bagi Bertha ada dua soal.

5. [duel_2]

Buatlah program **suddendeath** dimana salah satu pemain harus menjadi pemenangnya. Ada tiga atau lebih soal yang harus mereka jawab.

Masukan terdiri dari banyak baris. Baris pertama berisi nama kedua peserta (kedua peserta tidak harus Ahmad dan Badrun). Format baris-baris berikutnya sama saja, terdiri dari 6 nilai, 3 untuk pemain pertama, dan 3 berikutnya untuk pemain kedua. Triplet <0,0,0> jika yang bersangkutan tidak membuat soal tersebut.

Keluaran terdiri satu baris saja, yaitu siapa pemenangnya dan berapa soal yang berhasil diselesaikan.

Masukan	Astuti Bertha 0 0 0 60 60 60 99 17 16 0 0 0 28 8 30 60 20 0 0 0 80 30 85 0 55 75 60 0 0 0
Keluaran	Astuti pemenang dengan menyelesaikan 4 dari 5 soal.

Keterangan: Pemain adalah Astuti dan Bertha. Sampai 3 soal pertama, keduanya masih seri karena masing-masing hanya gagal 1 soal. Pada soal ke-4 keduanya juga berhasil. Baru pada soal ke-5 Bertha gagal, sehingga Astuti adalah pemenangnya.

Modul 9 Array dan Tipe Bentukan

Dari bentuk tipe data dasar kita dapat menstrukturkan data sesuai dengan kebutuhan:

1. Array adalah jenis tipe data yang merupakan sekumpulan data yang bertipe data sama. Karena itu masing-masing data (atau elemen) dari kumpulan tersebut dapat diakses dengan mengetahui posisinya (indeksnya) dalam kumpulan tersebut.
2. Structure atau record adalah jenis tipe data yang merupakan sekumpulan data yang masing-masing dapat mempunyai tipe yang berbeda. Karena itu masing-masing data (atau field) dari kumpulan tersebut harus diberi nama agar dapat diakses.

9.1. Record atau Structure

Record berguna untuk mengumpulkan data yang saling terkait dalam satu kelompok, sehingga dengan satu akses semua nilai dapat diperoleh. Masing-masing nilai tersimpan dalam field dari record tersebut.

Mirip bahasa C, di Go deklarasi record menggunakan kata kunci **struct**.

```
// Declaring abc as a record with fields field1, field2, ...
var abc struct {
    field1 type1
    field2 type2
    ...
}
```

Berbeda dengan bahasa pemrograman lain. kesamaan tipe dari dua variabel berjenis record bukan karena namanya tetapi karena strukturnya. Dua variabel dengan nama-nama field dan tipe field yang sama (dan dalam urutan yang sama) dianggap mempunyai tipe yang sama.

Tentunya akan lebih memudahkan jika record tersebut didefinisikan sebagai sebuah tipe baru, sehingga deklarasi record tidak perlu lagi seluruh fieldnya ditulis ulang berkali-kali.

```

// Tipe baru NewType sebagai record dengan beberapa field, ...
type NewType struct {
    field1 type1
    field2 type2
    ...
}
type (
    PointType struct {
        x, y int
    }
    CircType struct {
        center PointType
        radius float64
    }
    CircType2 struct {
        PointType
        radius float64
    }
)
// Deklarasi variabel abc dan def dengan tipe NewType
// dan phi sebagai pointer/alamat ke lokasi dengan tipe NewType
var (
    abc NewType
    def = NewType{field1:value1, field2:value2, ...}
    phi = *NewType
    circle CircType
    circ2 CircType2
)

```

Contoh penggunaan

```

// Akses field tertentu
abc.field1 = def.field1
circle.center.x = 10
circ2.x = 10

// phi menyimpan alamat def
phi = &def

// Sehingga ini menyalin semua isi abc ke def
*phi = abc

// Manipulasi field2 record yang ditunjuk phi, yaitu def
(*phi).field2 = newvalue2

// Punya arti yang sama dengan instruksi diatas
phi.field2 = newvalue2

```

9.2. Array

Array mempunyai ukuran (jumlah elemen) yang tetap (statik) selama eksekusi program, sehingga jumlah elemen array menjadi bagian dari deklarasi variabel dengan tipe array.

```
var (
    // array arr mempunyai 73 elemen, masing-masing bertipe CircType2
    arr [73]CircType

    // array buf dengan 5 elemen, dengan nilai awal 7, 3, 5, 2, dan 11.
    buf = [5]byte{7, 3, 5, 2, 11}

    // mhs adalah array dengan 2000 elemen bertipe NewType
    mhs [2000]NewType

    // rec adalah array dari array, aka matriks, atau array berdimensi-2
    rec [20][40]float64
)
```

Jumlah elemen array dapat diminta dengan fungsi **len** yang tersedia. Sebagai contoh **len(arr)** akan menghasilkan 73 untuk contoh diatas.

Indeks array dimulai dari **0**, sehingga indeks arr pada contoh adalah **0, 1.. len(arr)-1**

Contoh:

```
// Mengganti isi elemen ke-0 dengan nilai dari elemen ke-7
arr[0] = arr[7]

// Mengambil data field x dari elemen ke-i
currX = arr[i].center.x

// Mengambil elemen terakhir
n := len(arr)
buf := arr[n-1]
```

Slice (Array dinamik)

Array dalam Go juga dapat mempunyai ukuran yang dinamik. (Tidak digunakan di kelas DAP ini). Deklarasinya mirip dengan deklarasi array, tetapi jumlah elemennya dikosongkan.

```
// declaring chop as an empty slice of float64
var chop []float64

// declaring s101 as a slice
var s101 = []int{ 11, 2, 3, 5, 7, 13 }
```

Sebuah slice dapat diprealokasi menggunakan fungsi builtin **make**

```
// Prealokasi 10 elemen untuk s102 dan sejumlah tempat tambahan
var s102 []int = make([]int, 10, 20)

// Prealokasi 7 elemen untuk s103 tanpa tempat tambahan
var s103 []circType = make([]circType, 7)
```

Fungsi builtin **len** dapat digunakan untuk mengetahui ukuran slice. Fungsi lain, **cap**, dapat digunakan untuk mengetahui total tempat yang disediakan untuk slice tersebut.

```
// Cetak jumlah elemen dan tempat yang tersedia untuk sl02
fmt.Println( len(sl02), cap(sl02) )
```

Fungsi builtin **append** dapat digunakan untuk menambahkan elemen ke suatu slice, dan bila perlu memperbesar tempat untuk slice tersebut.

```
/* Append elemen baru, membuat slice baru, dan
menyimpan kembali slice baru ke variabel semula
Boleh juga disimpan ke variabel lain, sehingga
variabel semula masih menyimpan slice yang asli.
*/
sl01 = append(sl01, 17)
sl01 = append(sl01, 19, 23)
```

Sebuah slice baru juga dapat terbentuk dengan mengambil slice dari suatu array atau slice yang lain.

```
// Ambil 3 elemen pertama dari suatu slice atau array
sl04 = arr[:4]

// Ambil beberapa elemen terakhir, dimulai dari indeks 5
sl05 = sl01[5:]

// Salin semua dari slice/array aslinya
sl06 = sl05[:]

// Salin element dari indeks 3 sampai, tapi tidak termasuk, 5.
// Jadi dalam contoh hanya 2 elemen sl06[3] dan sl06[4] yang disalin
sl07 = sl06[3:5]
```

Map

Tipe array lain, sebuah array dinamik, dimana indeksnya (disini disebut **kunci**) tidak harus berbentuk integer. Indeks dapat berasal dari tipe apa saja. Struktur ini disebut **map**.

```
// Deklarasi variabel dct sebagai map bilangan bulat dengan kunci string
var dct map[string]int

// Deklarasi map lain dct1 dari elemen string dengan kunci juga string
// Mempunyai nilai awal dct1["john"] = "hi", dct1["anne"] = "darling"
var dct1 = map[string]string{ "john":"hi", "anne":"darling" }

// Deklarasi dan prealokasi tempat untuk map dct2
var dct2 map[float64]int = make(map[float64]int, 10)

// Mengambil nilai yang tersimpan dengan kunci "john"
fmt.Println( dct1["john"] )

// Mengganti nilai yang tersimpan pada kunci "anne", dan
// Membuat entri baru dengan kunci "boy"
dct1["anne"] = "lovely"
dct1["boy"] = "runaround"

// Menghapus entri dengan kunci "john"
delete(dct1, "john")
```

9.3. Konstanta Simbolik

Konstanta dapat diberi nama untuk memudahkan mengingat maksud dan manfaat dari nilai yang diberi nama tersebut. Seperti PI untuk merepresentasikan konstanta π .

```
const PI = 3.1415926535897932384626433
const MARKER = "AKHIR"
```

9.4. Tipe Bentukan (Type)

Pendefinisian tipe baru untuk memudahkan pemrograman. Dengan nama yang lebih ringkas untuk merepresentasikan tipe data yang mungkin kompleks. Tipe data apa saja dapat didefinisikan menjadi tipe baru.

```
type Mahasiswa struct {
    nama string
    ipk   real
}
type TabelMahasiswa [NMHS]Mahasiswa
type Cards [28][2]int
```

9.5. Proses Dasar Array

Pencarian Nilai Maksimum/Minimum

Ide algoritma sederhana sekali. Karena data harus diproses secara sekuensial, maka nilai atau indeks ke nilai maksimum dari data yang telah diproses disimpan untuk dibandingkan dengan data berikutnya. Nilai yang berhasil disimpan sampai algoritma tersebut berakhir adalah nilai maksimum yang dicari.

Notasi algoritma	Penulisan dalam bahasa Go
<pre>max ← 1 i ← 2 while i <= n do if a[i] > a[max] then max ← i endif i ← i + 1 endwhile</pre>	<pre>max := 0 i := 1 for i < n { if a[i] > a[max] { max = i } i = i + 1 }</pre>

Pencarian Sekuensial

Ini adalah salah satu versi pencarian secara sekuensial. Proses komputasi selalu dilakukan secara sekuensial, sehingga sehingga data harus diperiksa satu persatu. Hanya dapat memilih satu dari dua pilihan, yaitu:

- Data ditemukan, tidak melakukan proses lain, dan langsung keluar (tentunya melalui satu pintu keluar)
- Data tidak ditemukan, maju ke data berikutnya untuk diuji.

Notasi algoritma	Penulisan dalam bahasa Go
<pre>i ← 1 found ← false while i <= n and not found do if a[i] == v then found ← true else i ← i + 1 endif endwhile</pre>	<pre>i := 0 found := false for i < n && !found { if a[i] == v { found = true } else { i = i + 1 } }</pre>

Operasi Matriks: Penjumlahan Matriks

Penjumlahan dan perkalian matriks merupakan proses dasar pengolahan matriks. Agar seluruh data dapat diproses, perlu proses pengulangan yang bersarang, satu proses pengulangan untuk satu dimensi.

Tentunya dua matriks yang dijumlahkan dan matriks hasilnya harus mempunyai dimensi dan ukuran yang sama, $C[n][m] = A[n][m] + B[n][m]$

Notasi algoritma
<pre>i ← 1 while i <= n do j ← 1 while j <= m do C[i][j] ← A[i][j] + B[i][j] j ← j + 1 endwhile i ← i + 1 endwhile</pre>

Operasi Matriks: Perkalian Matriks

Perkalian matriks $C[n][m] = A[n][o] \times B[o][m]$ dimana $C[i][j] = \text{jumlah hasil perkalian } A[i][k] \times B[k][j]$ untuk semua k dari 1..o.

Notasi algoritma
<pre>i ← 1 while i <= n do j ← 1 while j <= m do C[i][j] ← 0 k ← 1 while k <= o do C[i][j] ← C[i][j] + A[i][k]*B[k][j] k ← k + 1 endwhile j ← j + 1 endwhile i ← i + 1 endwhile</pre>

9.6. Latihan

- Pos Pelayanan Terpadu (posyandu) sebagai tempat pelayanan kesehatan perlu mencatat data berat balita (dalam kg). Petugas akan memasukkan data tersebut kedalam array. Dari data yang diperoleh akan dicari berat balita terkecil, terbesar, dan reratanya.

Buatlah program dengan spesifikasi subprogram sebagai berikut:

```
procedure hitungMinMax (arrBerat : array of real; var bMin, bMax : real)
{I.S. Terdefinisi array dinamis arrBerat
 Proses: Menghitung berat minimum dan maksimum dalam array
 F.S. Menampilkan berat minimum dan maksimum balita}

function rerata (arrBerat: array of real) : real
{menghitung dan mengembalikan rerata berat balita dalam array}
```

Perhatikan sesi interaksi pada contoh berikut ini (**teks bergaris bawah adalah input/read**)

```
Masukan banyak data berat balita : 4
Masukan berat balita ke-1: 5.3
Masukan berat balita ke-2: 6.2
Masukan berat balita ke-3: 4.1
Masukan berat balita ke-4: 9.9
Berat balita minimum: 4.10 kg
Berat balita maksimum: 9.90 kg
Rerata berat balita: 6.38 kg
```

- Sebuah program digunakan untuk menyimpan dan menampilkan nama-nama klub yang memenangkan pertandingan bola pada suatu grup pertandingan.

Buatlah program yang digunakan untuk merekap skor pertandingan bola 2 buah klub bola yang berlaga.

Pertama-tama program meminta masukan nama-nama klub yang bertanding, kemudian program meminta masukan skor hasil pertandingan kedua klub tersebut. Yang disimpan dalam array adalah nama-nama klub yang menang saja. Proses input skor berhenti ketika skor salah satu atau kedua klub tidak valid (negatif). Di akhir program, tampilkan daftar klub yang memenangkan pertandingan.

Perhatikan sesi interaksi pada contoh berikut ini (**teks bergaris bawah adalah input/read**)

```
Klub A : MU
Klub B : Inter
Pertandingan 1 : 2      0      // MU = 2 sedangkan Inter = 0
Pertandingan 2 : 1      2
Pertandingan 3 : 2      2
Pertandingan 4 : 0      1
Pertandingan 5 : 3      2
Pertandingan 6 : 1      0
Pertandingan 7 : 5      2
Pertandingan 8 : 2      3
Pertandingan 9 : -1      2
Hasil 1 : MU
Hasil 2 : Inter
Hasil 3 : Draw
Hasil 4 : Inter
Hasil 5 : MU
Hasil 6 : MU
Hasil 7 : MU
Hasil 8 : Inter
Pertandingan selesai
```

3. Sebuah array digunakan untuk menampung sekumpulan karakter, anda diminta untuk membuat sebuah subprogram untuk melakukan membalikkan urutan isi array dan memeriksa apakah membentuk palindrom.

Lengkapi potongan algoritma berikut ini!

```

program Karakter
kamus
    constant NMAX : integer = 127
    type tabel = array [1..NMAX] of char
    tab : tabel
    m : integer

    procedure isiArray(output t:tabel, n:integer)
    { IS. Data tersedia dalam piranti masukan
      FS. Array t berisi sejumlah n karakter yang dimasukkan user,
      Proses input selama karakter bukanlah TITIK dan n <= NMAX}

    procedure cetakArray(input t:tabel, n:integer)
    { IS. Terdefinisi array t yang berisi sejumlah n karakter
      FS. n karakter dalam array muncul di layar}

    procedure balikanArray(input/output t:tabel, input n:integer)
    { IS. Terdefinisi array t yang berisi sejumlah n karakter
      FS. Urutan isi array t terbalik }

algoritma
    { Isi array tab dengan memanggil prosedur isiArray }
    { Balikian isi array tab dengan memanggil balikanArray }
    { Cetak isi array tab}

```

Perhatikan sesi interaksi pada contoh berikut ini (teks bergaris bawah adalah input/read)

Teks	:	S	E	N	A	N	G	.
Reverse teks	:	G	N	A	N	E	S	
Teks	:	K	A	T	A	K	.	
Reverse teks	:	K	A	T	A	K		

Modifikasi program tersebut dengan menambahkan fungsi palindrom. Tambahkan instruksi untuk memanggil fungsi tersebut dan menampilkan hasilnya pada program utama.

***Palindrom adalah teks yang dibaca dari awal atau akhir adalah sama, contoh: KATAK, APA, KASUR_RUSAK.**

```

function palindrom(t:tabel, n:integer) → Boolean
{ Mengembalikan true apabila susunan karakter didalam t membentuk
  palindrom, dan false apabila sebaliknya.
  Petunjuk: Manfaatkan prosedur balikanArray }

```

Perhatikan sesi interaksi pada contoh berikut ini (teks bergaris bawah adalah input/read)

Teks	:	K	A	T	A	K	
Palindrom	?	true					
Teks	:	S	E	N	A	N	G
Palindrom	?	false					

Modul 10 Pencarian

10.1. Pencarian

Pencarian adalah suatu proses yang lazim dilakukan di dalam kehidupan sehari-hari. Contoh penggunaannya dalam kehidupan nyata sangatlah beragam, misalnya pencarian file di dalam directory komputer, pencarian suatu teks di dalam sebuah dokumen, pencarian buku pada rak buku, dan contoh lainnya.

Pada modul ini ruang lingkup pencarian adalah pencarian suatu data tertentu pada kumpulan data atau array dengan tipe data dasar atau bentukan. Berikut ini adalah contoh kasus yang mungkin terjadi di dalam pencarian data, yaitu:

1. Pencarian nilai ekstrim (maksimum dan minimum)
2. Pencarian pada kumpulan data yang tersusun acak
3. Pencarian pada kumpulan data yang tersusun terurut

Catatan: Dalam pencarian data, sebenarnya informasi lokasi data lebih penting dibandingkan nilai yang dicari itu sendiri. Lokasi di sini maksudnya adalah indeks dari data yang ditemukan di dalam suatu array. Hal ini karena apabila data yang dicari ditemukan, maka posisi atau indeks dari data tersebut di dalam array juga diketahui. Sehingga bisa dimanfaatkan untuk proses selanjutnya, misalnya proses pertukaran data.

10.2. Pencarian Sekuensial

Pencarian secara sekuensial ini adalah pencarian yang dilakukan dari data pertama, kedua hingga terakhir secara satu persatu dan berurutan. Ciri khas dari pencarian ini adalah proses pencarian akan berhenti ketika data yang dicari ditemukan, walaupun masih terdapat elemen array yang belum dicek nilainya. Algoritma ini dikenal dengan nama Sequential Search, karena prosesnya melakukan pengecekan setiap elemen array secara satu persatu dan sekuensial dari data pertama hingga ditemukan.

1. Asumsi terdapat suatu array of integer T dengan indeks dari 0 hingga N-1, dan suatu nilai yang dicari pada array T, yaitu X.
2. Status pencarian digunakan untuk menandakan data yang dicari ditemukan atau tidak, misalnya found dengan tipe Boolean.
3. Pencarian dilakukan dari T[0] sampai ke N-1, setiap kali perbandingan dengan X, update nilai found.
4. Perulangan harus dihentikan apabila status pencarian found bernilai true (data ditemukan) atau T[N-1] telah dicek.

Notasi algoritma	Penulisan dalam 50ahasa Go
<pre>k ← 0 found ← false while k < N and not found do found ← T[k] == X k ← k + 1 endwhile</pre>	<pre>k := 0 found := false for k < N && !found { found = T[k] == X k = k + 1 }</pre>

Seperti yang telah dijelaskan sebelumnya, bahwa dalam pencarian yang perlu diperhatikan adalah lokasi dimana data ditemukan. Berikut ini adalah modifikasi algoritmanya di mana status pencarian berubah menjadi bilangan yang menyatakan indeks lokasi data ditemukan (untuk data tidak ditemukan bisa diasumsikan dengan menggunakan nilai indeks yang tidak valid, misalnya bilangan negatif seperti -1)

Notasi algoritma	Penulisan dalam 51ahasa Go
<pre> k ← 0 found ← -1 while k < N and found == -1 do if T[k] == x then found ← k endif k ← k + 1 endwhile </pre>	<pre> k := 0 found := -1 for k < N && found == -1 { if T[k] == x { found = k } k = k + 1 } </pre>

Variasi yang lain

Notasi algoritma	Penulisan dalam 51ahasa Go
<pre> k ← 0 found ← -1 while k < N-1 and T[k] != x do k ← k + 1 endwhile if T[k] == x then found ← k endif </pre>	<pre> k := 0 found := -1 for k < N-1 && T[k] != x { k = k + 1 } if T[k] == x { found = k } </pre>

10.3. Pencarian Biner/Belah Tengah

Ide algoritma adalah: (dengan asumsi data terurut dari kecil membesar, dan data dengan indeks kecil ada di ‘kiri’ dan indeks besar ada di ‘kanan’)

1. Ambil salah satu data dalam rentang data yang ada, algoritma dibawah menggunakan rentang dari **kr** s.d. **kn**. Untuk kemudahan dan alasan lainnya, biasanya yang diambil adalah paling tengah dalam rentang tersebut.
2. Jika data terambil tersebut terlalu kecil, maka ubah/geser rentang data ke sebelah kanan posisi data tersebut. Ini karena jika data terambil terlalu kecil, maka semua data sebalah kirinya juga akan terlalu kecil dari yang ingin dicari.
3. Begitu juga sebaliknya jika data terambil terlalu besar.

Notasi algoritma	Penulisan dalam bahasa Go
<pre> kr ← 1 kn ← n found ← false while kr <= kn and not found do med ← (kr+kn) div 2 if a[med] < v then kr ← med + 1 elseif a[med] > v then kn ← med - 1 else found ← true endif endwhile </pre>	<pre> kr := 0 kn := n found := false for kr < kn && !found { med := (kr+kn) / 2 if a[med] < v { kr = med + 1 } else if a[med] > v { kn = med } else { found = true } } </pre>

Modul 11 Latihan 4 : Array dan Pencarian

1. [pilkart_1]

Pada pemilihan ketua RT yang baru saja berlangsung, terdapat 20 calon ketua yang bertanding memperebutkan suara warga. Perhitungan suara dapat segera dilakukan karena warga cukup mengisi formulir dengan nomor dari calon ketua RT yang dipilihnya. Seperti biasa, selalu ada pengisian yang tidak tepat atau dengan nomor pilihan diluar yang tersedia, sehingga data juga harus divalidasi. Tugas anda untuk membuat program mencari siapa yang memenangkan pemilihan ketua RT.

Buatlah program **validasi** yang akan membaca, memvalidasi, dan menghitung suara yang diberikan dalam pemilihan ketua RT tersebut.

Masukan hanya satu baris data saja, berisi bilangan bulat valid yang kadang tersisipi dengan data tidak valid. Data valid adalah integer dengan nilai diantara 1 s.d. 20 (inklusif). Data berakhir jika ditemukan sebuah bilangan dengan nilai 0.

Keluaran dimulai dengan baris berisi jumlah data suara yang terbaca, diikuti baris yang berisi berapa banyak suara yang valid. Kemudian sejumlah baris yang mencetak data para calon apa saja yang mendapatkan suara.

Masukan	7 19 3 2 78 3 1 -3 18 19 0
Keluaran	Suara masuk: 10 Suara sah: 8 1: 1 2: 1 3: 2 7: 1 18: 1 19: 2

2. [pilkart_2]

Berdasarkan program sebelumnya, buat program **pilkart** yang mencari siapa pemenang pemilihan ketua RT. Sekaligus juga ditentukan bahwa wakil ketua RT adalah calon yang mendapatkan suara terbanyak kedua. Jika beberapa calon mendapatkan suara terbanyak yang sama, ketua terpilih adalah dengan nomor peserta yang paling kecil dan wakilnya dengan nomor peserta terkecil berikutnya.

Masukan hanya satu baris data saja, berisi bilangan bulat valid yang kadang tersisipi dengan data tidak valid. Data valid adalah bilangan bulat dengan nilai diantara 1 s.d. 20 (inklusif). Data berakhir jika ditemukan sebuah bilangan dengan nilai 0.

Keluaran dimulai dengan baris berisi jumlah data suara yang terbaca, diikuti baris yang berisi berapa banyak suara yang valid. Kemudian tercetak calon nomor berapa saja yang menjadi pasangan ketua RT dan wakil ketua RT yang baru.

Masukan	7 19 3 2 78 3 1 -3 18 19 0
Keluaran	Suara masuk: 10 Suara sah: 8 Ketua RT: 3 Wakil ketua: 19

Keterangan

Kebetulan suara terbanyak diperoleh calon nomor 3 dan 19, yaitu masing-masing mendapat 2 suara. Karena itu ketua adalah nomor 3.

3. [rahasia_1]

Gambar digital direpresentasikan oleh warna pada setiap piksel (titik) dalam gambar tersebut. Ada banyak cara menyimpan informasi tersebut, misalnya setiap titik menyimpan 3 nilai warna R (merah), G (hijau), dan B (biru) karena semua warna merupakan kombinasi dari ketiga warna dasar tersebut.

Gradasi nilai dari 0 s.d. 255 untuk setiap warna sudah cukup untuk merepresentasikan 16 juta warna yang berbeda!

Contoh input dibawah adalah contoh gambar dengan format .pnm. Silakan copy-paste contoh tersebut dalam file dengan akhiran .pnm dan kemudian lihat/buka melalui aplikasi gambar.

Gambar juga dapat tanpa warna-warni, hanya kelabu. Dalam hal ini tiap piksel dapat digambarkan oleh satu nilai intensitas cahaya saja, dengan nilai dari 0 s.d. 255.

Contoh output dibawah adalah contoh gambar kelabu dalam format .pgm. Silakan copy-paste contoh tersebut dalam file dengan akhiran .pgm dan kemudian lihat/buka melalui aplikasi gambar.

Buat program **ekstrak** yang mengekstrak informasi rahasia yang tersimpan dalam gambar input dengan format .pnm. Informasi rahasia yang diperoleh berupa gambar dengan format .pgm.

Aturan ekstrasinya sangat sederhana yaitu jika komponen merah (R) dari piksel tersebut ganjil, maka piksel pada gambar rahasia adalah putih (255). Sebaliknya jika genap, maka piksel gambar rahasia adalah hitam (0).

Masukan dimulai dengan baris isi string **P3** sebagai penanda bahwa ini adalah data dengan format .pnm. Diikuti dengan baris yang berisi dua buah integer, lebar w dan tinggi h gambar. Untuk soal disini, nilai w dan h akan sama. Baris ketiga berisi nilai integer maksimum. Dalam semua soal disini nilainya akan selalu 255. Baris-baris berikutnya berisi triplet RGB untuk setiap piksel, sehingga akan ada 3.w.h bilangan bulat 0..255.

Catatan: Ukuran gambar tidak akan lebih dari 512x512. Simpanlah gambar masukan dalam matriks (array dua dimensi) dimana tiap elemennya adalah piksel dengan informasi tersimpan dalam record dengan tiga field, masing-masing untuk nilai merah, hijau, dan biru.

```
const N = 512
type pixel struct {
    r, g, b int
}
var img [N][N]pixel
```

Keluaran dimulai dengan baris berisi string **P2** sebagai penanda bahwa output menggunakan format .pgm. Diikuti dengan baris yang berisi dua buah integer, ukuran gambar w dan h. Baris ketiga berisi angka 255. Baris-baris berikutnya berisi data piksel, yaitu bilangan bulat 0..255 sebanyak w.h.

Masukan	P3 6 6 255 87 132 122 254 151 152 114 155 61 150 119 151 42 59 88 239 96 154 166 131 50 67 143 132 237 37 52 209 180 151 17 138 76 146 174 11 72 211 44 156 103 220 155 48 43 253 118 95 214 123 222 60 84 129 204 250 83 18 103 170 101 90 165 171 1 22 206 107 248 176 133 140 136 83 222 85 37 103 107 215 18 49 247 102 11 154 38 172 86 97 83 241 37 216 225 13 146 93 85 172 90 150 194 137 197 35 25 11
Keluaran	P2 6 6 255 255 0 0 0 255 0 255 255 255 255 0 0 0 255 255 0 0 0 0 255 255 0 0 0 255 255 255 255 0 255 0 0 0 255

Keterangan:

Tiga baris pertama merupakan penanda format dan karakteristik dari gambar yang dibuat. Pada input 3 bilangan berikutnya adalah 87 132 122 merupakan warna piksel pertama (pojok kiri atas) untuk merah, hijau, dan biru. Karena merah (87) ganjil maka pada output diberikan nilai 255. Tiga bilangan setelahnya adalah 254 151 152. Kali ini merah (254) genap maka pada output dicetak nilai 0.

Gunakan input dari file (ekstensi .pnsm) yang diberikan asprak, dan simpan output ke dalam file dengan cara sbb: (Dan kemudian bukalah gambar1.pgm tersebut menggunakan aplikasi penampil gambar).

```
C:\ekstrak> go build ekstrak
C:\ekstrak> ekstrak <modul7-kecil6.pnm >gambar1.pgm
```


4. [rahasia_2]

Ada dua informasi rahasia tersimpan dalam gambar tersebut. Yang pertama telah terungkap pada soal sebelumnya. Informasi rahasia kedua dapat diungkap dengan menjumlahkan ketiga komponen merah, hijau, dan biru. Apabila jumlahnya ganjil maka informasi yang dimaksud adalah piksel putih (255), dan piksel hitam (0) jika genap.

Masukan dengan format masih sama seperti sebelumnya.

Keluaran dengan format masih sama seperti sebelumnya.

Contoh output: input sama seperti diatas

Masukan	P3 6 6 255 87 132 122 254 151 152 114 155 61 150 119 151 42 59 88 239 96 154 166 131 50 67 143 132 237 37 52 209 180 151 17 138 76 146 174 11 72 211 44 156 103 220 155 48 43 253 118 95 214 123 222 60 84 129 204 250 83 18 103 170 101 90 165 171 1 22 206 107 248 176 133 140 136 83 222 85 37 103 107 215 18 49 247 102 11 154 38 172 86 97 83 241 37 216 225 13 146 93 85 172 90 150 194 137 197 35 25 11
Keluaran	P2 6 6 255 255 255 0 0 255 255 255 0 0 0 255 255 255 255 0 0 255 255 255 255 0 0 255 255 255 255 0 0 255 255 255 0 0 0 0 255

Keterangan:

Pada input 3 bilangan piksel pertama adalah 87 132 122 merupakan warna piksel pertama (pojok kiri atas). Jumlah ketiga data adalah $87+132+122=341$, ganjil, sehingga output adalah nilai 255. Tiga bilangan setelahnya adalah 254 151 152. Jumlah ketiganya adalah $254+151+152=557$, ganjil, sehingga output adalah nilai 255. Kelompok 3 data berikutnya berjumlah $114+155+61=330$, genap, sehingga output adalah 0.

5. [rahasia_3]

Masukan dapat berisi karakter # (hashtag) pada awal baris. Jika ditemukan hashtag, maka baris tersebut harus diabaikan, dan pembacaan data dilanjutkan pada baris berikutnya.

Keluaran jika biasanya dibatasi tidak lebih dari 72 karakter per baris (boleh kurang).

Dan agar tampilan gambar lebih mulus, setiap piksel putih (255) yang bersebelahan dengan piksel hitam (0) dapat diubah menjadi piksel kelabu (128). Bersebelahan baik atas bawah maupun kiri kanan.

Perbaiki program diatas agar pembacaan input dapat mengatasi masalah hashtag ini, output juga tidak lebih dari 72 karakter per baris, dan kedua proses ekstraksi terbimbing dan mandiri digabungkan dalam satu program saja.

Gunakan os.Args seperti dicontohkan dalam lampiran buku modul, sehingga program dapat dipanggil sbb:

```
C:\ekstrak> ekstrak -merah <modul7-kecil6.pnm >gambarmerah.pgm  
C:\ekstrak> ekstrak -biru <modul7-kecil6.pnm >gambarbiru.pgm
```

6. [Ada_berapa?_1]

Sering kita menemukan problem bagaimana mengetahui jumlah data tertentu, atau berapa banyak data dalam rentang tertentu. Berapa banyak mahasiswa yang mendapatkan nilai tertinggi? Berapa orang dengan nilai toefl diatas 600? Berapa banyak pohon sepanjang suatu jalan? dsb.

Buatlah program **freqmax** yang dapat menghitung kemunculan dari nilai maksimum. Program tersebut akan membaca sejumlah nilai integer dari masukan yang kemudian dimasukkan kedalam array. Data dalam masukan tidak akan lebih dari 2000 data per sekali pemrosesan. Program mencetak keluaran nilai terbesar yang ditemukan dan berapa kali nilai tersebut muncul.

Masukan terdiri dari sejumlah baris. Baris pertama berupa sebuah bilangan bulat positif **n**, yang menyatakan berapa banyak himpunan data yang harus diproses. Setiap baris berikutnya berisi sejumlah bilangan yang diakhiri penanda (marker, bukan bagian dari data) bilangan 999.

Keluaran diperoleh untuk setiap himpunan data, sehingga akan ada **n** baris keluaran, dimana setiap baris berisi dua buah bilangan bulat, **x** dan **y**. **x** yang menyatakan nilai maksimum dalam himpunan data tersebut dan **y** yang menyatakan berapa kali data **x** tersebut muncul.

Program yang dibangun harus menggunakan subprogram dengan mengikuti kerangka yang sudah diberikan berikut ini. (Silakan dimulai dengan copy-paste teks tersebut ke editor kalian).

```
package main
import "fmt"

const N = 2000
type array [N]int

func main(){
/* buatlah kode utama sesuai dengan deskripsi masukan yang harus dibaca, proses yang harus dilakukan,dan keluaran yang harus dibuat. */
}

func isiArray(data *array, m *int){
/* IS. Data seperti terdefinisi dalam deskripsi sudah siap pada piranti masukan.
   FS. Array data berisi m (<= 2000) bilangan dari piranti masukan, diluar penanda akhir (marker) 999 */
}

func max(data array, m int) int {
/* mengembalikan nilai maksimum yang terdapat pada array data yang berisi sejumlah m bilangan bulat */
}

func hitungFrekuensi(data array, m int, key int, frek *int) {
/* IS. terdefinisi array data yang berisi m bilangan bulat, dan key yang berisi angka yang akan dicari.
   FS. frek berisi jumlah kemunculan dari key dalam array data */
}
```

Penjelasan:

Himpunan data pertama mempunyai nilai terbesar adalah 4, dan jumlah kemunculannya adalah 3 kali.

Himpunan yang kedua berisi satu nilai saja, yaitu 7 dengan total kemunculan 25 kali.

Himpunan yang terakhir berisi nilai terbesar 823. Data tersebut hanya muncul 1 kali.

Jika sudah benar, uji coba program anda dengan data yang sudah disediakan oleh asprak. (File dengan akhiran .in).

```
C:\freqmax\> go build freqmax
```

```
C:\freqmax\> freqmax <datacobal.in
```

7. [Ada berapa?_2]

Berdasarkan program sebelumnya, buat program **minmax** yang mencari nilai terkecil dan terbesar dalam data yang diberikan, kemudian menghitung berapa banyak data diantara data terkecil dan terbesar tersebut.

Masukan mempunyai format yang sama seperti tugas yang **freqmax** diatas.

Keluaran diperoleh juga untuk setiap himpunan data, sehingga akan ada **n** baris keluaran, dimana setiap baris berisi tiga buah bilangan bulat **x**, **y**, dan **z**. **x** adalah bilangan terkecil yang ditemukan, **y** bilangan terbesar yang ditemukan **z** adalah jumlah data yang berada diantara nilai **x** dan **y** tersebut.

Petunjuk modifikasi

- Tambahkan fungsi untuk mencari nilai minimum
 - Modifikasi prosedur **hitungFrekuensi** untuk menghitung nilai z , dengan menyesuaikan parameter key menjadi dua parameter sebagai batas rentang data yang dicari.

Penjelasan

Pada himpunan data pertama, nilai minimum adalah 1 dan maksimum adalah 4. Kemunculan bilangan lebih besar dari 1 tetapi kurang dari 4 ada 5 buah (2, 3, 2, 3, 3)

Jika sudah benar, uji coba program anda dengan data yang sama seperti tugas diatas. (File dengan akhiran .in).

```
C:\minmax\> go build minmax  
C:\minmax\> minmax <datacobal.in
```

8. Tanpa_basa-basi_1

Diberikan n data integer positif dalam keadaan terurut membesar dan sebuah integer lain k , apakah bilangan k tersebut ada dalam daftar bilangan yang diberikan? Jika ya, berikan indeksnya, jika tidak sebutkan "TIDAK ADA"

Buat program **mencari** dengan spesifikasi diatas. Terapkan algoritma pencarian secara sekuensial untuk pencarian ini.

Masukan terdiri dari dua baris. Baris pertama berisi dua buah integer positif, yaitu **n** dan **k**. **n** menyatakan banyaknya data, dimana $1 < n \leq 1000000$. **k** adalah bilangan yang ingin dicari. Baris kedua berisi **n** buah data integer positif yang sudah terurut membesar.

Keluaran terdiri dari satu baris saja, yaitu sebuah bilangan yang menyatakan posisi data yang dicari (**k**) dalam kumpulan data yang diberikan. Posisi data dihitung dimulai dari angka 0. Atau memberikan keluaran "TIDAK ADA" jika data **k** tersebut tidak ditemukan dalam kumpulan.

Program yang dibangun harus menggunakan subprogram dengan mengikuti kerangka yang sudah diberikan berikut ini. (Silakan dimulai dengan copy-paste teks tersebut ke editor kalian).

```
package main
import "fmt"

const NMAX = 1000000
var data [NMAX]int

func main(){
    /* buatlah kode utama yang membaca baris pertama (n dan k). kemudian data diisi oleh prosedur isiArray(n), dan pencarian oleh fungsi posisi(n,k), dan setelah itu output dicetak.*/
}

func isiArray(n int){
    /* IS. Data n sudah siap pada piranti masukan.
       FS. Array data berisi n (<=NMAX) bilangan */
}

func posisi(n, k int) int {
    /* mengembalikan posisi k dalam array data dengan n elemen. Posisi dimulai dari posisi 0. Jika tidak ada kembalikan -1 */
}
```

Masukan	12 534 1 3 8 16 32 123 323 323 534 543 823 999
Keluaran	8

Penjelasan:

Data 534 berada pada posisi ke-8 dihitung dari awal data.

Masukan	12 535 1 3 8 16 32 123 323 323 534 543 823 999
Keluaran	TIDAK ADA

9. Tanpa_basa-basi_2

Setelah berhasil, buatlah program lain **biner**, dimana metoda dalam fungsi pencarian posisi diganti menggunakan metoda pencarian biner.

Lakukan sekali lagi dengan data diatas. Pastikan hasilnya sama. Kemudian jika tidak ada masalah, gunakan data yang diberikan asprak. Cobalah dengan program **mencari** dan juga program **biner**. Perhatikan kecepatan eksekusinya.

```
C:\mencari\> go build mencari  
C:\mencari\> mencari <datacoba2.in  
C:\biner\> go build biner  
C:\biner\> biner <datacoba2.in
```

Modul 12 Pengurutan

12.1. Metoda Pengurutan Seleksi

Ide algoritma adalah: (disini data akan diurut membesar, dan data dengan indeks kecil ada di ‘kiri’ dan indeks besar ada di ‘kanan’)

1. Cari nilai terbesar didalam rentang data tersisa
2. Pindahkan / tukar tempat dengan data yang berada pada posisi paling kanan pada rentang data tersisa tersebut.
3. Ulangi proses ini sampai tersisa hanya satu data saja.

Notasi algoritma	Penulisan dalam bahasa Go
<pre>i ← n while i > 1 do max ← 1 j ← 2 while j ≤ i do if a[j] > a[max] then max ← j endif j ← j + 1 endwhile t ← a[max] a[max] ← a[i] a[i] ← t i ← i - 1 endwhile</pre>	<pre>i := n - 1 for i > 0 { max := 0 j := 1 for j < i { if a[j] > a[max] { max = j } j = j + 1 } t := a[max] a[max] = a[i] a[i] = t i = i - 1 }</pre>

12.2. Metoda Pengurutan Insersi

Ide algoritma adalah: (disini data akan diurut membesar, dan data dengan indeks kecil ada di ‘kiri’ dan indeks besar ada di ‘kanan’)

1. Untuk satu data yang belum terurut dan sejumlah data yang sudah diurutkan:

Geser data yang sudah terurut tersebut (ke kanan), sehingga ada satu ruang kosong untuk memasukkan data yang belum terurut ke dalam data yang sudah terurut dan tetap menjaga keterurutan.
2. Ulangi proses tersebut untuk setiap data yang belum terurut terhadap rangkaian data yang sudah terurut.

Notasi algoritma	Penulisan dalam bahasa Go
<pre> i ← 2 while i <= n do t ← a[i] j ← i - 1 while j >= 1 and a[j] > t do a[j+1] ← a[j] j ← j - 1 endwhile a[j+1] ← t i ← i + 1 endwhile </pre>	<pre> i := 1 for i < n { t = a[i] j = i - 1 for j >= 0 && a[j] > t { a[j+1] = a[j] j = j - 1 } a[j+1] = t i = i + 1 } </pre>

12.3. Metoda Pengurutan Dengan Dihitung

Ide algoritma adalah:

1. Hitung setiap kemunculan data yang sama (dicacah)
2. Kalkulasi nilai akumulatif data atau yang lebih kecilnya (pada data terbesar maka nilai akumulatif sama dengan jumlah seluruh data).
3. Tempatkan data sesuai dengan nilai akumulatifnya.
 - a) Sebaiknya dimulai dengan data yang paling kanan
 - b) Setiap kali satu data berhasil ditempatkan pada posisi akumulatifnya, maka nilai akumulatifnya dikurang satu.

Notasi algoritma	Penulisan dalam bahasa Go
<pre> m ← .. {rentang nilai data, c[m]} n ← .. {ukuran data, a[n], b[n]} i ← 1 while i <= m do c[i] ← 0 i ← i + 1 endwhile i ← 1 while i <= n do c[a[i]] = c[a[i]] + 1 i ← i + 1 endwhile i ← 2 while i <= m do c[i] = c[i] + c[i-1] i ← i + 1 endwhile i ← n while i >= 1 do b[c[a[i]]] ← a[i] c[a[i]] ← c[a[i]] - 1 i ← i - 1 endwhile </pre>	<pre> for i := 0; i < n; i++ { c[a[i]]++ } for i := 1; i < m; i++ { c[i] += c[i-1] } for i := n-1; i >= 0; i-- { b[c[a[i]]] = a[i] c[a[i]]-- } </pre>

12.4. Teknik Peringkat

Digunakan untuk menghindari memindah-mindah data untuk membuat keterurutan yang baru. Ini diperlukan terutama ketika:

- Data terlalu besar untuk dipindah-pindah.
- Data akan sering dipindah-pindah, misalkan karena akan diurutkan menurut field yang berlainan sesuai kebutuhan. Misalkan sekumpulan data personal perlu terurut berdasar nama, lain kali berdasarkan alamat, atau tanggal lahir, dsb.

Idenya adalah dengan menggunakan array tambahan, peringkat (rank), yang menyimpan informasi tempat baru dari data tersebut tanpa perlu memindahkan data tersebut secara eksplisit. Untuk contoh diatas, dapat dibuat array peringkat untuk masing-masing nama, alamat, dan tanggal lahir.

Selection sort dengan rank	Binary search dengan rank
<pre>i ← n while i > 1 do max ← 1 j ← 2 while j ≤ i do if a[r[j]] > a[r[max]] then max ← j endif j ← j + 1 endwhile t ← r[max] r[max] ← r[i] r[i] ← t i ← i - 1 endwhile</pre>	<pre>kr ← 1 kn ← n found ← false while kr ≤ kn and not found do med ← (kr+kn) div 2 if a[r[med]] < v then kr ← med + 1 elseif a[r[med]] > v then kn ← med - 1 else found ← true endif endwhile</pre>

12.5. Latihan

1. Dengan memodifikasi algoritma pengurutan berbasis seleksi (selection sort) diatas, buatlah algoritma untuk mengacak data. Apabila algoritma semula mencari data bernilai terbesar untuk ditempatkan disisi kanan rentang, maka algoritma pengacak akan mengambil data secara acak untuk ditempatkan disisi kanan rentang. Asumsi ada fungsi **random(n)** yang memberikan nilai acak antara 0 s.d. n-1.
2. Liga Indonesia ingin membuat pengurutan poin (klasemen) klub sepak bola. Setiap klub sepak bola memiliki nama klub dan perolehan poin.

Tugas Anda adalah membuat program berdasarkan spesifikasi berikut:

```

program KlasemenLiga
kamus
    constant NMAX : integer = 999
    type Klub < nama : string,
               poin : integer>
    type TabelKlub : array [1..NMAX] of Klub
    m           : integer                      {jumlah klub}
    tabKlub     : tabelKlub

procedure isiData(output tab: TabelKlub, n: integer)
{ I.S. data tersedia di piranti masukan
  Proses. Baca nilai n, kemudian baca n data klub
  F.S. tab berisi n data klub }

procedure cetakData(input tab: TabelKlub, n: integer)
{ I.S. Terdefinisi tab yang berisi n data klub
  F.S. Menampilkan data nama klub dan perolehan poin }

procedure sortAsc(input/output tab: TabelKlub, input n: integer)
{ I.S. Terdefinisi tab yang berisi N data klub
  Proses: Gunakan metoda selection sort pada nama klub
  F.S. Tabel tab terurut membesar berdasarkan nama klub}

procedure sortDesc(input/output tab: TabelKlub, input n: integer)
{ I.S. Terdefinisi tab yang berisi n data klub
  Proses: Gunakan metoda insertion sort pada poin perolehan klub
  F.S. Tabel tab terurut mengecil berdasarkan perolehan poin}

function indeksKlub(tab: TabelKlub, n: integer) --> integer
{ I.S. Terdefinisi tab yang berisi n data klub
  Proses: Gunakan metoda binary search pada nama klub
  F.S. Mengembalikan indeks pada tab dimana nama klub yang diberikan
       berada, atau -1 jika tidak klub dengan nama tersebut }

```

Simak sesi interaksi berikut (font **underline** menyatakan input)

```
Masukan jumlah klub: 5
Nama klub 1: Persebaya
Poin klub 1: 50
Nama klub 2: Persib
Poin klub 2: 52
Nama klub 3: Bhayangkara
Poin klub 3: 53
Nama klub 4: PSM
Poin klub 4: 61
Nama klub 5: Persija
Poin klub 5: 62
```

Urutan klub berdasarkan nama ascending

1. Nama: Bhayangkara dengan poin: 53
2. Nama: PSM dengan poin: 61
3. Nama: Persebaya dengan poin: 50
4. Nama: Persib dengan poin: 52
5. Nama: Persija dengan poin: 62

Cari klub: PSM

Klub PSM memperoleh 61 poin

Cari klub: Persema

Klub Persema tidak ada dalam daftar

Cari klub: cukup

Urutan klub berdasarkan poin descending

1. Nama: Persija dengan poin: 62
2. Nama: PSM dengan poin: 61
3. Nama: Bhayangkara dengan poin: 53
4. Nama: Persib dengan poin: 52
5. Nama: Persebaya dengan poin: 50

Modul 13 Latihan 5 : Pengurutan

1. [Hercules_punya_rumah?_1]

Hercules, preman terkenal seantero ibukota, memiliki kerabat di banyak daerah. Tentunya Hercules sangat suka mengunjungi semua kerabatnya itu.

Diberikan masukan nomor rumah dari semua kerabatnya disuatu daerah, buatlah program **rumahkerabat** yang akan menyusun nomor-nomor rumah kerabatnya secara terurut membesar menggunakan algoritma selection sort.

Masukan dimulai dengan sebuah integer **n** ($0 < n < 1000$), banyaknya daerah dimana kerabat Hercules tinggal. Isi **n** baris berikutnya selalu dimulai dengan sebuah integer **m** ($0 < m < 1000000$) yang menyatakan banyaknya rumah kerabat di daerah tersebut, diikuti dengan rangkaian bilangan bulat positif, nomor rumah para kerabat.

Keluaran terdiri dari **n** baris, yaitu rangkaian rumah kerabatnya terurut membesar di masing-masing daerah.

No.	Masukan	Keluaran
1	3 5 2 1 7 9 13 6 189 15 27 39 75 133 3 4 9 1	1 2 7 9 13 15 27 39 75 133 189 1 4 9

Keterangan:

Terdapat 3 daerah dalam contoh input, dan di masing-masing daerah mempunyai 5, 6, dan 3 kerabat.

2. [Hercules_punya_rumah?_2]

Belakangan diketahui ternyata Hercules itu tidak berani menyebrang jalan, maka selalu diusahakan agar hanya menyebrang jalan sesedikit mungkin, hanya diujung jalan. Karena nomor rumah sisi kiri jalan selalu ganjil dan sisi kanan jalan selalu genap, maka buatlah program **kerabatdekat** yang akan menampilkan nomor rumah mulai dari nomor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

Format **Masukan** masih persis sama seperti sebelumnya.

Keluaran terdiri dari **n** baris, yaitu rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap, di masing-masing daerah.

No.	Masukan	Keluaran
1	3 5 2 1 8 12 13 6 189 15 27 39 75 133 3 4 8 2	1 13 12 8 2 15 27 39 75 133 189 8 4 2

Keterangan:

Terdapat 3 daerah dalam contoh masukan. Baris kedua berisi campuran bilangan ganjil dan genap. Baris berikutnya hanya berisi bilangan ganjil, dan baris terakhir hanya berisi bilangan genap.

Petunjuk:

- a. Waktu pembacaan data, bilangan ganjil dan genap dipisahkan kedalam dua array yang berbeda, untuk kemudian masing-masing diurutkan tersendiri.
- b. Atau, tetap disimpan dalam satu array, diurutkan secara keseluruhan. Tetapi pada waktu pencetakan, mulai dengan mencetak semua nilai ganjil lebih dulu, kemudian setelah selesai cetaklah semua nilai genapnya.

3. [Ayo_kamu_juga_bisa!_1]

Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi ternama. Dalam kompetisi tersebut, setiap tim berlomba untuk menyelesaikan sebanyak mungkin problem yang diberikan. Dari 13 problem yang diberikan, ada satu problem yang menarik. Problem tersebut mudah dipahami, hampir semua tim mencoba untuk menyelesaikannya, tetapi hanya 3 tim yang berhasil. Apa sih problemnya?

"Median adalah nilai tengah dari suatu koleksi data yang sudah terurut. Jika jumlah data genap, maka nilai median adalah rerata dari kedua nilai tengahnya. Pada problem ini, semua data merupakan bilangan bulat positif, dan karenanya rerata nilai tengah dibulatkan kebawah."

Buatlah program **median** yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah 0.

Masukan berbentuk rangkaian bilangan bulat. Masukan tidak akan berisi lebih dari 1000000 data, tidak termasuk bilangan 0. Data 0 merupakan tanda bahwa median harus dicetak, tidak termasuk data yang dicari mediannya. Data masukan diakhiri dengan bilangan bulat -5313541.

Keluaran adalah median yang diminta, satu data perbaris.

No.	Masukan	Keluaran
1	7 23 11 0 5 19 2 29 3 13 17 0 -5313541	1 13 12 8 2 15 27 39 75 133 189 8 4 2

Masukan	7 23 11 0 5 19 2 29 3 13 17 0 -5313541
Keluaran	11 12

Keterangan:

Sampai bilangan 0 yang pertama, data terbaca adalah 7 23 11, setelah tersusun: 7 **11** 23, maka median saat itu adalah 11.

Sampai bilangan 0 yang kedua, data adalah 7 23 11 5 19 2 29 3 13 17, setelah tersusun diperoleh: 2 3 5 7 **11** **13** 17 19 23 29. Karena ada 10 data, genap, maka median adalah $(11+13)/2=12$.

Petunjuk:

Untuk setiap data bukan 0 (dan bukan marker -5313541) simpan ke dalam array,

Dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metoda insertion sort dan ambil mediannya.

Modul 14 Skema Pemrosesan Sekuensial (Pengayaan)

Dengan dipersenjatai bentuk pengulangan dan bentuk percabangan, banyak problem komputasi yang dapat diselesaikan. Berikut ini beberapa skema (pola) yang umum ditemukan untuk pemrosesan data (secara sekuensial).

14.1. Pembacaan data tanpa marker pada akhir rangkaian data

Proses sekuensial tanpa marker	
input n i ← 1 while i <= n do input dat // kode untuk memproses dat i ← i + 1 endwhile	fmt.Scanln(&n) i := 0 for i < n { fmt.Scanln(&dat) // kode untuk memproses dat i = i + 1 }

Contoh mencari nilai maksimum:

```
input n  
max ← BILANGAN_KECIL  
i ← 1  
while i <= n do  
    input dat  
    if dat > max then  
        max ← dat  
    endif  
    i ← i + 1  
endwhile  
output "Data terbesar ", max
```

14.2. Pembacaan data dengan marker pada akhir rangkaian data

Proses sekuensial dengan marker	
input dat while dat <> MARKER do { kode untuk memproses dat } input data endwhile	fmt.Scanln(&dat) for dat != MARKER { // kode untuk memproses dat fmt.Scanln(&dat) }

Contoh mencari nilai maksimum:

```
max ← BILANGAN_KECIL  
input dat  
while dat <> MARKER do  
    if dat > max then  
        max ← dat  
    endif  
    i ← i + 1  
endwhile  
output "Data terbesar ", max
```

14.3. Kemungkinan rangkaian data kosong, hanya ada marker

Proses sekuensial dengan kemungkinan kosong (hanya MARKER)

<pre>input dat if dat == MARKER then // kode untuk data kosong else repeat // kode memproses dat input dat until dat == MARKER endif</pre>	<pre>fmt.Scanln(&dat) if dat === MARKER { // kode untuk data kosong } else { for done=false; !done; { // kode memproses dat fmt.Scanln(&dat) done = dat==MARKER } }</pre>
--	---

Contoh mencari nilai maksimum:

```
input dat
if dat == MARKER then
    output "Tidak ada terbesar karena tidak ada data"
else
    max ← BILANGAN_KECIL
    repeat
        if dat > max then
            max ← dat
        endif
        input dat
    until dat == MARKER
    output "Data terbesar ", max
endif
```

14.4. Elemen pertama perlu diproses tersendiri / kasus khusus

Proses sekuensial dengan elemen pertama diproses khusus

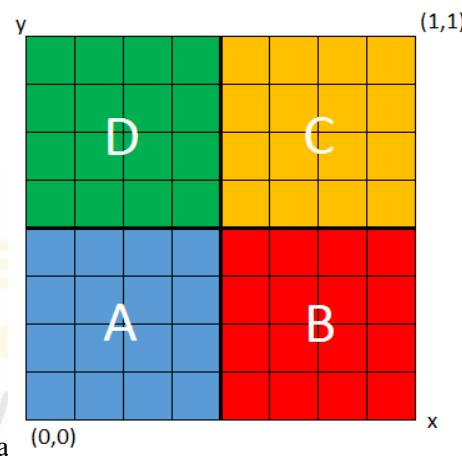
<pre>input dat if dat == MARKER then // kode untuk data kosong else // proses data pertama input dat while dat <> MARKER do // proses data berikutnya input dat endwhile endif</pre>	<pre>fmt.Scanln(&dat) if dat == MARKER { // kode untuk data kosong } else { // proses data pertama fmt.Scanln(&dat) for dat != MARKER { // proses data berikutnya fmt.Scanln(&dat) } }</pre>
--	--

Contoh mencari nilai maksimum:

```
input dat
if dat == MARKER then
    output "Tidak ada terbesar karena tidak ada data"
else
    max ← dat
    input dat
    while dat <> MARKER do
        if dat > max then
            max ← dat
        endif
        input dat
    endwhile
    output "Data terbesar ", max
endif
```

14.5. Latihan

1. Diberikan sejumlah bilangan real yang diakhiri dengan marker 9999, cari rerata dari bilangan-bilangan tersebut.
2. Diberikan string x dan n buah string, dimana x adalah data pertama yang dibaca, n adalah data bilangan yang dibaca kedua, dan n data berikutnya adalah data string. Buat algoritma untuk menjawab pertanyaan berikut:
 - a) Apakah string x ada dalam kumpulan n data string tersebut?
 - b) Pada posisi keberapa string x tersebut ditemukan?
 - c) Ada berapakah string x dalam kumpulan n data string tersebut?
 - d) Adakah sedikitnya dua string x dalam n data string tersebut?
3. Empat daerah A, B, C, dan D yang berdekatan ingin mengukur curah hujan. Keempat daerah tersebut digambarkan pada bidang berikut:



Gambar - 6. 1

Misal curah hujan dihitung berdasarkan banyaknya tetesan air hujan. Setiap tetesan berukuran 0.0001 ml curah hujan. Tetesan air hujan turun secara acak dari titik $(0,0)$ sampai $(1,1)$. Jika diterima input yang menyatakan banyaknya tetesan air hujan. Tentukan curah hujan untuk keempat daerah tersebut.

Buatlah program yang menerima input berupa banyaknya tetesan air hujan. Kemudian buat koordinat/titik (x, y) secara acak dengan menggunakan fungsi `rand.Float64()`. Hitung dan tampilkan banyaknya tetesan yang jatuh pada daerah A, B, C dan D.
Konversikan satu tetesan berukuran 0.0001 milimeter.

Catatan: Lihat lampiran untuk informasi menggunakan paket math/rand untuk menggunakan `rand.Float64()` yang menghasilkan bilangan riil acak [0..1]

Perhatikan contoh sesi interaksi program di bawah ini (teks bergaris bawah adalah input/read):

Banyaknya tetesan: **10000000**
 Curah hujan daerah A: 250.0066 milimeter
 Curah hujan daerah B: 249.8981 milimeter
 Curah hujan daerah C: 249.9930 milimeter
 Curah hujan daerah D: 250.1023 milimeter

4. Berdasarkan formula Leibniz, nilai π dapat dinyatakan sebagai deret harmonik ganti sebagai berikut:

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots = \frac{\pi}{4}$$

Suku ke- i dinyatakan sebagai S_i dan **Jumlah deret** adalah S . Apabila diketahui suku pertama $S_1 = 1$, suku kedua $S_2 = \frac{-1}{3}$. Temukan rumus untuk suku ke- i atau S_i .

Berdasarkan rumus tersebut, buatlah program yang menghitung S untuk 1000000 suku pertama.

Perhatikan contoh sesi interaksi program di bawah ini (**teks bergaris bawah** adalah input/read):

N suku pertama: **1000000**
 Hasil PI: 3.1415951

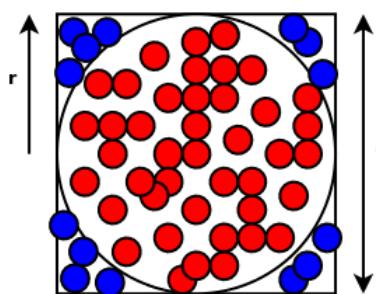
Setelah jalan, modifikasi program tersebut agar menyimpan nilai dua suku yang bersebelahan, S_i dan S_{i+1} . Buatlah agar program tersebut sekarang berhenti apabila selisih dari kedua suku tersebut tidak lebih dari 0.00001.

Perhatikan contoh sesi interaksi program di bawah ini (**teks bergaris bawah** adalah input/read):

Hasil PI: 3.1415876535
 Hasil PI: 3.1415976535
 Pada i ke: 200002

5. Monti bekerja pada sebuah kedai pizza, saking ramainya kedai tersebut membuat Monti tidak ada waktu untuk bersantai. Suatu ketika saat sedang menaburkan topping pada pizza yang diletakkan pada wadah berbentuk persegi, terfikirkan oleh Monti cara menghitung berapa **banyak topping yang dia butuhkan**, dan cara menghitung **nilai π** .

Ilustrasi seperti gambar yang diberikan dibawah, topping adalah lingkaran-lingkaran kecil. Ada yang tepat berada diatas pizza, dan ada yang jatuh didalam kotak tetapi berada diluar pizza.



Gambar - 6. 2

Apabila luas pizza yang memiliki radius r adalah $LuasPizza = \pi r^2$ dan luas wadah pizza yang memiliki panjang sisi $d = 2r$ adalah $LuasWadah = d^2 = 4r^2$, maka diperoleh perbandingan luas kedua bidang tersebut

$$\frac{LuasPizza}{LuasWadah} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

Persamaan lingkaran adalah $(x - x_c)^2 + (y - y_c)^2 = r^2$ dengan titik pusat lingkaran adalah (x_c, y_c) . Suatu titik sembarang (x, y) dikatakan berada didalam lingkaran apabila memenuhi ketidaksamaan:

$$(x - x_c)^2 + (y - y_c)^2 \leq r^2$$

Pada illustrasi topping berbentuk bulat kecil merah dan biru pada gambar adalah titik-titik (x, y) acak pada sebuah wadah yang berisi pizza. Dengan jumlah yang sangat banyak dan ditaburkan merata (secara acak), maka kita bisa mengetahui berapa banyak titik/topping yang berada tepat di dalam pizza menggunakan ketidaksamaan diatas.

Buatlah program yang menerima input berupa banyaknya topping yang akan ditaburkan, kemudian buat titik acak (x, y) dari bilangan acak riil pada kisaran nilai 0 hingga 1 sebanyak topping yang diberikan. Hitung dan tampilkan berapa banyak topping yang jatuh tepat diatas pizza.

Titik pusat pizza adalah $(0.5, 0.5)$ dan jari-jari pizza adalah 0.5 satuan wadah.

Perhatikan contoh sesi interaksi program di bawah ini (teks bergaris bawah adalah input/read):

Banyak Topping: <u>1234567</u> Topping pada Pizza: 969000	Banyak Topping: <u>10000000</u> Topping pada Pizza: 7856565
--	--

Apabila topping yang ditaburkan oleh Monti secara merata berjumlah yang sangat banyak, maka topping akan menutupi keseluruhan wadah pizza. **Luas Pizza sebanding dengan topping yang berada pada pizza**, sedangkan **Luas Wadah sebanding dengan banyaknya topping yang ditaburkan**. Dengan menggunakan rumus perbandingan luas yang diberikan diatas, maka nilai konstanta π dapat dihitung.

Modifikasi program diatas sehingga dapat menghitung dan menampilkan nilai konstanta π . Perhatikan contoh sesi interaksi program di bawah ini (teks bergaris bawah adalah input/read):

Banyak Topping: <u>1234567</u> Topping pada Pizza: 969206 PI : 3.1402297324	Banyak Topping: <u>10</u> Topping pada Pizza: 5 PI : 2.0000000000
Banyak Topping: <u>256</u> Topping pada Pizza: 198 PI : 3.0937500000	Banyak Topping: <u>5000</u> Topping pada Pizza: 3973 PI : 3.1784000000

Modul 15 Subprogram Lanjutan (Pengayaan)

15.1. Rekursif

Rekursif secara sederhana dapat diartikan sebagai cara menyelesaikan suatu masalah dengan cara menyelesaikan sub-masalah yang identik dari masalah utama. Misalnya contoh kasusnya adalah membagi kertas ke dalam delapan bagian sama besar. Pada permasalahan ini terdapat 2 cara penyelesaian:

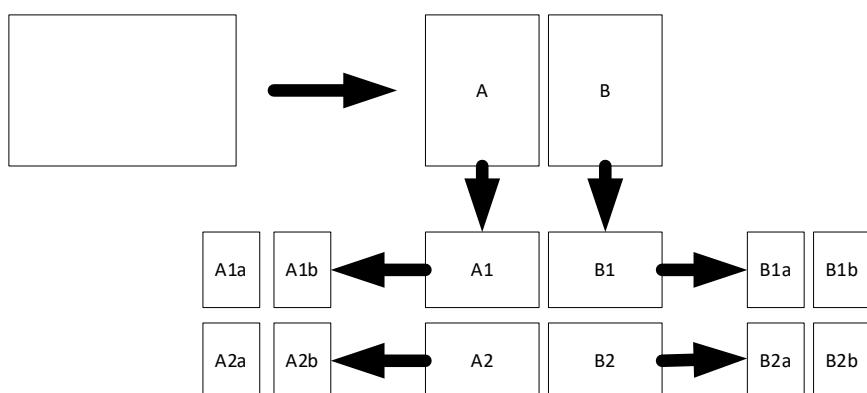


Gambar 1. Rekursi dalam sebuah gambar yang berisi lukisan seseorang yang sedang melukis dirinya sendiri yang sedang melukis (secara berulang) [6]

1. Menggunakan perulangan, di mana kertas cukup dibagi menjadi delapan bagian secara membujur seperti gambar berikut ini.



2. Menggunakan rekursif, dengan cara ini, kertas dibagi dua terlebih dahulu menjadi potongan A dan B, kemudian untuk masing-masing potongan di potong kembali menjadi dua bagian. A menjadi A1 dan A2, B menjadi B1 dan B2. Selanjutnya dari 4 potongan yang dihasilkan, masing-masingnya dipotong kembali menjadi dua bagian sama besar yaitu A1a, A1b, A2a, A2b, B1a, B1b, B2a, dan B2b.



Adakah yang pernah menonton film **Inception** yang dibintangi oleh Leonardo Dicaprio dan Ken Watanabe. The movie tells about someone who can dream inside a dream. If you think that the “dream” in the movie is a function, then you’ll find it really similar concept to the recursive function.

15.2. Induksi Matematika

Rekursif sangat erat kaitannya dengan induksi matematika. Contohnya pada perhitungan pangkat dan juga nilai faktorial.

1. Dua Pangkat n

Basis : $2^0 = 1$

Induksi : $2^n = 2 \times 2^{n-1}$, misalnya $n = 10$, maka $2^{10} = 2 \times 2^9$.

2. Faktorial

Basis : $0! = 1$

Induksi : $A! = A \times (A - 1)!$, misalnya $A = 5$, maka $5! = 5 \times 4!$

15.3. Rekursif dan Pemrograman

Umumnya rekursif menggantikan peran perulangan, dengan memanfaatkan proses induksi matematika pada suatu subprogram (function atau procedure). Oleh karena itu di dalam rekursif meniadakan struktur kontrol perulangan (while/for) di dalam algoritmanya.

Komponen Algoritma Rekursif

Algoritma rekursif terdiri dari dua komponen utama:

1. **Basis** atau **base case**, yaitu bagian untuk menghentikan proses rekursif dan menjadi komponen terpenting di dalam sebuah rekursif.
2. **Induksi**, yaitu bagian pemanggilan subprogramnya.

Misalnya diberikan sebuah subprogram berikut ini!

Notasi Algoritma	Notasi Go
<pre>function power(n:integer) → integer {fungsi untuk 2 pangkat n} algoritma if n == 0 then return 1 else return 2 * power(n-1) endif endfunction</pre>	<pre>func power(n int) int { // fungsi untuk 2 pangkat n if n == 0 { return 1 } else { return 2 * power(n-1) } }</pre>

Notasi Algoritma	Notasi Go
<pre> function faktorial(A:integer)→integer algoritma if A == 0 then return 1 else return A * faktorial(A-1) endif endfunction </pre>	<pre> func factorial(A int) int{ if A == 0 { return 1 }else{ return A * factorial(A-1) } } </pre>

15.4. Latihan

- Deret fibonanci adalah sebuah deret dengan nilai suku ke-0 dan ke-1 adalah 0 dan 1, dan nilai suku ke-n selanjutnya adalah hasil penjumlahan dua suku sebelumnya. Secara umum dapat diformulasikan $S_n = S_{n-1} + S_{n-2}$. Berikut ini adalah contoh nilai deret fibonanci hingga suku ke-10.

<i>n</i>	0	1	2	3	4	5	6	7	8	9	10
S_n	0	1	1	2	3	5	8	13	21	34	55

Buatlah program yang mengimplementasikan fungsi rekursif pada deret fibonanci tersebut.

- Buatlah sebuah program yang digunakan untuk mengecek suatu elemen dari array of char membentuk suatu pola Palindrom. Contohnya "katak", "tamat", dan "ibu ratna antar ubi".
- Buatlah sebuah program yang digunakan untuk menampilkan pola bintang berikut ini dengan menggunakan fungsi rekursif. N adalah masukan dari user.

Contoh untuk N = 5:

```

*
**
***
****
*****

```

- Sebuah program digunakan untuk membalikkan susunan array (reverse an array) dengan memanfaatkan fungsi rekursif. Buatlah program tersebut.
- Buatlah implementasi dari fungsi pencarian belah dua atau binary search dengan menggunakan fungsi rekursif.

Modul 16 Mesin Abstrak (Pengayaan)

Model komputasi terdiri dari data yang dapat diolah dan operasi-operasi dasar pengolahan data tersebut. Mesin abstrak adalah model komputasi yang dirancang diatas model mesin komputasi yang telah ada, yaitu tipe data dan operasi-operasi dasarnya dibuat menggunakan tipe data dan operasi-operasi yang tersedia dimesin dibawahnya. Teknik ini merupakan salah satu cara untuk membangun perangkat lunak.

16.1. Contoh kasus mesin domino

Tipe data dan operasi dasar mesin domino:

- Mesin abstrak ini menyediakan operasi-operasi dasar yang dapat dilakukan untuk bermain domino; seperti:
 - **kocok kartu**; mengubah dan mengacak urutan kartu yang diberikan, sehingga urutannya tidak dapat diduga lagi.
 - **ambil kartu**; mengambil satu kartu dari tumpukan kartu.
 - **gambar kartu**; melihat gambar (suit) kartu beserta nilainya.
 - **nilai kartu**; memberikan nilai kartu
- Operasi-operasi tersebut bekerja terhadap data yang spesifik, yaitu kartu domino
 - Setiap kartu mempunyai dua sisi, yang masing-masing mempunyai titik/pip dari nol (tidak ada) s.d. maksimum 6 pip.
 - Jika satu sisi kartu dianggap gambar (suit), maka sisi lainnya dianggap nilai dari kartu dengan gambar (suit) tersebut. Contohnya kartu (1,4) dapat dianggap kartu gambar 1 (titik) dengan nilai 4 atau kartu gambar 4 (titik) dengan nilai 1.
 - Nilai dari setiap kartu, yaitu jumlah titik dari kedua sisi, dari nol s.d. 12 poin
 - Karena tidak ada duplikat, dimana kartu (1,4) sama saja dengan kartu (4,1), maka semuanya ada 28 kartu yang berbeda.

Kasus sederhana bermain kartu domino:

- Diberikan sebuah kartu, cari kartu lain dari tumpukan kartu dimana salah satu gambar (suit)-nya sama dengan kartu pertama.
- Uji apakah kartu yang dipegang adalah kartu balak, yaitu kartu dimana kedua sisinya mempunyai nilai yang sama.
- Diberikan sepasang kartu, apakah nilai kedua kartu tersebut 12?

Beberapa permainan kartu domino:

- **Gapleh**; membuat rangkaian kartu domino, yang sambung menyambung berdasarkan kesamaan gambar (suit)

- **Kiu-kiu**; mendapatkan 4 kartu domino dengan total nilai yang paling tinggi, sesuai dengan aturan permainan.
- **Luzon**; sesuai dengan aturan permainan, mencari sebanyak-banyaknya pasangan kartu domino dengan jumlah nilai 12.
- **Texas 42**; seperti permainan cangkulan dengan menggunakan kartu domino.

16.2. Latihan

1. Implementasi operasi dasar mesin domino sebagai sebuah subprogram:

- Buat tipe data kartu domino (Domino) yang menyimpan informasi
 - gambar (suit) kedua sisi kartu
 - nilai kartu
 - Boolean data yang menyatakan kartu ini balak atau bukan
 Buat tipe data satu set kartu domino (Dominoes)
- Array menyimpan 28 kartu Domino
- Jumlah kartu tersisa dalam array tersebut
- b) prosedur **kocokKartu**(Dominoes)
- c) fungsi **ambilKartu**(Dominoes) → Domino
- d) fungsi **gambarKartu**(Domino,suit int) → int
- e) fungsi **nilaiKartu**(Domino) → int

2. Realisasi aksi berikut menggunakan operasi-operasi dasar mesin domino:

- prosedur **galiKartu**(Dominoes,Domino) yang mengambil kartu dari tumpukan sampai diperoleh kartu dengan gambar (suit) yang sama dengan kartu yang diberikan
- fungsi **sepasangKartu**(Domino,Domino) → Boolean; yang memberikan nilai **true** jika total nilai kartu adalah 12 dan **false** jika tidak.
- Implementasi salah satu permainan domino. Lihat lampiran untuk deskripsi permainan Gapleh.
- Implementasi mesin abstrak karakter yang bekerja terhadap untaian karakter (yang diakhiri dengan penanda titik ('.') dan mempunyai sejumlah operasi dasar.
 - Operasi dasar mesin karakter:
 - Prosedur **start()**; yang menyiapkan mesin karakter diawal rangkaian karakter.
 - Prosedur **maju()**; yang memajukan pembaca ke posisi karakter berikutnya.
 - Fungsi **eop()**; yang mengembalikan nilai true apabila sudah mencapai akhir rangkaian, sampai ke penanda titik ('.)

- Fungsi **cc()**; yang mengembalikan karakter yang sedang terbaca, atau berada pada posisi pembacaan mesin
- b) Dengan operasi dasar diatas buat algoritma untuk:
- Membaca seluruh karakter yang diberikan ke mesin karakter tersebut.
 - Menghitung berapa banyak karakter yang terbaca.
 - Menghitung ada berapa huruf 'A' yang terbaca.
 - Menghitung frekuensi kemunculan huruf 'A' terhadap seluruh karakter terbaca.
 - Menghitung ada berapa kata 'LE' (pasangan berturutan huruf 'L' dan 'E') yang terbaca.

DAFTAR PUSTAKA

1. Tutorial: <https://tour.golang.org/>
2. Dokumentasi lengkap: <https://golang.org/>
3. Daftar paket standar yang tersedia dalam bahasa Go: <https://golang.org/pkg/>
4. Situs tutorial dari pihak ketiga: <https://gobyexample.com>, <https://golangbot.com>,
<https://golangtutorial.com>, <https://www.tutorialpoint.com/go/>,
<https://www.gotutorial.org/>
5. Situs latihan pemrograman: <https://hackerrank.com>, <https://spoj.com>
6. [http://pattern-blog.com.ua/article/ponimanie-i-primerenie-rekursii-v-css](http://pattern-blog.com.ua/article/ponimanie-i-primenenie-rekursii-v-css)



LAMPIRAN

A. Paket-paket Penting

1. Paket math

Berikut adalah konstanta dan fungsi yang tersedia dalam paket **math**. Manfaat fungsi tersebut sudah sangat jelas, kecuali:

- Fungsi **Modf(x) (b,p)** mengembalikan bagian bilangan bulat **b** dan bilangan pecahan **p** dari suatu bilangan riil **x**.
- Fungsi **Pow(x,y)** menghitung **x** pangkat **y** (x^y)
- Fungsi **Pow10(n)** menghitung **10** pangkat **n** (10^n)

```
const E = 2.718281828458945235...
const Pi= 3.145926535897932384...
const Sqrt2= 1.41421356237309504880...

func Cos(x float64) float64
func Sin(x float64) float64
func Tan(x float64) float64

func Abs(x float64) float64
func Floor(x float64) float64
func Round(x float64) float64
func Trunc(x float64) float64

func Log10(x float64) float64
func Log2(x float64) float64
func Modf(x float64) (int float64, frac float64)
func Pow(x,y float64) float64
func Pow10(n int) float64
func Sqrt(x float64) float64
```

Contoh:

```
package main
import (
    "fmt"
    "math"
)
func main() {
    sinX := math.Sin(1)
    cosX := math.Cos(1)
    fmt.Println( (sinX*sinX + cosX*cosX), math.Sqrt(2) )
}

$ go build
$ ./contohmath
1 1.4142135623730951
$
```

2. Paket math/rand

Paket ini berisi pembangkit nilai-acak semu berbasis rumus matematika:

- Prosedure **Seed(s)** untuk menginisialisasi awal nilai acak
- Fungsi **Int()** menghasilkan nilai acak integer dalam rentang [-2³¹ s.d. 2³¹]
- Fungsi **Intn(n)** menghasilkan nilai integer acak dalam rentang [0 .. n), yaitu antara 0 s.d. n-1
- Fungsi **Float64()** menghasilkan nilai riil acak dalam rentang [0 .. 1.0)

```
func Seed(seed int64)
func Int() int
func Intn(n int) int
func Float64() float64
```

Contoh:

```
package main
import (
    "fmt"
    "math/rand"
)
func main() {
    rand.Seed(1)
    for i:=1; i<10; i++ {
        fmt.Println(rand.Intn(10), " ")
    }
    fmt.Println()
}

$ go build
$ ./contohrand
1 7 7 9 1 8 5 0 6
$ ./contohrand
1 7 7 9 1 8 5 0 6
$
```

3. Paket time

Paket ini berisi berbagai fungsi untuk mengambil informasi tanggal dan waktu dari sistem:

- Tipe **Time** mewakili waktu sampai satuan nanosekon
- Fungsi **Date(...)** untuk membuat data waktu dari parameter yang diberikan
- Fungsi **Now()** untuk mengambil data waktu dari sistem
- Metoda **t.Date() (y,m,d)** untuk mengambil tanggal dari data **Time t**.
- Metoda **t.Clock() (h,m,s)** untuk mengambil waktu dari data **Time t**.
- Metoda **t.UnixNano()** untuk mengambil data nanosekon dari data **Time t**.
- Metoda **t.Unix()** untuk mengambil data detik dari data **Time t**.

```

type Time
func Date( yy int, mo Month, dd, hh, mm, ss, ns int, loc *Location) Time
func Now() Time
func (t Time)Date() (year int, month Month, day int)
func (t Time)Clock() (hour, min, sec int)
func (t Time)UnixNano() int64
func (t Time)Unix() int64

```

Contoh:

```

package main
import (
    "fmt"
    "math/rand"
    "time"
)
func main() {
    curtime := time.Now()
    rand.Seed(curtime.UnixNano())
    for i:=1; i<10; i++ {
        fmt.Println(rand.Intn(10), " ")
    }
    fmt.Println()
}
$ go build
$ ./contohrandtime
6 2 2 2 8 0 4 0 6
$ ./contohrandtime
2 6 2 5 4 9 8 3 1
$ ./contohrandtime
7 5 1 2 4 7 8 6 9
$ 

```

4. Paket os

Paket ini berisi akses untuk layanan sistem operasi.

- Argumen yang diberikan waktu mengeksekusi program, termasuk nama program itu sendiri.
- ```

var Args[]string
 • Tipe data untuk handel file
type File
 • Handel file untuk input standar, diantaranya digunakan oleh fmt.Scan
var Stdin
 • Handel file untuk output standar, diantaranya digunakan oleh fmt.Print
var Stdout
 • Handel file untuk output error standar, diantaranya digunakan oleh log.Print
var Stderr
 • Fungsi Create membuat file baru atau menimpah file lama.
func Create(n string) (*File, error)
 • Fungsi Open membuka file yang sudah ada
func Open(n string) (*File, error)
 • Fungsi Close menutup file yang sudah dibuka oleh Create/Open
func (f *File)Close() error
 • Metoda Read membaca data dalam bytes dengan mengembalikan jumlah data yang berhasil dibaca.
func (f *File)Read(b []byte) (int, error)

```

- Metoda Write menulis data dalam bytes dengan mengembalikan jumlah data yang berhasil ditulis.

```
func (f *File)Write(b []byte) (int, error)
```

Contoh:

```
package main

import "log"
import "os"

func main() {
 fin, err := os.Open(os.Args[1])
 if err != nil {
 log.Fatal(err)
 } else {
 defer fin.Close()
 }
 fout, err := os.Create(os.Args[2])
 if err != nil {
 log.Fatal(err)
 } else {
 defer fout.Close()
 }
 data := make([]byte, 1000000)
 n, _ := fin.Read(data)
 fout.Write(data[:n])
}
```

```
$ go build
$ ls -l
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 15:35 sumber.dat
$./contohos sumber.dat tujuan.dat
$ ls -l
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 15:35 sumber.dat
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 17:13 tujuan.dat
$
```

## 5. Paket fmt

Ini merupakan paket yang paling sering dipakai karena menyediakan proses standar input/output teks. Terdapat tiga variasi instruksi; dua ala bahasa Pascal, dan satu ala bahasa C. Juga terdapat tiga target proses I/O, yaitu ke/dari file standar (layar monitor + keyboard), ke/dari file teks umum, dan ke/dari data string.

- Membaca teks dari **os.Stdin**, input standar yang biasanya berasal dari keyboard atau dari pengalihan input

```
func Scan(a ...interface{}) (n int, err error)
func Scanf(format string, a ...interface{}) (n int, err error)
func Scanln(a ...interface{}) (n int, err error)
```

- Membaca teks dari file umum, biasanya file yang telah dibuka oleh **os.Open**

```
func Fscan(r io.Reader, a ...interface{}) (n int, err error)
func Fscanf(r io.Reader, fmt string, ...interface{}) (int, error)
func Fscanln(r io.Reader, a ...interface{}) (n int, err error)
```

- Mengkonversi data string dengan proses **Scan**

```
func Sscan(s string, a ...interface{}) (n int, err error)
func Sscanf(s string, fmt string, a ...interface{}) (int, error)
func Sscanln(s string, a ...interface{}) (n int, err error)
```

- Menulis teks ke **os.Stdout**, output standar yang biasanya layar monitor atau dapat dialih-output ke media lain.

```
func Print(a ...interface{}) (n int, err error)
func Printf(format string, a ...interface{}) (n int, err error)
func Println(a ...interface{}) (n int, err error)
```

- Menulis teks ke file umum, biasanya file yang telah dibuka oleh **os.Create**

```
func Fprint(w io.Writer, a ...interface{}) (n int, err error)
func Fprintf(w io.Writer, fmt string, ...interface{}) (int, error)
func Fprintln(w io.Writer, a ...interface{}) (n int, err error)
```

- Mengkonversi data ke bentuk string dengan proses **Print**.

```
func Sprint(a ...interface{}) string
func Sprintf(format string, a ...interface{}) string
func Sprintln(a ...interface{}) string
```

Berikut ini beberapa kode pemformat yang dapat digunakan dalam string format:

- **%v** adalah bentuk format yang paling umum, mengambil nilai dari argumen yang diberikan dan mencetak tampilan sesuai dengan tipe data dari argumen tersebut. **%+v** dapat digunakan jika nama field yang terkait dari record/struct juga ingin dicetak.
- **%%** mencetak simbol persen (%)
- **%t** mencetak nilai Boolean: **true** atau **false**
- **%b** mencetak bilangan dengan basis-2 (biner, tiap bit)

- **%c** mencetak karakter (ASCII atau Unicode) dari bilangan integer. **%v** akan selalu mencetak nilai integernya, tidak akan mencetak karakter.
- **%d** mencetak bilangan dengan basis-10 (bilangan integer yang biasa kita lihat)
- **%o** mencetak bilangan dengan basis-8 (oktal, per 3 bit)
- **%X** mencetak bilangan dengan basis-16 (heksadesimal, per 4 bit)
- **%E** mencetak bilangan riil menggunakan notasi ilmiah (dalam bentuk  $x.y\text{E}\pm z$ ).
- **%f** mencetak bilangan riil menggunakan titik desimal. Lebar tampilan dan presisi dapat diberikan menggunakan format **%w.pf**, **w** adalah lebar tampilan dan **p** presisi yang ditampilkan.
- **%G** mencetak bilangan riil serapih mungkin.
- **%s** mencetak string

Contoh

```
Package main

import "fmt"
import "os"

func main() {
 if len(os.Args) != 3 {
 fmt.Printf("Run as: %v read|write filename\n", os.Args[0])
 fmt.Printf("eg. %v write short.txt\n", os.Args[0])
 } else if os.Args[1] == "write" {
 if file, err := os.Open(os.Args[2]); err == nil {
 file.Close()
 fmt.Printf("%v is already exist\n", os.Args[2])
 } else if file, err := os.Create(os.Args[2]); err != nil {
 fmt.Printf("%v when creating file %v\n", err, os.Args[2])
 } else {
 var oneword string

 defer file.Close()
 fmt.Print("Enter a word to store, '9999' to exit: ")
 fmt.Scanln(&oneword)
 for oneword != "9999" {
 fmt.Fprintln(file, oneword)
 fmt.Print("Enter another word to store, '9999' to exit: ")
 fmt.Scanln(&oneword)
 }
 }
 } else if os.Args[1] == "read" {
 if file, err := os.Open(os.Args[2]); err != nil {
 fmt.Printf("%v when opening file %v\n", err, os.Args[2])
 } else {
 var oneword string

 defer file.Close()
 _, err := fmt.Fscanln(file, &oneword)
 for err == nil {
 fmt.Println("Word from file:", oneword)
 _, err = fmt.Fscanln(file, &oneword)
 }
 }
 } else {
 fmt.Printf("Command is neither 'write' nor 'read'\n")
 }
}
```

## 6. Paket log

Layanan yang diberikan paket ini adalah pembuatan log, misalnya dalam kasus terjadi kesalahan eksekusi. Ada tiga jenis log:

1. **Print**: yang hanya mencetak pesan beserta waktu kejadian ke log

```
func Print(a ...interface{})
func Printf(format string, a ...interface{})
func Println(a ...interface{})
```

2. **Panic**: yang sama seperti **Print**, tetapi kemudian melakukan eksekusi **panic()**, yaitu menghentikan eksekusi normal, kemudian menjalankan semua operasi **defer**. Operasi **defer** dijalankan dari operasi **defer** terakhir s.d. **defer** yang pertama atau sampai menemui operasi **recover()**.

```
func Panic(a ...interface{})
func Panicf(format string, a ...interface{})
func Panicln(a ...interface{})
```

3. **Fatal**: melakukan operasi **Print** seperti diatas, tetapi kemudian langsung terminasi program melalui **os.Exit(1)**

```
func Fatal(a ...interface{})
func Fatalf(format string, a ...interface{})
func Fataln(a ...interface{})
```

Contoh penggunaan paket **log** ini dapat dilihat pada [contohos](#) untuk paket **os**.

## 7. Paket io

Paket ini tersedia untuk operasi file bukan teks. Beberapa fungsi yang berguna adalah:

- Untuk menduplikasi data dari satu (handel) file ke (handel) file lain. File **dest** dapat dibuat menggunakan **os.Create** dan file **src** dapat dibuka menggunakan **os.Open**.

```
func Copy(dst File, src File) (written int64, err error)
 • Untuk membaca file. Bandingkan dengan perintah os.Read sebelumnya
func ReadAtLeast(r File, buf []byte, min int) (n int, err error)
 • Untuk menuliskan string s kedalam file.
func WriteString(w File, s string) (n int, err error)
```

```

package main
/* Ini hanya contoh
Program yang baik seharusnya memeriksa berbagai kemungkinan error:
Argumen yang salah,
Ada/tidaknya file masukan,
Output akan menimpa file yang sudah ada,
Gagal copy, ...
*/
import "os"
Import "io"

func main() {
 if len(os.Args) == 3 {
 ifile,_ := os.Open(os.Args[1])
 ofile,_ := os.Create(os.Args[2])
 defer ifile.Close()
 defer ofile.Close()
 io.Copy(ofile, ifile)
 }
}

$ go build
$ ls -l
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 15:35 sumber.dat
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 17:13 tujuan.dat
$./contohio sumber.dat duplikat.dat
$ ls -l
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 15:35 sumber.dat
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 17:13 tujuan.dat
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 19:24 duplikat.dat
$
```

## 8. Paket net/http

Paket ini berisi fungsi dan prosedur untuk membuat layanan berbasis web. Berikut ini adalah beberapa fungsi/prosedur yang sering digunakan:

- Mendengarkan permintaan layanan TCP pada alamat IP dan nomor port tertentu. Permintaan yang datang akan dikirim ke handler. Handler dapat diset nil jika digunakan handler default.

```
func ListenAndServe(addr string, handler Handler) error
```

- Prosedur handler mencatat handler yang mana yang diaktifkan sesuai dengan permintaan yang diterima.

```
func Handle(pattern string, handler Handler)
```

- Handler berupa fungsi dapat langsung dideklarasikan dan dicatatkan pada prosedur handleFunc. HandleFunc akan mencatat fungsi tersebut perlu diaktifkan jika sesuai dengan permintaan

```
func HandleFunc(pattern string, handler func(ResponseWriter,
*Request))
```

- Fungsi FileServer merupakan handler khusus yang sudah tersedia untuk melayani akses ke sistem file melalui web.

```
func FileServer(root FileSystem) Handler
```

- Tipe Dir adalah implementasi dari tipe FileSystem yang berisi antarmuka yang mendukung operasi **Open**.

```
type FileSystem
```

```
type Dir
```

```
func (d Dir) Open(name string) (File, error)
```

Contoh penggunaan. Hasilnya dapat dilihat di web melalui URL <http://localhost:8080/>

```
package main

import "net/http"

func main() {
 panic(http.ListenAndServe(":8080", http.FileServer(http.Dir("."))))
}

$ go build
$ ls -l
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 15:35 sumber.dat
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 17:13 tujuan.dat
-rw-rw-r--. 1 jimmyt jimmyt 336 Jul 2 19:24 duplikat.dat
$./contohweb
```

## 9. Paket strconv

Paket ini dapat digunakan untuk mengonversi data string dari/ke tipe yang lain, yang tidak dapat dilakukan dengan cara *casting*.

- Konversi dari string representasi nilai Boolean menjadi data dengan tipe Boolean, dimana nilai Boolean **true** dapat diperoleh dari string “1”, “t”, “TRUE”, atau “true”. Begitu juga nilai Boolean **false** dapat diperoleh dari string “0”, “f”, “FALSE”, atau “false”.

```
func ParseBool(str string) (bool, error)
```

- Konversi dari string representasi nilai bilangan riil menjadi data dengan tipe **float64**. Bitsize digunakan untuk menentukan ketelitian yang diinginkan, 32 atau 64 (bit).

```
func ParseFloat(s string, bitSize int) (float64, error)
```

- Konversi dari string representasi nilai bilangan integer menjadi data dengan tipe **int**.

```
func Atoi(s string) (int, error)
```

- Konversi data integer ke representasi string, mirip dengan **fmt.Sprintf("%v", val)**

```
func Itoa(i int) string
```

Konversi sebaliknya dari tipe data lain ke tipe string dapat dilakukan menggunakan **fmt.Sprint**. Lihat paket **fmt** untuk detilnya.

Contoh konversi data:

```
package main

import "fmt"
import "strconv"

func main() {
 var (
 b bool
 f float64
 i int
 s string
)
 b, _ = strconv.ParseBool("T")
 f, _ = strconv.ParseFloat("2018.0909", 64)
 i, _ = strconv.Atoi("2018")
 s = strconv.Itoa(i)
 fmt.Println(b, f, i, s)
}

$ go build
$./contohconv
true 2018.0909 2018 2018
$
```

## 10. Paket strings

Operasi terhadap data dengan tipe **string**, selain operasi primitif **len()** dan operator primitif **+**

- Fungsi untuk memecah string menjadi beberapa potongan berdasarkan marker **sep**. Potongan string dikembalikan sebagai **slice** dari **string**.

```
func Split(s, sep string) []string
```

- Fungsi untuk mencari potongan string **subs** didalam string yang diberikan. Fungsi ini mengembalikan lokasi/index awal dimana potongan tersebut ditemukan atau -1 jika tidak ada.

```
func Index(s, substring string) int
```

- Fungsi untuk mencari potongan string **subs** didalam string yang diberikan. Sama seperti diatas, tetapi fungsi ini mengembalikan nilai Boolean **true** jika ada dan **false** jika tidak ada.

```
func Contains(s, substring string) bool
```

- Mengkonversi setiap huruf dalam string menjadi huruf kecil, atau sebaliknya huruf kapital.

```
func ToLower(s string) string
```

```
func ToUpper(s string) string
```

- Menghapus spasi dibagian depan dan belakang suatu string.

```
func TrimSpace(s string) string
```

Contoh operasi terhadap string:

```
package main

import "fmt"
import "strings"

func main() {
 many := strings.Split("Telkom University", "i")
 pos := strings.Index("Telkom University", "kom")
 found := strings.Contains("Telkom University", "Uni")
 lower := strings.ToLower("Gopher")
 upper := strings.ToUpper("Gopher")
 trimmed := strings.TrimSpace(" \t\n Hello, Gophers \n\t\r\n")
 fmt.Println(many, pos, found, lower, upper, trimmed)
}

$ go build
$./contohstr
[Telkom Un vers ty] 3 true gopher GOPHER Hello, Gophers
$
```

## 11. Paket archive/zip

Paket zip adalah satu contoh paket yang sudah tersedia dalam perkakas Go untuk kompresi, pengolahan citra, kripto, dlsb.

- Fungsi **OpenReader** digunakan untuk membaca isi file zip.

```
func OpenReader(name string) (*ReadCloser, error)
```

- Tipe **ReadCloser** menyediakan slice **File** yang berisi pasangan-pasangan nama file dan data file tersebut dalam file .zip. Selain itu tipe ini mendukung antarmuka yang mempunyai metoda **Close()**.

```
type ReadCloser
```

```
func (rc *ReadCloser) Close() error
```

- Fungsi **NewWriter** dapat digunakan untuk membuat file kompresi .zip baru atau menimpah yang lama. Fungsi ini mengembalikan handle Writer terhadap .zip file yang dapat digunakan untuk membuat/menyimpan file didalam .zip file.

```
func NewWriter(w io.Writer) *Writer
```

Contoh ala perkakas zip dan unzip. Program yang sesungguhnya perlu menguji hasil operasi terhadap file, apakah terjadi error atau tidak; seperti file tidak ada, direktori tidak ada, dlsb.

```

package main

import "archive/zip"
import "io"
import "os"

func main() {
 switch os.Args[1] {
 case "x":
 xzip, _ := zip.OpenReader(os.Args[2])
 defer xzip.Close()
 for _, file := range xzip.File {
 ifl, _ := file.Open()
 ofl, _ := os.Create(file.Name)
 io.Copy(ofl, ifl)
 ofl.Close()
 ifl.Close()
 }
 case "a":
 f, _ := os.Create(os.Args[2])
 defer f.Close()
 azip := zip.NewWriter(f)
 defer azip.Close()
 for _, file := range os.Args[3:] {
 ifl, _ := os.Open(file)
 ofl, _ := azip.Create(file)
 io.Copy(ofl, ifl)
 ifl.Close()
 }
 }
}

$ go build
$./contohzip a arsip.zip readme.txt data.dat
$
```

## 12. Paket testing

Paket ini digunakan untuk mendukung proses uji unit, uji tolok-ukur, dan uji keluaran dari program aplikasi Go. Program pengujian dibuat dalam folder yang sama dan dengan nama *package* yang sama pula, tetapi dengan nama file yang mempunyai akhir **\_test.go**. Akhiran tersebut digunakan oleh perkakas Go sebagai penanda program pengujian dan proses pengujian dieksekusi dengan memanggil **go test** sebagai pengganti **go build** dan **go run**.

- Uji unit dilakukan dalam fungsi bernama yang diawali dengan **Test** dan mempunyai parameter **\*testing.T**

```

func TestXxx(t *testing.T)
{
 • Uji tolok-ukur dilakukan dalam fungsi bernama yang diawali dengan Benchmark dan
 mempunyai parameter *testing.B
func BenchmarkXxx(b *testing.B)
{
 • Uji keluaran dilakukan dalam fungsi bernama yang diawali dengan Example dan tanpa
 parameter
```

```
func ExampleXXX()
```

Pengujian dapat melaporkan melalui metoda berikut, yang ada dalam antarmuka T dan B

- **T.Error()** untuk melaporkan hasil uji gagal, tetapi proses tetap dilanjutkan
- **T.Fatal()** untuk melaporkan hasil uji gagal dan langsung menghentikan proses pengujian
- **T.Log()** untuk melaporkan suatu catatan, tidak terkait kegagalan pengujian.

```
package hiworld

import (
 "testing"
 "fmt"
)

func TestHiDefault(t *testing.T) {
 if GetHi() != "Hello World!" {
 t.Error("Default was", GetHi(), "should be 'Hello World!'")
 }
}

func TestHiBad(t *testing.T) {
 SetHi("Welcome to the world")
 if GetHi() == "Welcome to me" {
 t.Error("Different values on purpose")
 }
}

func BenchmarkHello(b *testing.B) {
 for i := 0; i < b.N; i++ {
 fmt.Sprintf("hello")
 }
}

func ExampleOutput() {
 SetHi("World in my hand")
 fmt.Println(GetHi())
 // Output:
 // World in my hand
}

$ go test
PASS
ok hello2/hiworld 0.002s
$ go test -bench Hello
goos: linux
goarch: amd64
pkg: hello2/hiworld
BenchmarkHello-4 20000000 92.6 ns/op
PASS
ok hello2/hiworld 1.947s
$
```

## B. Perintah Go

Bahasa pemrograman Go agak unik dalam arti perkakas pendukung terintegrasi dalam satu perkakas saja, yaitu perintah **go**

- **go version**: Menampilkan versi dan platform implementasi perkakas Go
- **go help**: Menampilkan ringkasan informasi penggunaan perkakas Go
- **go env GOPATH**: Menampilkan lokasi folder untuk menyimpan program Go
  - Program sumber di **\$GOPATH/src**
  - Program biner/eksekutabel di **\$GOPATH/bin**
  - Pustaka paket program di **\$GOPATH/pkg**
- **go env GOROOT**: Menampilkan lokasi instalasi perkakas Go
- **go build**: Melakukan proses kompilasi dalam folder *current*. Jika sukses, akan menghasilkan program eksekutabel. (Yaitu file dengan ekstensi *.exe* untuk os Windows)
- **go clean**: Membersihkan sisa-sisa hasil kompilasi sebelumnya
- **go run filename.go**: Mengompilasi dan langsung mengeksekusi program tersebut. Program eksekutabel tidak disimpan, sehingga memberikan ilusi seolah Go adalah sebuah interpreter,
- **go fmt**: Me-reformat program sumber yang ada di folder *current*, sehingga memenuhi standar format penulisan program sumber Go
- **go test**: Melakukan uji (tahap unit), baik uji logik maupun tolok-ukur (*benchmark*). Program sumber uji harus mempunyai ekstensi **\_test.go**.
- **go install**: Untuk menginstall program hasil kompilasi pada lokasi yang dituju, biasanya di **\$GOPATH/bin/**
- **go get**: Untuk mengunduh dan menginstal paket dari pihak ketiga atau versi baru/lain perkakas Go. Paket akan diinstal di **\$GOPATH/pkg/**, sedangkan perkakas akan disimpan di **\$GOPATH/bin/**.
  - **go get golang.org/dl/go1.12.5**: Untuk mengunduh Go versi 1.12.5
  - **go get lokasi\_paket**: Untuk mengunduh dan menginstal paket dari pihak ketiga.  
Lihat instalasi paket GUI dibawah untuk contoh.

## C. Instalasi Go Perkakas

1. Unduh perkakas Go dengan versi yang sesuai untuk perangkat keras dan sistem operasi yang digunakan di <https://golang.org/dl> dan ikuti petunjuk cara instalasinya. Selanjutnya, semua program sumber dan hasil kompilasinya akan disimpan di **%GOPATH%**

- a) **\$HOME/go**, dengan subfolder **src** untuk program sumber, **bin** untuk instal program eksekutable hasil kompilasi, dan **pkg** untuk instal paket hasil kompilasi atau paket pihak ketiga.
  - b) **C:\Users\username\go**, dimana username adalah nama user yang sedang login .
2. Untuk menghapus perangkat lunak Go dari sistem, silakan hapus folder **%GOROOT%** berikut:
- a) Linux dan Mac: folder **/usr/local/go**
  - b) Windows: folder **C:\Go**
3. Untuk menginstal perkakas versi lain (terbaru) dari Go, melalui command prompt/terminal/shell lakukan perintah Go berikut untuk menginstal versi 1.12.6:

```
go get golang.org/dl/go1.12.6
```

```
go1.12.6 download
```

#### D. Instalasi Paket Pihak ke-3 (Contoh paket GUI)

Kunjungi situs dimana paket tersebut tersedia, umumnya ada di <https://github.com/>. Baca dan ikuti petunjuk instalasinya. Paket GUI yang berada di <https://github.com/andlabs/ui> dapat diinstal dengan menggunakan perintah **go get** sbb:

```
go get -u github.com/andlabs/ui
```

Contoh program untuk mencoba hasil instal paket GUI tersebut

```
package main
import "strconv"
import "github.com/andlabs/ui"

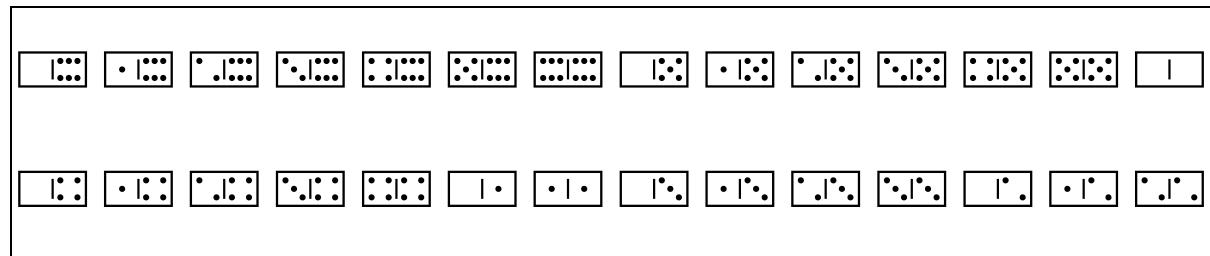
const STRETCHED = true
const GREETING = "Colek kiborna, teras pencet tombolna "

func gui() {
 input := ui.NewEntry()
 exbutton := ui.NewButton("Rengse")
 greeting := uiNewLabel(GREETING)
 vbox := ui.NewVerticalBox()
 vbox.Append(greeting, STRETCHED)
 vbox.Append(input, !STRETCHED)
 vbox.Append(exbutton, STRETCHED)
 window := ui.NewWindow("Sampurasun Dunya", 200, 100, false)
 window.SetMargined(true)
 window.SetChild(vbox)
 input.OnChanged(func(*ui.Entry) {
 greeting.SetText(GREETING + strconv.Itoa(len(input.Text())))
 })
 exbutton.OnClicked(func(*ui.Button) {
 ui.Quit()
 })
 window.OnClosing(func(*ui.Window) bool {
 ui.Quit()
 return true
 })
 window.Show()
}

func main() {
 err := ui.Main(gui)
 if err != nil {
 panic(err)
 }
}
```

## E. Contoh TUBES: Permainan Solitaire Gapleh

Kartu domino yang biasa digunakan bermain gapleh berisi 28 kartu seperti yang ditampilkan dalam gambar dibawah ini. Bagian depan kartu terbagi oleh sebuah garis menjadi dua sisi. Tiap sisi dapat memiliki titik dari 0 s.d. 6. Jika kedua sisi mempunyai jumlah titik yang sama, maka disebut kartu balak. Ada 7 kartu balak; 0-0, 1-1, s.d. 6-6. Nilai total kartu adalah jumlah titik di kedua sisi, sehingga nilainya bervariasi dari 0 s.d. 12.



Gambar - 6. 3

Kartu domino dikocok, kemudian pemain dibagi 7 kartu dari tumpukan kartu. Satu kartu dari tumpukan dibuka untuk memulai permainan. Pemain harus memilih salah satu kartunya yang mempunyai gambar yang sama dengan salah satu ujung rantai kartu dan dipasangkan diujung rantai kartu tersebut.

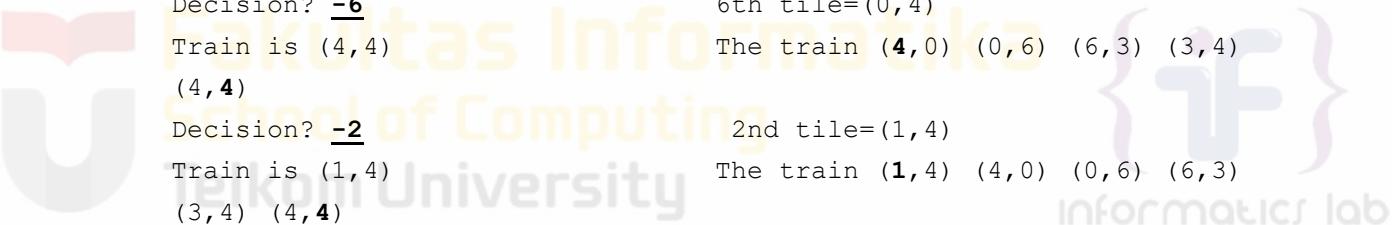
Nilai yang diperoleh pemain adalah berapa banyak kartu yang dia bisa tempatkan dalam rantai kartu, 0..7. Maksimum nilai per ronde adalah 7.

Program anda harus dapat:

1. Mengocok kartu
2. Mencatat nilai dan jumlah ronde yang dimainkan
3. Pada setiap ronde, membagikan 7 kartu dan membuka satu kartu dari tumpukan untuk memulai permainan.
4. Memeriksa kartu pilihan pemain (-7..-1, 0, 1, ..7, dan 9) dimana -1 .. -7 untuk meletakkan kartu diujung kiri rantai, 1..7 untuk ujung kanan rantai, 0 untuk selesai ronde, 9 untuk selesai permainan

```
Welcome to the game of Gaple
Your score is 0/0 At the beginning scores=0, games=0
Dealing ... Shuffle the tiles
Your tiles: (3,6) (1,4) (5,5) (2,5) (5,6) (2,3) (3,4)
Starting train is (0,4)
Decision? +7 7th/last tile=(3,4) put on right
end
Train is (0,3) The train (0,4) (4,3), or just
(0,3)
Decision? u1 1st tile=(3,6)
Train is (0,6) The train (0,4) (4,3) (3,6), or
just (0,6)
Decision? +5 5th tile=(5,6)
Train is (0,5) The train (0,4) (4,3) (3,6) (6,5),
or just (0,5)
Decision? +3 3rd tile=(5,5)
```

Train is (0,5)  
 (5,5)  
 Decision? **+4**  
 Train is (0,2)  
 (5,5) (5,2)  
 Decision? **+6**  
 Train is (0,3)  
 (0,4) (4,3) (3,6) (6,5) (5,5) (5,2) (2,3)  
 Decision? **0**  
 You earn 6 points  
 Your score is 6/1  
 Dealing ...  
 Your tiles: (3,5) (1,4) (0,1) (3,6) (4,4) (0,4) (0,6)  
 Starting train is (3,4)  
 Decision? **+5**  
 Train is (3,4)  
 Decision? **-4**  
 Train is (6,4)  
 Decision? **-7**  
 Train is (0,4)  
 Decision? **-6**  
 Train is (4,4)  
 (4,4)  
 Decision? **-2**  
 Train is (1,4)  
 (3,4) (4,4)  
 Decision? **-3**  
 Train is (0,4)  
 (0,1) (1,4) (4,0) (0,6) (6,3) (3,4) (4,4)  
 Decision? **0**  
 You earn 6 points  
 Your score is 12/2  
 Dealing ...  
 Your tiles: (5,6) (0,6) (4,6) (2,4) (4,5) (4,4) (6,6)  
 Starting train is (5,5)  
 Decision? **9**  
 Your last score is 12/2  
 Thank you for playing with us.  
 Your winning rate is 85.71%      which is total scores/ (#rounds\*7)


  
 Institut Teknologi Sepuluh Nopember  
 ITS Informatics Lab

