

# MODUL PRAKTIKUM

# ALGORITMA PEMROGRAMAN

## S1 INFORMATIKA



Published by school of computing

## LEMBAR PENGESAHAN

Saya yang bertanda tangan di bawah ini:

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

NIP : 19890017

Koordinator Mata Kuliah : Algoritma Pemrograman

Prodi : S1 Informatika

Menerangkan dengan sesungguhnya bahwa modul ini digunakan untuk pelaksanaan praktikum di Semester Genap Tahun Ajaran 2022/2023 di Laboratorium Informatika, Fakultas Informatika, Universitas Telkom.



Bandung, 20 Februari 2023



Mengesahkan,

Koordinator Mata Kuliah

Algoritma Pemrograman



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

NIP. 19890017

Mengetahui,

Kaprodi S1 Informatika



Dr. Erwin Budi Setiawan, S.Si., M.T.

NIP. 00760045

## DAFTAR ISI

<b>LEMBAR PENGESAHAN.....</b>	ii
<b>DAFTAR ISI .....</b>	iii
<b>DAFTAR GAMBAR.....</b>	vi
<b>PRAKATA.....</b>	vii
<b>BAB 1. PENGANTAR PRAKTIKUM.....</b>	1
<b>MODUL 1. RUNNING MODUL .....</b>	3
1.1. <b>Peraturan Praktikum Laboratorium Informatika .....</b>	3
1.2. <b>Penyampaian Keluhan Praktikum IFLAB Melalui iGracias .....</b>	4
1.3. <b>Penyampaian Keluhan Praktikum IFLAB Melalui Situs Web .....</b>	5
1.4. <b>Aturan Penulisan dan Submisi Tugas.....</b>	5
1.5. <b>Instalasi Program Penunjang Praktikum .....</b>	9
<b>MODUL 2. REVIEW PENGENALAN PEMROGRAMAN 1.....</b>	11
2.1. <b>Struktur Program Go .....</b>	11
2.2. <b>Tipe Data dan Instruksi Dasar .....</b>	14
2.3. <b>Struktur Kontrol Perulangan .....</b>	21
2.4. <b>Struktur Kontrol Percabangan .....</b>	27
<b>MODUL 3. REVIEW PENGENALAN PEMROGRAMAN 2.....</b>	32
3.1. <b>Soal Latihan Modul 3.....</b>	32
<b>BAB 2. PEMROGRAMAN MODULAR .....</b>	36
<b>MODUL 4. FUNGSI .....</b>	37
4.1. <b>Definisi Function.....</b>	37
4.2. <b>Deklarasi Function.....</b>	37
4.3. <b>Cara Pemanggilan Function.....</b>	38
4.4. <b>Contoh Program dengan Function .....</b>	39
4.5. <b>Soal Latihan Modul 4.....</b>	40
<b>MODUL 5. PROSEDUR .....</b>	43
5.1. <b>Definisi Procedure .....</b>	43
5.2. <b>Deklarasi Procedure .....</b>	43
5.3. <b>Cara Pemanggilan Procedure .....</b>	44
5.4. <b>Contoh Program dengan Procedure .....</b>	45
5.5. <b>Parameter .....</b>	46
5.6. <b>Soal Latihan Modul 5.....</b>	49
<b>MODUL 6. REKURSIF.....</b>	52

6.1.	Pengantar Rekursif .....	52
6.2.	Komponen Rekursif .....	55
6.4.	Soal Latihan Modul 6.....	57
	<b>MODUL 7. ASESMEN PRAKTIKUM 1 .....</b>	<b>59</b>
7.1.	Contoh Soal Asesmen Praktikum 1.....	59
	<b>MODUL 8. REVIEW ASESMEN PRAKTIKUM 1 .....</b>	<b>62</b>
8.1.	Review Soal Asesmen Praktikum 1.....	62
	<b>BAB 3. PENCARIAN PADA DATA TERSTRUKTUR.....</b>	<b>65</b>
	<b>MODUL 9. TIPE DATA TERSTRUKTUR.....</b>	<b>66</b>
9.1.	Tipe Bentukan.....	66
9.2.	Array .....	68
9.3.	Soal Latihan Modul 9.....	70
	<b>MODUL 10. PENCARIAN NILAI EKSTRIM.....</b>	<b>75</b>
10.1.	Ide Pencarian Nilai Ekstrim .....	75
10.2.	Pencarian Nilai Ekstrim pada Array Bertipe Data Dasar .....	75
10.3.	Pencarian Nilai Ekstrim pada Array Bertipe Data Bentukan.....	76
10.4.	Soal Latihan Modul 10.....	78
	<b>MODUL 11. PENCARIAN NILAI TERTENTU .....</b>	<b>80</b>
11.1.	Pencarian Sekuensial.....	80
11.2.	Pencarian Biner/Belah Tengah .....	82
11.3.	Pencarian pada Array Bertipe Data Bentukan .....	84
11.4.	Soal Latihan Modul 11.....	85
	<b>MODUL 12. ASESMEN PRAKTIKUM 2 .....</b>	<b>88</b>
12.1.	Contoh Soal Asesmen Praktikum 2.....	88
12.2.	Review Asesmen Praktikum 2.....	90
	<b>BAB 4. PENGURUTAN PADA DATA TERSTRUKTUR.....</b>	<b>97</b>
	<b>MODUL 13. PENGURUTAN DATA SECARA SELEKSI .....</b>	<b>98</b>
13.1.	Ide Algoritma Pencarian Seleksi .....	98
13.2.	Algoritma Selection Sort .....	98
13.3.	Soal Latihan Modul 13.....	100
	<b>MODUL 14. PENGURUTAN DATA SECARA INSERSI .....</b>	<b>103</b>
14.1.	Ide Algoritma Pencarian Inversi .....	103
14.2.	Algoritma Insertion Sort.....	103
14.3.	Soal Latihan Modul 14.....	105
	<b>MODUL 15. ASESMEN PRAKTIKUM 3 .....</b>	<b>107</b>
15.1.	Contoh Soal Asesmen Praktikum 3.....	107

<b>MODUL 16. REVIEW ASESMEN PRAKTIKUM 3 .....</b>	111
<b>16.1. Review Soal Asesmen Praktikum 3.....</b>	111
<b>BAB 5. PENGAYAAN .....</b>	117
<b>MODUL 17. SKEMA PEMROSESAN SEKUENSIAL .....</b>	118
<b>17.1. Pengantar Skema Pemrosesan Sekuensial.....</b>	118
<b>17.2. Pembacaan Data Tanpa Marker pada Akhir Rangkaian Data.....</b>	118
<b>17.3. Pembacaan Data Dengan Marker pada Akhir Rangkaian Data .....</b>	118
<b>17.4. Kemungkinan Rangkaian Data Kosong, Hanya Ada Marker .....</b>	119
<b>17.5. Elemen Pertama Perlu Diproses Tersendiri/Kasus Khusus .....</b>	120
<b>17.6. Soal Latihan Modul 17.....</b>	121
<b>MODUL 18. MESIN ABSTRAK .....</b>	125
<b>18.1. Pengantar Mesin Abstrak.....</b>	125
<b>18.2. Contoh Kasus Mesin Domino .....</b>	125
<b>18.3. Soal Latihan Modul 18.....</b>	126
<b>DAFTAR PUSTAKA.....</b>	128
<b>LAMPIRAN .....</b>	129
<b>A. Paket-Paket Penting .....</b>	129
<b>1. Paket "math" .....</b>	129
<b>2. Paket "math/rand" .....</b>	130
<b>3. Paket "time" .....</b>	130
<b>4. Paket "os" .....</b>	131
<b>5. Paket "fmt" .....</b>	132
<b>6. Paket "log" .....</b>	135
<b>7. Paket "io" .....</b>	135
<b>8. Paket "net/http" .....</b>	136
<b>9. Paket "strconv" .....</b>	137
<b>10. Paket "strings" .....</b>	138
<b>11. Paket "archive/zip" .....</b>	139
<b>12. Paket "testing" .....</b>	140
<b>B. Perintah Go .....</b>	142
<b>C. Instalasi Go Perkakas .....</b>	143
<b>D. Instalasi Paket Pihak ke-3 (Contoh paket GUI).....</b>	143
<b>E. Contoh Program .....</b>	144

## **DAFTAR GAMBAR**

Gambar 1. Ilustrasi proses forward dan backward pada saat rekursif.....	54
Gambar 2. Ilustrasi denah daerah yang digunakan untuk mengukur curah hujan .....	122



## PRAKATA

Buku modul praktikum Algoritma Pemrograman ini merupakan panduan mahasiswa dalam pelaksanaan praktikum dalam mata kuliah Algoritma Pemrograman di Fakultas Informatika, Universitas Telkom. Buku modul ini berisi materi, contoh soal dan kumpulan soal-soal latihan yang bisa digunakan mahasiswa untuk belajar sebelum pelaksanaan praktikum dimulai. Bahasa pemrograman yang digunakan adalah Go Language (Go Lang). Materi dalam buku modul praktikum ini melanjutkan materi pada mata kuliah Pengenalan Pemrograman, sehingga pembahasan terkait dasar pemrograman dengan Go, struktur kontrol perulangan dan percabangan tidak lagi dibahas lebih detail pada modul ini, karena sudah diberikan pada mata kuliah sebelumnya.

Buku modul praktikum ini dibagi ke dalam lima bab, yaitu: (1) Pengantar Praktikum; (2) Pemrograman Modular; (3) Pencarian pada Data Terstruktur; (4) Pengurutan pada Data Tersetruktur; dan (5) Pengayaan. Selanjutnya pada masing-masing bab akan berisi beberapa modul praktikum berkaitan dengan babnya, dengan total terdapat 16 modul praktikum utama. Setiap pekan praktikum, mahasiswa akan melaksanakan kegiatan praktikum sesuai dengan urutan modul yang terdapat pada buku modul praktikum ini. Walaupun buku modul ini berisi materi yang mengajarkan algoritma dengan pendekatan praktik pemrograman dengan Go Lang, tetapi buku modul ini tidak bisa dijadikan sebagai referensi utama untuk pembelajaran pemrograman yang lebih luas. Hal ini dikarenakan fokus kedalaman materi dan tingkat kesulitan soal yang diberikan menyesuaikan pada course learning outcome (CLO) pada rencana pembelajaran semester yang telah ditetapkan pada mata kuliah Algoritma Pemrograman di Fakultas Informatika, Universitas Telkom.

Buku modul ini diharapkan bisa memberikan manfaat kepada mahasiswa atau praktikan dalam menguasai materi yang diberikan pada mata kuliah Algoritma Pemrograman.

Bandung, 1 Februari 2023

Tim Penulis

## BAB 1. PENGANTAR PRAKTIKUM

Praktikum Algoritma Pemrograman merupakan salah satu aktivitas dalam rangkaian perkuliahan pada mata kuliah Algoritma Pemrograman. Oleh karena itu susunan materi pada modul ini dibuat sedemikian rupa sehingga *inline* dengan materi yang diajarkan di kelas teori. Penilaian pada praktikum Algoritma Pemrograman ini menjadi salah satu komponen penilaian pada CLO-4 *Mahasiswa mampu mengimplementasikan solusi algoritmik dengan tepat kedalam suatu bahasa pemrograman.*

Terdapat tiga jenis penugasan pada kegiatan praktikum ini, yaitu:

- a) **Tugas Pendahuluan (TP)**, merupakan tugas yang diberikan sebelum praktikum di mulai, estimasi penggerjaan tugas ini adalah 60 menit di luar perkuliahan/praktikum. Tugas pendahuluan ini biasanya diumumkan di akhir pekan dan dikumpulkan pada hari senin pagi.
- b) **Tugas Jurnal**, merupakan tugas utama praktikum dengan estimasi penggerjaan selama 100-120 menit. Pada saat penggerjaan jurnal ini, asisten praktikum akan membimbing praktikan dalam mengerjakan tugas dan asisten juga akan memberikan tutorial pada beberapa contoh soal. Praktikan diperbolehkan bertanya terkait materi, ide solusi, dan notasi pemrograman.
- c) **Asesmen Praktikum**, merupakan sesi di mana praktikan mengerjakan soal secara mandiri tanpa bantuan dari asisten praktikum. Pertanyaan hanya boleh terait hal teknis, kejelasan soal dan notasi pemrograman. Estimasi penggerjaan selama 100-120 menit.

Adapun rencana agenda praktikum selama satu semester ini dapat dilihat pada tabel\* berikut:

N o	Modul Praktikum	TP	Jurnal	Asesmen
1	Running Modul	-	-	-
2	Review Pengenalan Pemrograman 1	✓	✓	-
3	Review Pengenalan Pemrograman 2	✓	✓	-
4	Fungsi	✓	✓	-
5	Prosedur	✓	✓	-
6	Rekursif	✓	✓	-
7	Asesmen Praktikum 1	✓	-	✓
8	Review Asesmen Praktikum 1	-	-	-
9	Tipe Data Terstruktur	✓	✓	-
10	Pencarian Nilai Ekstrim	✓	✓	-
11	Pencarian Nilai Tertentu	✓	✓	-
12	Asesmen Praktikum 2	✓	-	✓
13	Pengurutan Data Secara Seleksi	✓	✓	-
14	Pengurutan Data Secara Inversi	✓	✓	-
15	Asesmen Praktikum 3	✓	-	✓
16	Review Asesmen Praktikum 3	-	-	-

\* susunan materi dan penugasan bisa berubah sesuai dengan kondisi pelaksanaan praktikum.

Selain itu pada bab 1 ini berisi tiga modul, yaitu:

a) **Modul 1. Running Modul**

Pada running modul ini akan dijelaskan terkait tata tertib pelaksanaan praktikum, mulai dari peraturan praktikum, cara penyampaian keluhan, aturan penulisan dan submisi tugas hingga cara instalasi beberapa aplikasi penunjang praktikum.

b) **Modul 2. Review Pengenalan Pemrograman 1**

Pada modul ini akan di bahas kembali terkait materi yang sudah dipelajari pada mata kuliah pengenalan pemrograman, tujuannya untuk *refresh* dan *review* materi yang sudah pernah dipelajari sebelumnya. Materi yang dibahas adalah tipe data, instruksi dasar, struktur kontrol perulangan dan percabangan.

c) **Modul 3. Review Pengenalan Pemrograman 2.**

Modul 3 ini merupakan kelanjutan dari modul 2, yang mana berisi soal-soal latihan untuk kasus-kasus yang memanfaatkan kombinasi struktur kontrol perulangan dan percabangan.



## MODUL 1. RUNNING MODUL

### 1.1. Peraturan Praktikum Laboratorium Informatika

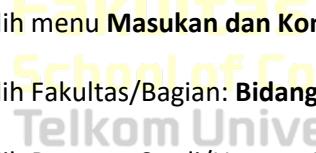
- 1) Praktikum diampu oleh dosen kelas dan dibantu oleh asisten laboratorium dan asisten praktikum.
- 2) Praktikum dilaksanakan di Gedung Telkom University Lanmark Tower (TULT) Lantai 6 dan Lantai 7 sesuai jadwal yang ditentukan.
- 3) Praktikan wajib membawa modul praktikum, KTM, dan alat tulis.
- 4) Praktikan wajib mengisi daftar hadir *rooster* dan BAP praktikum dengan bolpoin bertinta hitam.
- 5) Durasi kegiatan praktikum S-1 = 2 jam (100 menit).
- 6) Jumlah pertemuan praktikum sebanyak 16 kali.
- 7) Praktikan wajib hadir minimal 75% dari seluruh pertemuan praktikum di lab. Jika total kehadiran kurang dari 75% maka nilai UAS/ Tugas Besar = 0.
- 8) Praktikan yang datang terlambat :
  - a. Keterlambatan kurang atau sama dengan 5 menit diperbolehkan mengikuti praktikum tanpa tambahan praktikum.
  - b. Keterlambatan lebih dari 5 menit tidak diperbolehkan mengikuti praktikum.
- 9) Saat praktikum berlangsung, asisten praktikum dan praktikan:
  - a. Wajib menggunakan seragam sesuai aturan institusi.
  - b. Wajib mematikan/ mengkondisikan semua alat komunikasi.
  - c. Dilarang membuka aplikasi yang tidak berhubungan dengan praktikum yang berlangsung.
  - d. Dilarang mengubah pengaturan *software* maupun *hardware* komputer tanpa ijin.
  - e. Dilarang membawa makanan maupun minuman di ruang praktikum.
  - f. Dilarang memberikan jawaban ke praktikan lain.
  - g. Dilarang menyebarkan soal praktikum.
  - h. Dilarang membuang sampah di ruangan praktikum.

- i. Wajib meletakkan alas kaki dengan rapi pada tempat yang telah disediakan.

10) Setiap praktikan dapat mengikuti praktikum susulan maksimal dua modul untuk satu mata kuliah praktikum.

  - Praktikan yang dapat mengikuti praktikum susulan hanyalah praktikan yang memenuhi syarat sesuai ketentuan institusi, yaitu: sakit (dibuktikan dengan surat keterangan medis), tugas dari institusi (dibuktikan dengan surat dinas atau dispensasi dari institusi), atau mendapat musibah atau kedukaan (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 tidak dapat diganggu gugat.

## **1.2. 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/St**
  - 5) Pilih Layanan: **Praktikum**
  - 6) Pilih Kategori: **Pelaksanaan Praktikum**, lalu pilih **Sub Kategori**.
  - 7) Isi **Deskripsi** sesuai komplain yang ingin disampaikan.



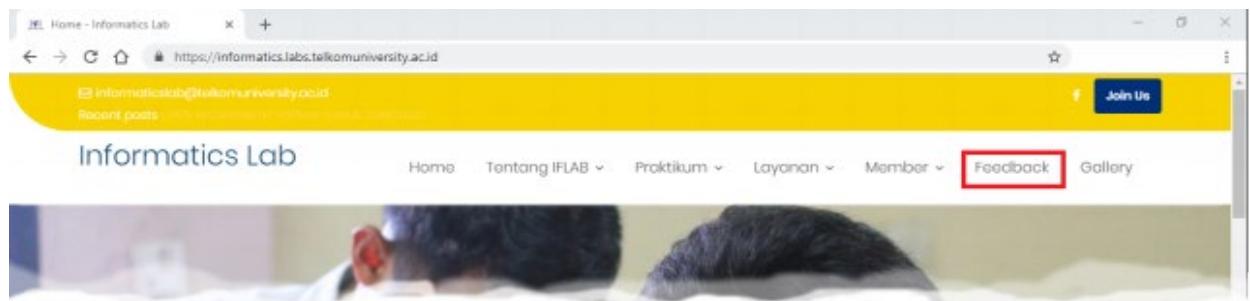
**Input Keluhan**

Fakultas / Bagian :	BIDANG AKADEMIK (FIF)
Program Studi / Urusan :	URUSAN LABORATORIUM/BENGKEL/STUDIO (FIF)
Pelapor :	RIZQILLAH ZAHRA LESTARI
Penanganan :	PRAKTIKUM
Kategori :	Pelaksanaan Praktikum
Sub Kategori :	Please Select...
Tipe Masukan :	<input checked="" type="radio"/> Komplain <input type="radio"/> Masukan
	
Deskripsi :	<p> </p>

- 8) Lampirkan file jika perlu. Lalu klik Kirim.

### 1.3. 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.

A screenshot of a web browser displaying the 'Feedback - Informatics Lab' page. The URL in the address bar is https://informatics.labs.telkomuniversity.ac.id/feedback/. The page has a yellow header with the text 'Informatics Lab' and navigation links: Home, Tentang IFLAB, Praktikum, Layanan, Member, Feedback (highlighted with a red box), and Gallery. The main content area features a large image of a person's hands working on a computer keyboard. Below the image is a section titled 'FEEDBACK' with the following text: 'Layanan feedback ini dibuat sebagai sarana untuk menyampaikan keluhan atau berdiskusi untuk memajukan Laboratorium Informatika. Berikut link untuk pengisian feedback. Untuk melakukan pengisian feedback wajib menggunakan akun SSO Telkom University terlebih dahulu.' A blue button labeled 'Link Form Feedback' is shown, with a red box highlighting it.

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

### 1.4. Aturan Penulisan dan Submisi 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 di dalam struktur while-loop, harus masuk 4 spasi ke dalam dibandingkan dengan struktur kontrolnya.

Gunakan perintah **go fmt** sebelum submisi untuk memastikan program Anda mempunyai struktur yang konsisten. Pada contoh di bawah spasi digambarkan dengan simbol █ agar lebih jelas.

Gunakan perintah **go clean** sebelum submisi untuk membuang semua file yang tidak diperlukan di dalam 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.
- 1) **JANGAN** menggunakan satu huruf (**a, b, c, ...**) dari alfabet untuk nama variabel bagi semua kebutuhan
  - 2) **JANGAN** menggunakan kata kunci sebagai bagian dari nama variabel, seperti: **\_true, \_false, \_for, forloop, ...**
  - 3) Pilih nama variabel yang berhubungan dengan nilai yang disimpan dalam variabel tersebut, seperti: **jumlah, rerata, terbesar, ganjil, ketemu, mhs, skor, min, max, ...**
  - 4) 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

5) 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 diekspor oleh paket.

Contoh: **nilaiTerkecil**, **mhsBaru**, **mhsLama**, ...

- Dengan **\_underscore**: pada nama berbentuk kata majemuk, kata-kata digabungkan 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 AP **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.

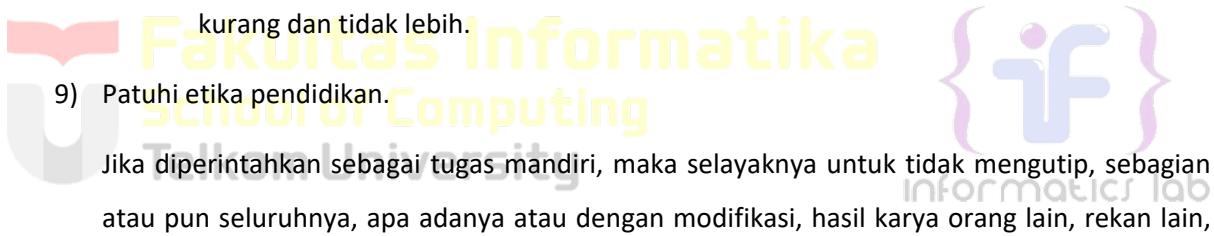
- a. Sekuens instruksi dimulai dari instruksi pertama, eksekusi secara berurutan, dan berakhir pada satu instruksi terakhir.
- b. Bentuk pengulangan dimulai dari instruksi pertama, masuk ke dalam dan mengulang iterasi badan loop, dan kemudian keluar dari **SATU** lokasi saja, yaitu dari kondisi iterasi 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** di dalam 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 di atas.
- 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**:
- 1) Tipe **integer**, tetapi nilai yang dipakai hanya 0 dan 1 (atau sejenisnya), di mana manfaatnya adalah untuk menyatakan suatu keadaan (ada/tidak ada, ketemu/tidak ketemu), dsb.. Tipe yang tepat yang sebaiknya digunakan sudah jelas adalah tipe **Boolean**.
  - 2) 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>
<code>abc ← def * -1</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 di berbagai 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 diperlukan 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.

## 1.5. Instalasi Program Penunjang Praktikum

Kode program dengan bahasa pemrograman Go tidak bisa langsung dijalankan oleh komputer, perlu diterjemahkan terlebih dahulu menjadi sebuah program yang dimengerti oleh komputer. Proses penerjemahan ini dikenal dengan istilah kompilasi, di mana pada proses ini kode program dicek struktur atau tata bahasanya, kemudian diterjemahkan menjadi suatu program dengan bahasa yang dimengerti oleh komputer yaitu bahasa mesin. Program yang digunakan untuk menerjemahkan ini disebut Compiler, artinya setiap bahasa pemrograman memiliki compiler-nya masing-masing.

Selain compiler diperlukan juga sebuah program teks editor khusus atau yang biasa disebut **Integrated Development Environment (IDE)**. Program ini digunakan untuk menuliskan kode

program, dan biasanya sudah dilengkapi fitur-fitur yang membantu programmer dalam membuat suatu program.

### **(1) Instalasi Go Languange (Golang)**

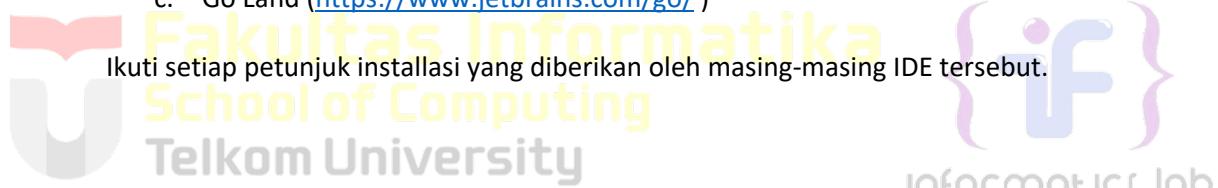
Compiler dari program Go dapat diunduh pada tautan berikut <https://go.dev/dl/>, dan proses installasi dapat dilakukan dengan mengikuti petunjuk yang terdapat pada tautan berikut <https://go.dev/doc/install>. Kemudian pilih sesuai dengan sistem operasi yang digunakan oleh komputer.

Video tutorial instalasi Go: <https://youtu.be/k46ZjkT5-Rc>

### **(2) Instalasi Integrated Development Environment (IDE)**

Pada pemrograman terdapat banyak sekali IDE yang bisa digunakan, berikut ini adalah beberapa IDE yang umum digunakan untuk pemula.

- a. Notepad++ (<https://notepad-plus-plus.org/downloads/>)
- b. Visual Studio Code (<https://code.visualstudio.com/>)
- c. Go Land (<https://www.jetbrains.com/go/>)



Ikuti setiap petunjuk installasi yang diberikan oleh masing-masing IDE tersebut.

## MODUL 2. REVIEW PENGENALAN PEMROGRAMAN 1

### 2.1. 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 di mana saja di dalam 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 '\*/'.

```
1 // Setiap program utama dimulai dengan "package main"
2 package main
3
4 // Impor paket yang dibutuhkan, "fmt" berisi proses I/O standar
5 import "fmt"
6
7 // Kode program utama dalam "fungsi main"
8 func main() {
9     ...
10 }
```

Contoh sebuah program dalam bahasa pemrograman Go (nama file hello.go).

```
1 package main
2 import "fmt"
3 func main() {
4     var greetings = "Selamat datang di dunia DAP"
5     var a, b int
6
7     fmt.Println(greetings)
8     fmt.Scanln(&a, &b)
9     fmt.Printf("%v + %v = %v\n", a, b, a+b)
10 }
```

```
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) 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 ke dalam (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 eksekutable tersebut dari terminal yang sama. Jangan memanggil program tersebut dengan mengklik eksekutable 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.

Video tutorial memulai program Go: <https://youtu.be/S5Rt2qO2QEg>

### 2) Latihan

1. Selidiki bahasa-bahasa pemrograman berikut, apakah termasuk diinterpretasi, dikompilasi, dikompilasi (ke instruksi perantara) kemudian diinterpretasi:

- Pascal
- C dan C++
- Java
- Python

2. Instal kompilator Go di komputer yang Anda gunakan. kemudian salin contoh program di atas 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 di dalamnya. 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).

## 2.2. Tipe Data dan Instruksi Dasar

### 1) Data dan Variabel

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

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

Contoh: **ketemu, found, rerata, mhs1, data\_2, ...**

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	<b>false</b> dan <b>true</b>
karakter	byte //uint8 rune //int32	tabel ASCII/UTF-8 tabel UTF-16
string	string	

- 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.
- **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.

- **Informasi alamat** atau lokasi dari variabel dapat diperoleh dengan menambahkan prefiks **&** di depan nama variabel tersebut.

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

- Jika variabel berisi alamat memori, prefiks **\*** pada variabel tersebut akan memberikan nilai yang tersimpan dalam memori di mana 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 di mana variabel **found** berada, alias sama saja dengan menyebutkan langsung **found** 8=).

- Operasi yang dapat dilakukan terhadap tipe data di atas 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)
&   ^ & <sup>^</sup>	integer	operasi <b>per-bit</b> 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 <b>Boolean</b> AND, OR, dan NOT
* &	variabel apasaja	mendapatkan data dari lokasi memori dan mendapatkan lokasi dari variabel

Contoh:

Operasi	Hasil
"non suffi" + "cit mundo"	"non sufficit mundo"
2019.01 + 1.0102	2020.0202
2020 / 20	22.22
20.2 * 1.1	101
2020 % 1999	21
2020 & 1111	2104
2020 ^ 1111	1663
2020 >> 2	505
"minutus" < "magnus"	false
2020 >= 1234	true
! false && true	true

- Bahasa Go menganut kesesuaian tipe data yang ketat. Tipe data yang berbeda tidak boleh dicampur dalam satu ekspresi, bahkan tipe data masih yang sejenis, misalnya masih sama-sama integer (**int** dan **int32**). Untuk menyesuaikan tipe data, ada beberapa cara yang dapat dilakukan:

- ✓ Casting, **tipe(data)**, mengubah tipe dari data yang diberikan ke tipe yang diinginkan.
- ✓ Memanfaatkan fungsi **Sprint** dan **Sscan** dari paket **fmt**.
- ✓ Memanfaatkan fungsi-fungsi dalam paket **strconv**, seperti **Atoi**, **Itoa**, dan **ParseBool**.

Lihat lampiran untuk contoh penggunaan.

Contoh:

Operasi	Hasil
<code>2020.0 % 19</code>	will be an illegal expression error
<code>int(2020.0) % 19</code>	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 di sebelah kiri (MSB)
	real	real	format data tidak berubah, hanya penyesuaian jumlah bit. Kekurangan bit, maka bit mantisa diisi bit 0.
	real	integer	format data disesuaikan dengan tipe data tujuan
	integer	real	format data disesuaikan dengan 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", &amp;v)</code>	string	any type	baca string ke variabel dengan tipe tertentu
<code>fmt.Sscanf(s, "%c", &amp;v)</code>	string	karakter	baca string ke variabel bertipe karakter

- 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 ekuivalen 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
kamus a : tipe	var a tipe	a diinisialisasi dengan nilai default
kamus a : tipe  algoritma a <- nilai_awal	var a tipe = nilai_awal var a = (tipe)nilai_awal	a diinisialisasi dengan nilai_awal
	a := nilai_awal a := (tipe)nilai_awal	secara <b>implisit</b> , tipe variabel a ditentukan dari nilai inisialisasinya

Contoh:



```

1 func main() {
2     var a int
3     a = 2019
4     r := 2019.0707
5     b := false
6     c := 'x'
7     s := "a string is a string"
8 }
```

## 2) Instruksi Dasar

Notasi instruksi dasar	Penulisan dalam bahasa Go	Keterangan
v1 <- e1 v1 <- v1 + e1 v1 <- v1 - e1 v1 <- v1 + 1 v1 <- v1 - 1	v1 = e1 v1 += e1 // atau v1 = v1 + e1 v1 -= e1 // atau v1 = v1 - e1 v1++ // atau v1 = v1 + 1 v1-- // atau v1 = v1 - 1	operasi assignment, mengisi data ke lokasi memori (variabel)
input(v1, v2)	fmt.Scan( &v1, &v2 ) fmt.Scanln( &v1, &v2 ) fmt.Scanf( "%v %v", &v1, &v2 )	Pembacaan data memerlukan alamat memori ke mana 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 ditulis.

Contoh:

```
1 package main
2 import "fmt"
3
4 func main() {
5     var a, b, c float64
6     var hipotenusa bool
7
8     fmt.Scanln( &a, &b, &c )
9     hipotenusa = (c*c) == (a*a + b*b)
10    fmt.Println( "Sisi c adalah hipotenusa segitiga a,b,c: ", hipotenusa)
11 }
```

### 3) Konstanta Simbolik

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

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

### 4) Soal Latihan Modul 2A

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?

```
1 package main
2 import "fmt"
3
4 func main() {
5     var (
6         satu, dua, tiga string
7         temp string
8     )
9     fmt.Print("Masukan input string: ")
10    fmt.Scanln(&satu)
11    fmt.Print("Masukan input string: ")
12    fmt.Scanln(&dua)
13    fmt.Print("Masukan input string: ")
14    fmt.Scanln(&tiga)
15    fmt.Println("Output awal = " + satu + " " + dua + " " + tiga)
16    temp = satu
17    satu = dua
18    dua = tiga
19    tiga = temp
20    fmt.Println("Output akhir = " + satu + " " + dua + " " + tiga)
21 }
```

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):

1	Tahun: <b>2016</b> Kabisat: true
2	Tahun: <b>2000</b> Kabisat: true
3	Tahun: <b>2018</b> 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  $luas_{bola} = 4\pi r^2$ . Di mana  $\pi \approx 3.1415926535$ .

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

Jejari = <b>5</b> Bola dengan jejari 5 memiliki volume 523.5988 dan luas kulit 314.1593
--



4. Dibaca nilai temperatur dalam derajat Celsius. 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 Celsius: <b>50</b> Derajat Fahrenheit: 122
--

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

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

Temperatur Celsius: <b>50</b> Derajat Reamur: 40 Derajat Fahrenheit: 122 Derajat Kelvin: 323
---

5. Tipe karakter sebenarnya hanya apa yang tampak dalam tampilan. Di dalamnya 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.



### 2.3. Struktur Kontrol Perulangan

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

#### Bentuk pengulangan dalam bahasa Go

```
1 for inisialisasi; kondisi; update {
2     // .. for-loop ala C
3     // .. ke-3 bagian opsional, tetapi ";" tetap harus ada
4 }
5 for kondisi {
6     // .. ulangi kode di sini selama kondisi terpenuhi
7     // .. sama seperti "for ; kondisi; {"
8 }
9 for {
10    // .. tanpa kondisi, berarti loop tanpa henti (perlu if-break)
11 }
12 for ndx, var := range slice/array {
13     // .. iterator mengunjungi seluruh isi slice/array
14     // .. pada setiap iterasi ndx diset indeks dan var diisi nilainya
15 }
```

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 di mana 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.

#### 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
1 2 3 4	while (kondisi) do ... kode yang diulang endwhile	for kondisi { ... kode yang diulang }

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

	<b>Notasi algoritma</b>	<b>Penulisan dalam bahasa Go</b>
1	<code>e &lt;- 0.0000001</code>	<code>e := 0.0000001</code>
2	<code>x &lt;- 2.0</code>	<code>x := 2.0</code>
3	<code>y &lt;- 0.0</code>	<code>y := 0.0</code>
4	<code>y1 &lt;- x</code>	<code>y1 := x</code>
5	<code>while y1-y &gt; e or y1-y &lt; -e do</code>	<code>for y1-y &gt; e    y1-y &lt; -e {</code>
6	<code>    y &lt;- y1</code>	<code>    y = y1</code>
7	<code>    y1 &lt;- 0.5*y + 0.5*(x/y)</code>	<code>    y1 = 0.5*y + 0.5*(x/y)</code>
8	<code>endwhile</code>	<code>}</code>
9	<code>output("sqrt(",x,",")=",y)</code>	<code>fmt.Printf( "sqrt(%v)=%v\n", x, y )</code>

## 2) Bentuk Repeat-Until

Bentuk repeat-until di mana 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!

	<b>Notasi Algoritma</b>	<b>Penulisan dalam bahasa Go</b>
1	<code>repeat</code>	<code>for selesai:=false; !selesai; {</code>
2	<code>.. kode yang diulang</code>	<code>.. kode yang diulang</code>
3	<code>until (kondisi)</code>	<code>selesai = kondisi</code>
4		<code>}</code>
5		<code>for selesai:=false; !selesai;</code>
6		<code>selesai=kondisi {</code>
7		<code>.. kode yang diulang</code>
8		<code>}</code>
9		

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

	<b>Notasi Algoritma</b>	<b>Penulisan dalam bahasa Go</b>
1	<code>maxF &lt;- 100</code>	<code>maxF := 100</code>
2	<code>f0 &lt;- 0</code>	<code>f0 := 0</code>
3	<code>f1 &lt;- 1</code>	<code>f1 := 1</code>
4	<code>f2 &lt;- 1</code>	<code>f2 := 1</code>
5	<code>output("Bilangan pertama:", f1 )</code>	<code>fmt.Println("Bilangan pertama:", f1)</code>
6	<code>repeat</code>	<code>for selesai:=false; !selesai; {</code>
7	<code>    f0 &lt;- f1</code>	<code>    f0 = f1</code>
8	<code>    f1 &lt;- f2</code>	<code>    f1 = f2</code>
9	<code>    f2 &lt;- f1 + f0</code>	<code>    f2 = f1 + f0</code>
10	<code>    output("Bilangan</code>	<code>    fmt.Println("Bilangan berikutnya:", f1)</code>
11	<code>berikutnya:", f1)</code>	<code>    selesai = f2 &gt; maxF</code>
12	<code>until f2 &gt; maxF</code>	<code>}</code>

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

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

```
1 maxF := 100
2 f0 := 0
3 f1 := 1
4 f2 := 1
5 fmt.Println("Bilangan pertama:", f1 )
6 for f2 <= maxF {
7     f0 = f1
8     f1 = f2
9     f2 = f1 + f0
10    fmt.Println("Bilangan berikutnya:", f1 )
11 }
```

### 3) Soal Latihan Modul 2B

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 di bawah 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 ke dalam pita.

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

Tampilkan isi pita setelah proses input selesai.

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

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

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

Perhatikan contoh sesi interaksi program seperti di bawah 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 di kiri-kanan motor. Sepeda motor tidak akan oleng jika selisih berat barang di kedua kantong sisi tidak lebih dari 9 kg.

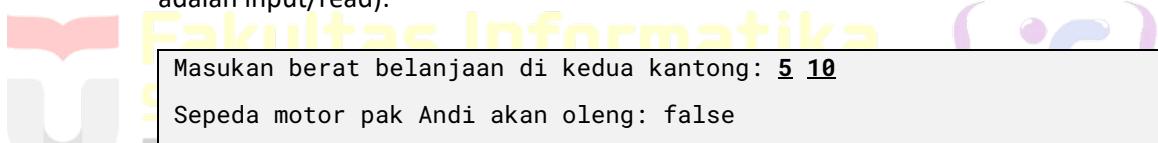
Buatlah program Pak Andi yang menerima input dua buah bilangan real positif yang menyatakan berat total masing-masing isi kantong terpal. Program akan terus meminta input bilangan tersebut hingga salah satu kantong terpal berisi 9 kg atau lebih.

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

```
Masukan berat belanjaan di kedua kantong: 5.5 1.0
Masukan berat belanjaan di kedua kantong: 7.1 8.5
Masukan berat belanjaan di kedua kantong: 2 6
Masukan berat belanjaan di kedua kantong: 9 5.8
Proses selesai.
```

Modifikasi program tersebut, di mana 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 di bawah ini (**teks bergaris bawah** adalah input/read):



```
Masukan berat belanjaan di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: 55.6 70.2
Sepeda motor pak Andi akan oleng: true
Masukan berat belanjaan di kedua kantong: 72.3 66.9
Sepeda motor pak Andi akan oleng: false
Masukan berat belanjaan di kedua kantong: 59.5 98.7
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 di atas.

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

Nilai K = 100

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 di bawah ini (**teks bergaris bawah** adalah input/read):

1	Nilai K = <u>10</u> Nilai akar 2 = 1.4062058441
2	Nilai K = <u>100</u> Nilai akar 2 = 1.4133387072
3	Nilai K = <u>1000</u> Nilai akar 2 = 1.4141252651



School of Computing  
Telkom University



## 2.4. Struktur Kontrol Percabangan

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

### 1) Bentuk If-Else

Berikut ini bentuk-bentuk **if-else** yang mungkin dilakukan dalam bahasa Go. Semua bentuk di bawah 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
1	if (kondisi) then .. kode untuk kondisi true endif	if kondisi { .. kode untuk kondisi true }
4	if (kondisi) then .. kode untuk kondisi true else .. kode untuk kondisi false endif	if kondisi { .. kode untuk kondisi true } else { .. kode untuk kondisi false }
9	if (kondisi-1) then .. kode untuk kondisi-1 true 11 else if (kondisi-2) then 12 .. kode untuk kondisi-2 true 13 .. dst. dst. 14 else 15 .. kode jika semua kondisi 16 .. di atas false 17 endif	if kondisi_1 { .. kode untuk kondisi_1 true } else if kondisi_2 { .. kode untuk kondisi_2 true .. dst. dst. } else { .. kode jika semua kondisi .. di atas false }

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

```
1 if nilai > 75 && adaTubes {  
2     indeks = 'A'  
3 } else if nilai > 65 {  
4     indeks = 'B'  
5 } else if nilai > 50 && pctHadir > 0.7 {  
6     indeks = 'C'  
7 } else {  
8     indeks = 'F'  
9 }  
10 fmt.Printf( "Nilai %v dengan kehadiran %v% dan buat tubes=%v, mendapat  
11 indeks %c\n", nilai, pctHadir, adaTubes, indeks )
```

## 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
1 2 3 4 5 6 7 8 9 10	<pre> 1 depend on expresi 2     nilai_1: 3         .. kode jika ekspresi bernilai_1 4     nilai_2: 5         .. kode jika ekspresi bernilai_2 6         .. dst. dst. 7 } 8 9 10 </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>
11 12 13 14 15 16 17 18 19 20	<pre> 11 depend on (daftar variabel) 12     kondisi_1: 13         .. kode jika ekspresi_1 true 14     kondisi_2: 15         .. kode jika ekspresi_2 true 16         .. dst. dst. 17 } 18 19 20 </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:

```

1 switch indeks {
2     case 'A':
3         batasA = 100
4         batasB = 75
5     case 'B':
6         batasA = 75
7         batasB = 65
8     case 'C':
9         batasA = 65
10        batasB = 50
11    default:
12        batasA = 50
13        batasB = 0
14    }
15 fmt.Printf( "Rentang nilai %v adalah: %v..%v\n", indeks, batasB, batasA )

```

```

16 switch {
17     case nilai > 75 && adaTubes:
18         indeks = 'A'

```

```

19 case nilai > 65:
20     indeks = 'B'
21 case nilai > 50 && pctHadir > 0.7:
22     indeks = 'C'
23 default:
24     indeks = 'F'
25 }
26 fmt.Printf( "Nilai %v dengan kehadiran %v% dan buat tubes=%v, mendapat
27 indeks %c\n", nilai, pctHadir, adaTubes, indeks )

```

### 3) Soal Latihan Modul 2C

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

Dari berat parsel (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 di bawah ini (teks bergaris bawah adalah input/read):

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

- 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.

```

1 package main
2 import "fmt"
3 func main() {
4     var nam float64
5     var nmk string
6     fmt.Println("Nilai akhir mata kuliah: ")
7     fmt.Scanln(&nam)
8     if nam > 80 {
9         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 saja 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**, di mana **b > 1**. Program harus dapat mencari dan menampilkan semua faktor dari bilangan tersebut!

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

Bilangan: <b>12</b>	Bilangan: <b>7</b>
Faktor: 1 2 3 4 6 12	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 di bawah ini (teks bergaris bawah adalah input/read):

Bilangan: <b>12</b>	Bilangan: <b>7</b>
Faktor: 1 2 3 4 6 12	Faktor: 1 7
Prima: false	Prima: true

## MODUL 3. REVIEW PENGENALAN PEMROGRAMAN 2

Pada modul ini berisi soal-soal untuk latihan yang memanfaatkan penggunaan kombinasi struktur kontrol perulangan dan juga percabangan.

### 3.1. Soal Latihan Modul 3

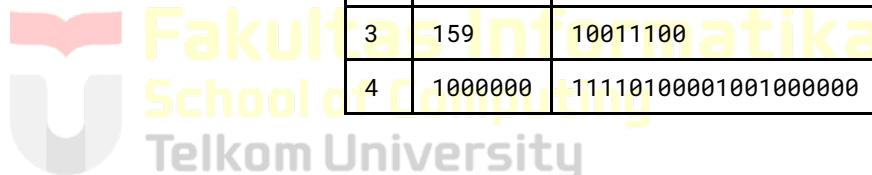
- 1) Buatlah program yang digunakan untuk konversi bilangan decimal (basis sepuluh) ke bilangan biner (basis dua).

**Masukan** terdiri dari sebuah bilangan bulat sebagai bilangan decimal yang akan dikonversi.

**Keluaran** berupa string biner hasil konversi.

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	1	1
2	7	111
3	159	10011100
4	1000000	11110100001001000000



- 2) Sebuah program digunakan untuk mencari daun yang paling lebar.

**Masukan** terdiri dari beberapa bilangan. Bilangan pertama adalah n yang menyatakan banyaknya daun. Kemudian n bilangan berikutnya adalah lebar dari masing-masing daun.

**Keluaran** berupa bilangan yang menyatakan lebar daun yang paling lebar.

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	3 2 5 2	5
2	1 10	10
3	5 9 5 8 30 1	30
4	10 11 32 53 64 85 96 57 38 29 80	96

- 3) Sebuah program digunakan untuk mencacah digit dari suatu bilangan.

**Masukan** terdiri dari sebuah bilangan bulat positif.

**Keluaran** berupa digit terbesar pada bilangan tersebut.

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	123	3
2	321	3
3	159	9
4	1000000	1

- 4) Sebuah program digunakan untuk mencari nilai digit tertentu

**Masukan** terdiri dari dua bilangan bulat positif  $x$  dan  $n$ , di mana nilai  $x \leq n$ .

**Keluaran** berupa boolean yang menyatakan apakah  $x$  terdapat di dalam  $n$ .

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	2 123	true
2	8 321478	true
3	4 159	false
4	5 1000000	false



- 5) Seorang mahasiswa sedang membuat aplikasi monitoring kendaraan yang dipasang pada gerbang tol Buahbatu. Terdapat tiga tipe kendaraan yang melintas, yaitu kendaraan tipe A, tipe B dan tipe C.

**Masukan** terdiri dari beberapa karakter ('A', 'B', 'C') yang dipisahkan oleh spasi. Masukan akan berakhir apabila karakter tidak valid.

**Keluaran** terdiri dari tiga baris, yang masing-masingnya menyatakan banyaknya kendaraan tipe A, B, dan C yang melintasi di gerbang tol Buahbatu.

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	A A A C A B B A B C A A E	Tipe A = 7 Tipe B = 3 Tipe C = 2

- 6) Sebuah sensor temperature perlu untuk dipasang di suatu lab rahasia. Sensor ini mampu merekam perubahan suhu yang terjadi dan data statistic lainnya. Buatlah aplikasi sensor berdasarkan ketentuan berikut ini!

**Masukan** terdiri dari beberapa bilangan bulat yang dipisahkan oleh spasi. Bilangan ini menyatakan temperature yang direkam oleh sensor. Masukan akan berakhir apabila temperature adalah 0 sebanyak dua kali berturut-turut.

**Keluaran** berupa tiga nilai, yang menyatakan temperature tertinggi, terendah dan rata-rata temperature yang direkam. Nol terakhir tidak dimasukkan ke dalam perhitungan.

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran	Penjelasan
1	2 5 9 0 3 -2 4 0 0	9 -2 2.625	Tertinggi 9 Terendah -2 Rata-rata=(2 + 5 + 9 + 0 + 3 + (-2) + 4 + 0) / 8 = 2.625
2	5 0 0	5 0 2.5	Tertinggi 5 Terendah 0 Rata-rata=(5 + 0) / 2 = 2.5

- 7) Sebuah program digunakan untuk menampilkan pola tertentu!

**Masukan** terdiri dari sebuah bilangan bulat positif x.

**Keluaran** terdiri dari sebuah pola seperti contoh berikut,

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	1	1
2	7	1 2 3 4 5 6 7 1 2 3 4 5 6 7

- 8) Sebuah program digunakan untuk menampilkan pola tertentu!

**Masukan** terdiri dari sebuah bilangan bulat positif x.

**Keluaran** terdiri dari sebuah pola seperti contoh berikut,

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	1	1
2	7	1 1 1 1 1 1 1 2 2 3 3 4 4 5 5 6 6 7 7 7 7 7 7 7

- 9) Sebuah program digunakan untuk menampilkan pola tertentu!

**Masukan** terdiri dari sebuah bilangan bulat positif x.

**Keluaran** terdiri dari sebuah pola seperti contoh berikut,

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	1	1
2	7	1 1 1 1 1 1 1 2 2 3 3 4 4 5 5 6 6 7 7



Fakultas Informatika  
School of Computing  
Telkom University



## BAB 2. PEMROGRAMAN MODULAR

Pemrograman modular merupakan suatu pendekatan atau konsep di mana suatu program yang besar dan rumit dipecah-pecah menjadi beberapa bagian yang lebih kecil dan sederhana. Cirinya adalah program yang dibuat tersusun dari beberapa blok program yang saling terpisah atau yang dikenal dengan istilah subprogram. Di sini terdapat sebuah program utama (atau main program) yang bertugas sebagai induk dari program yang dibuat atau acuan ketika program dieksekusi, sedangkan subprogram hanya akan dieksekusi apabila dipanggil sebagai instruksi baik secara langsung atau tidak langsung oleh program utama. Sebagai contoh **func main()** pada GoLang merupakan sebuah subprogram yang bertugas sebagai program utama. Selain itu argumen atau parameter yang dikirimkan ke subprogram merupakan data awal agar subprogram tersebut dapat bekerja.

Secara umum subprogram dibedakan menjadi dua jenis, yaitu procedur dan fungsi. Pada bahasa GoLang hanya menggunakan kata kunci **func** untuk deklarasi fungsi dan prosedur. Perbedaan keduanya ada pada:

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

Pembahasan lengkap terkait pemrograman secara modular dengan menggunakan Go ini akan diuraikan pada modul 3 sampai dengan modul 8.

### a) Modul 4. Fungsi

Modul 4 membahas terkait fungsi, cara pemanggilan serta contoh dan soal latihannya.

### b) Modul 5. Prosedure

Modul ini membahas terkait prosedur, cara pemanggilan serta contoh dan soal latihannya.

### c) Modul 6. Rekursif

Modul 6 membahas terkait rekursif di dalam pemrograman modular, contoh dan soal latihannya.

### d) Modul 7. Asesmen Praktikum 1

Modul ini berisi contoh soal asesmen yang berkaitan dengan pemrograman modular.

### e) Modul 8. Review Asesmen Praktikum 1

Modul 8 ini berisi pembahasan soal asesmen praktikum 1.

## MODUL 4. FUNGSI

### 4.1. Definisi Function

Fungsi merupakan satu kesatuan rangkaian instruksi yang memberikan atau menghasilkan suatu nilai, di mana biasanya memetakan input ke suatu nilai yang lain. Oleh karena itu, fungsi selalu menghasilkan/mengembalikan nilai. Suatu subprogram dikatakan fungsi apabila:

1. **Ada** deklarasi tipe nilai yang dikembalikan, dan
2. **Terdapat** kata kunci **return** dalam badan subprogram.

Maka fungsi digunakan di mana suatu nilai biasanya diperlukan, seperti:

- 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: median, rerata, nilaiTerbesar, ketemu, selesai, ...

### 4.2. Deklarasi Function

Deklarasi fungsi sama dengan prosedur, yaitu berada pada blok yang terpisah dengan program utama.

Notasi Algoritma	
1	function <nama function> (<params>) -> <type>
2 kamus	
3	{deklarasi variabel lokal dari fungsi}
4	...
5	algoritma
6	{badan algoritma fungsi}
7	...
8	return <value/variabel>
9	endfunction
Notasi dalam bahasa Go	
10	func <nama function> (<params>) <type> {
11	/* deklarasi variabel lokal dari fungsi */
12	...
13	/* badan algoritma fungsi*/
14	...
15	return <value/variabel>
16	}
17	}

Pada bagian deklarasi terlihat setelah parameter terdapat **tipe data dari nilai** yang dikembalikan, sedangkan pada bagian badan fungsi terdapat **return dari nilai** yang dikembalikan.

Berikut adalah contoh fungsi untuk menghitung volume dari tabung apabila jari-jari alas dan tinggi tabung diketahui.

Notasi Algoritma	
1	function volumeTabung(jari_jari,tinggi : integer) -> real
2	kamus
3	luasAlas, volume: real
4	algoritma
5	luasAlas <- 3.14 * (jari_jari * jari_jari)
6	volume <- luasAlas * tinggi
7	return volume
8	endfunction
Notasi dalam bahasa Go	
10	func volumeTabung(jari_jari,tinggi int) float64 {
11	var luasAlas,volume float64
12	luasAlas = 3.14 * float64(jari_jari * jari_jari)
13	volume = luasAlas * tinggi
14	return volume
15	}

### 4.3. Cara Pemanggilan Function

Sama halnya dengan prosedur, pemanggilan fungsi cukup dilakukan dengan penulisan nama fungsi beserta argumen yang diminta oleh parameter dari fungsi. Perbedaannya dengan prosedur adalah fungsi bisa di-assign ke suatu variabel, menjadi bagian dari ekspresi, dan argumen dari suatu subprogram.

Notasi Algoritma	
1	program ContohProsedur
2	kamus
3	r,t : integer
4	v1,v2 : real
5	algoritma
6	r <- 5;
7	t <- 10
8	v1 <- volumeTabung(r,t) {cara pemanggilan #1}
9	v2 <- volumeTabung(r,t) + volumeTabung(15,t) {cara pemanggilan #2}
10	output(volumeTabung(14,100)) {cara pemanggilan #3}
11	endprogram
Notasi dalam bahasa Go	
12	func main() {
13	var r,t int
14	var v1,v2 float64
15	r = 5
16	t = 10
17	v1 = volumeTabung(r,t) // cara pemanggilan #1

```

18     v2 = volumeTabung(r, t) + volumeTabung(15, t)    // cara pemanggilan #2
19     fmt.Println(volumeTabung(14, 100))                // cara pemanggilan #3
20 }
```

Pada contoh pemanggilan fungsi di atas terlihat tidak ada perbedaan pada saat pemanggilan fungsi pada pseudocode ataupun GoLang. Di sini terlihat fungsi bisa di-assign ke suatu variabel pada saat pemanggilan, bisa dioperasikan sesuai dengan tipe data yang dikembalikan, dan juga bisa langsung ditampilkan dengan perintah output ataupun print.

#### 4.4. Contoh Program dengan Function

Berikut ini adalah contoh penulisan fungsi pada suatu program lengkap.

Buatlah sebuah program beserta fungsi yang digunakan untuk menghitung nilai faktorial dan permutasi.

**Masukan** terdiri dari dua buah bilangan positif  $a$  dan  $b$ .

**Keluaran** berupa sebuah bilangan bulat yang menyatakan nilai  $a$  permutasi  $b$  apabila  $a \geq b$  atau  $b$  permutasi  $a$  untuk kemungkinan yang lain.

```

1 package main
2 import "fmt"
3 func main(){
4     var a,b int
5     fmt.Scan(&a, &b)
6     if a >= b {
7         fmt.Println(permutasi(a,b))
8     }else{
9         fmt.Println(permutasi(b,a))
10    }
11 }
12 func faktorial(n int) int{
13     var hasil int = 1
14     var i int
15     for i = 1; i <= n; i++ {
16         hasil = hasil * i
17     }
18     return hasil
19 }
20 func permutasi(n,r int) int {
21     return faktorial(n) / faktorial(n-r)
22 }
```

```

D:\DEV\DEMO>go build contoh.go
D:\DEV\DEMO>contoh.exe
2 5
20
```

Pada contoh di atas fungsi faktorial dipanggil secara tidak langsung melalui fungsi permutasi, dan fungsi faktorial dan permutasi dipanggil sebagai ekspresi dari suatu statement.

#### 4.5. Soal Latihan Modul 4

1. Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika diskrit untuk mempelajari kombinasi dan permutasi. Jonas salah seorang mahasiswa, iseng untuk mengimplementasikannya ke dalam suatu program. Oleh karena itu bersediakah kalian membantu Jonas? (tidak tentunya ya :p)

**Masukan** terdiri dari empat buah bilangan asli  $a, b, c$ , dan  $d$  yang dipisahkan oleh spasi, di mana  $a \geq c$  dan  $b \geq d$ .

**Keluaran** terdiri dari dua baris, di mana baris pertama adalah hasil permutasi dan kombinasi  $a$  terhadap  $c$ , sedangkan baris kedua adalah hasil permutasi dan kombinasi  $b$  terhadap  $d$ .

**Catatan:** permutasi ( $P$ ) dan kombinasi ( $C$ ) dari  $n$  terhadap  $r$  di mana ( $n \geq r$ ) dapat dihitung dengan menggunakan persamaan berikut!

$$P(n, r) = \frac{n!}{(n-r)!}, \text{ sedangkan } C(n, r) = \frac{n!}{r!(n-r)!}$$

Contoh

Fakultas Informatika  
School of Computing

No	Masukan	Keluaran	Penjelasan
1	5 10 3 10	60 10 3628800 1	$P(5, 3) = 5! / 2! = 120 / 2 = 60$ $C(5, 3) = 5! / (3! \times 2!) = 120 / 12 = 10$ $P(10, 10) = 10! / 0! = 3628800 / 1 = 3628800$ $C(10, 10) = 10! / (10! \times 0!) = 10! / 10! = 1$
2	8 0 2 0	56 28 1 1	

Selesaikan program tersebut dengan memanfaatkan subprogram yang diberikan berikut ini!

```
function factorial(n: integer) → integer
{mengembalikan nilai faktorial dari n}

function permutation(n, r : integer) → integer
{Mengembalikan hasil n permutasi r, di mana n >= r}

function combination(n, r : integer) → integer
{Mengembalikan hasil n kombinasi r, di mana n >= r}
```

2. Diberikan tiga buah fungsi matematika yaitu  $f(x) = x^2$ ,  $g(x) = x - 2$  dan  $h(x) = x + 1$ . Fungsi komposisi  $(fogoh)(x)$  artinya adalah  $f(g(h(x)))$ . Tuliskan  $f(x)$ ,  $g(x)$  dan  $h(x)$  dalam bentuk function.

**Masukan** terdiri dari sebuah bilangan bulat  $a$ ,  $b$  dan  $c$  yang dipisahkan oleh spasi.

**Keluaran** terdiri dari tiga baris, di mana baris pertama adalah  $(fogoh)(a)$ , baris kedua  $(gohof)(b)$ , dan baris ketiga adalah  $(hofog)(c)$ !

#### Contoh

No	Masukan	Keluaran	Penjelasan
1	7 2 10	36 3 65	$(fogog)(7) = 36$ $(gohof)(2) = 3$ $(hofog)(10) = 65$
2	5 5 5	16 24 10	$(fogog)(5) = 16$ $(gohof)(5) = 24$ $(hofog)(5) = 10$
3	3 8 4	4 63 5	$(fogog)(5) = 4$ $(gohof)(5) = 63$ $(hofog)(5) = 5$

3. [Lingkaran] Suatu lingkaran didefinisikan dengan koordinat titik pusat  $(cx, cy)$  dengan radius  $r$ . Apabila diberikan dua buah lingkaran, maka tentukan posisi sebuah titik sembarang  $(x, y)$  berdasarkan dua lingkaran tersebut.

**Masukan** terdiri dari beberapa tiga baris. Baris pertama dan kedua adalah koordinat titik pusat dan radius dari lingkaran 1 dan lingkaran 2, sedangkan baris ketiga adalah koordinat titik sembarang. Asumsi sumbu x dan y dari semua titik dan juga radius direpresentasikan dengan bilangan bulat.

**Keluaran** berupa string yang menyatakan posisi titik "Titik di dalam lingkaran 1 dan 2", "Titik di dalam lingkaran 1", "Titik di dalam lingkaran 2", atau "Titik di luar lingkaran 1 dan 2".

#### Contoh

No	Masukan	Keluaran
1	1 1 5 8 8 4 2 2	Titik di dalam lingkaran 1
2	1 2 3	Titik di dalam lingkaran 2

	4 5 6 7 8	
3	5 10 15 -15 4 20 0 0	Titik di dalam lingkaran 1 dan 2
4	1 1 5 8 8 4 15 20	Titik di luar lingkaran 1 dan 2

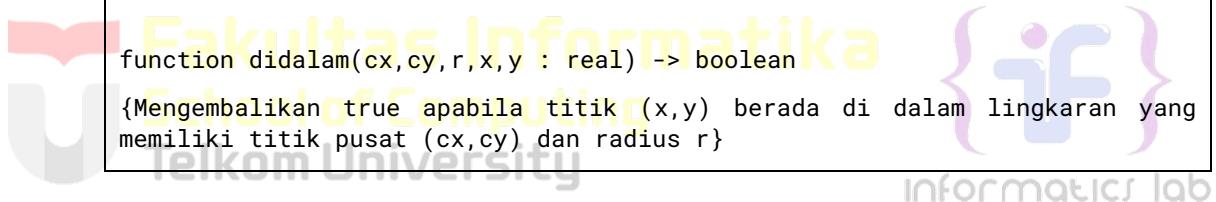
Fungsi untuk menghitung jarak titik (a, b) dan (c, d) dimana rumus jarak adalah:

$$jarak = \sqrt{(a - c)^2 + (b - d)^2}$$

dan juga fungsi untuk menentukan posisi sebuah titik sembarang berada di dalam suatu lingkaran atau tidak.

```
function jarak(a,b,c,d : real) -> real
{Mengembalikan jarak antara titik (a,b) dan titik (c,d)}

function didalam(cx,cy,r,x,y : real) -> boolean
{Mengembalikan true apabila titik (x,y) berada di dalam lingkaran yang memiliki titik pusat (cx,cy) dan radius r}
```



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

## MODUL 5. PROSEDUR

### 5.1. Definisi Procedure

Prosedur dapat dianggap sebagai potongan beberapa instruksi program menjadi suatu **instruksi baru** yang dibuat untuk mengurangi kerumitan dari kode program yang kompleks pada suatu program yang besar. Prosedur akan menghasilkan suatu akibat atau efek langsung pada program ketika dipanggil pada program utama. Suatu subprogram dikatakan prosedur apabila:

1. **Tidak ada** deklarasi tipe nilai yang dikembalikan, dan
2. **Tidak terdapat** kata kunci **return** dalam badan subprogram.

Kedudukannya prosedur sama seperti instruksi dasar yang sudah ada sebelumnya (**assignment**) dan/atau instruksi yang berasal dari paket (**fmt**), seperti **fmt.Scan** dan **fmt.Print**. Karena itu selalu pilih nama prosedur yang berbentuk **kata kerja** atau sesuatu yang merepresentasikan proses sebagai nama dari prosedur. Contoh: **cetak**, **hitungRerata**, **cariNilai**, **belok**, **mulai**, ...

### 5.2. Deklarasi Procedure

Berikut ini adalah cara penulisan deklarasi prosedur pada notasi Pseudocode dan Go Lang.

Notasi Algoritma	
1	procedure <nama procedure> (<params>)
2	kamus
3	{deklarasi variabel lokal dari procedure}
4	...
5	algoritma
6	{badan algoritma procedure}
7	...
8	endprocedure
Notasi dalam bahasa Go	
9	func <nama procedure> <(params)> {
10	/* deklarasi variabel lokal dari procedure */
11	...
12	/* badan algoritma procedure */
13	...
14	}

Penulisan deklarasi ini berada di luar blok yang dari program utama atau **func main()** pada suatu program Go, dan bisa ditulis sebelum atau setelah dari blok program utama tersebut.

Contoh deklarasi prosedur mencetak  $n$  nilai pertama dari deret Fibonacci.

Notasi Algoritma	
1	procedure cetakNFibo(in n : integer)
2	kamus
3	f1, f2, f3, i : integer
4	algoritma
5	f2 $\leftarrow$ 0
6	f3 $\leftarrow$ 1
7	for i $\leftarrow$ 1 to n do
8	output(f3)
9	f1 $\leftarrow$ f2
10	f2 $\leftarrow$ f3
11	f3 $\leftarrow$ f1 + f2
12	endfor
13	endprocedure
Notasi dalam bahasa Go	
14	func cetakNFibo(n int) {
15	var f1, f2, f3 int
16	f2 = 0
17	f3 = 1
18	for i := 1; i $\leq$ n; i++ {
19	fmt.Println(f3)
20	f1 = f2
21	f2 = f3
22	f3 = f1 + f2
23	}
24	}

Catatan: Kata kunci **in** pada contoh di atas akan dijelaskan pada materi parameter di modul 5 ini.



### 5.3. Cara Pemanggilan Procedure

Seperti yang sudah dijelaskan sebelumnya, suatu **prosedur hanya akan dieksekusi apabila dipanggil** baik secara langsung atau tidak langsung oleh program utama. Tidak langsung di sini maksudnya adalah prosedur dipanggil oleh program utama melalui perantara subprogram yang lain.

Pemanggilan suatu prosedure cukup mudah, yaitu dengan hanya **menuliskan nama beserta parameter atau argumen yang diminta dari suatu prosedur**. Sebagai contoh prosedur *cetakNFibo* di atas dipanggil dengan menuliskan namanya, kemudian sebuah variabel atau nilai integer tertentu sebagai argumen untuk paramter *n*. Contoh:

Notasi Algoritma	
1	program contohprosedur
2	kamus
3	x : integer
4	algoritma
5	x $\leftarrow$ 5
6	cetakNFibo(x)       {cara pemanggilan #1}
7	cetakNFibo(100)     {cara pemanggilan #2}

8	<pre>endprogram</pre> <p style="text-align: center;"><b>Notasi dalam bahasa Go</b></p>
9	<pre>func main() {     var x int     x = 5     cetakNFibo(x)    {cara pemanggilan #1}     cetakNFibo(100)   {cara pemanggilan #2} }</pre>

Dari contoh di atas terlihat bahwa cara pemanggilan dengan notasi pseudocode dan GoLang adalah sama. Argumen yang digunakan untuk parameter n berupa integer (sesuai deklarasi) yang terdapat pada suatu variabel (cara pemanggilan #1) atau nilainya secara langsung (cara pemanggilan #2).

#### 5.4. Contoh Program dengan Procedure

Berikut ini adalah contoh penulisan prosedur pada suatu program lengkap.

Buatlah sebuah program beserta prosedur yang digunakan untuk menampilkan suatu pesan error, warning atau informasi berdasarkan masukan dari user.

**Masukan** terdiri dari sebuah bilangan bulat **flag** (0 s.d. 2) dan sebuah string pesan **M**.

**Keluaran** berupa string pesan **M** beserta jenis pesannya, yaitu error, warning atau informasi berdasarkan nilai flag 0, 1 dan 2 secara berturut-turut.

```

1 package main
2 import "fmt"
3
4 func main(){
5     var bilangan int
6     var pesan string
7     fmt.Scan(&bilangan, &pesan)
8     cetakPesan(pesan,bilangan)
9 }
10
11 func cetakPesan(M string, flag int){
12     var jenis string = ""
13     if flag == 0 {
14         jenis = "error"
15     }else if flag == 1 {
16         jenis = "warning"
17     }else if flag == 2 {
18         jenis = "informasi"
19     }
20     fmt.Println(M,jenis)
21 }
```

```

D:\DEV\DEMO>go build contoh.go
D:\DEV\DEMO>contoh.exe
1 hello_world
hello_world error
```

Penulisan argumen pada parameter `cetakPesan(pesan, bilangan)` harus sesuai urutan tipe data pada `func cetakPesan(M string, flag int)`, yaitu string kemudian integer.

## 5.5. Parameter

Suatu subprogram yang dipanggil dapat berkomunikasi dengan pemanggilnya melalui argumen yang diberikan melalui parameter yang dideklarasikan pada subprogramnya. Berikut ini jenis atau pembagian dari parameter.

Berdasarkan letak penulisannya pada program, maka parameter dapat dikelompokkan menjadi dua, yaitu parameter formal dan parameter aktual.

```
1 func volumeTabung(jari_jari,tinggi int) float64 {
2     var luasAlas,volume float64
3
4     luasAlas = 3.14 * float64(jari_jari * jari_jari)
5     volume = luasAlas * tinggi
6     return volume
7 }
8
9 func main() {
10    var r,t int
11    var v1,v2 float64
12    r = 5; t = 10
13    v1 = volumeTabung(r,t)
14    v2 = volumeTabung(r,t) + volumeTabung(15,t)
15    fmt.Println(volumeTabung(14,100))
16 }
```

informatics lab

### 1. Parameter Formal

Parameter formal adalah parameter yang ditulis pada saat deklarasi suatu subprogram, parameter ini berfungsi sebagai petunjuk bahwa argumen apa saja yang diperlukan pada saat pemanggilan subprogram.

Sebagai contoh parameter `jari_jari, tinggi` pada deklarasi `fungsi volumeTabung` adalah parameter formal (teks berwarna merah). Artinya ketika memanggil `volumeTabung` maka kita harus mempersiapkan dua integer (berapapun nilainya) sebagai `jari_jari` dan `tinggi`.

### 2. Parameter Aktual

Sedangkan parameter aktual adalah argumen yang digunakan pada bagian parameter saat pemanggilan suatu subprogram. Banyaknya argumen dan tipe data yang terdapat pada parameter aktual harus mengikuti parameter formal.

Sebagai contoh argumen `r, t, 15, 14` dan `100` pada contoh kode di atas (teks berwarna biru) adalah parameter aktual, yang menyatakan nilai yang kita berikan sebagai `jari-jari` dan `tinggi`.

Selain itu parameter juga dikelompokkan berdasarkan alokasi memorinya, yaitu pass by value dan pass by reference.

### 1. Pass by Value

Nilai pada parameter aktual akan disalin ke variabel lokal (parameter formal) pada subprogram. Artinya parameter aktual dan formal dialokasikan di dalam memori komputer dengan alamat memori yang berbeda. Subprogram dapat menggunakan nilai pada parameter formal tersebut untuk proses apapun, tetapi tidak dapat mengembalikan informasinya ke pemanggil melalui parameter aktual karena pemanggil tidak dapat mengakses memori yang digunakan oleh subprogram. Pass by value bisa digunakan baik oleh fungsi ataupun prosedur.

Pada notasi pseudocode, secara semuanya parameter formal pada fungsi adalah pass by value, sedangkan pada prosedur diberi kata kunci `in` pada saat penulisan parameter formal. Sedangkan pada bahasa pemrograman Go sama seperti fungsi pada pseudocode, tidak terdapat kata kunci khusus untuk parameter formal fungsi dan prosedur.

### 2. Pass by Reference (Pointer)

Ketika parameter didefinisikan sebagai pass by reference, maka pada saat pemanggilan parameter formal akan berperan sebagai pointer yang menyimpan alamat memori dari parameter aktual. Sehingga perubahan nilai yang terjadi pada parameter formal tersebut akan berdampak pada parameter aktual. Artinya nilai terakhirnya akan dapat diketahui oleh si pemanggil setelah subprogram tersebut selesai dieksekusi. Pass by reference sebaiknya digunakan hanya untuk prosedur.

Penulisan parameter pass by reference pada prosedur baik pseudocode dan Go menggunakan kata kunci atau identifier khusus. Pada pseudocode menggunakan kata kunci `in/out`, sedangkan pada bahasa Go diberi identifier asterik (\*) sebelum tipe data di parameter formal yang menjadi pass by reference.

#### Catatan:

- Parameter pada fungsi sebaiknya adalah pass by value, hal ini dikarenakan fungsi bisa mengembalikan (return) nilai ke pemanggil dan tidak memberikan efek langsung pada program, walaupun tidak menutup kemungkinan menggunakan pass by reference.
- Penggunaan pass by reference sebaiknya pada prosedur karena prosedur tidak bisa mengembalikan nilai ke pemanggil. Dengan memanfaatkan pass by reference maka prosedur seolah-olah bisa mengirimkan nilai kepada si pemanggil.

Untuk lebih jelas perhatikan contoh sebuah subprogram yang digunakan untuk menghitung persamaan berikut ini:

$$f(x,y) = 2x - \frac{y}{2} + 3$$

Notasi Algoritma	
<pre> function f1(x,y : integer) → real kamus     hasil : real algoritma     hasil ← 2*x - 0.5*y + 3     return hasil endfunction  procedure f2(in x,y : integer, in/out hasil:real) algoritma     hasil ← 2*x - 0.5*y + 3 endprocedure  program Contoh kamus     a,b : integer     c : real algoritma     input(a,b)     f2(a,b,c)     output(c, f1(b,a)) endprogram </pre>	<p>x dan y pada fungsi f1 dan prosedur f2 adalah <b>pass by value</b>, sedangkan variabel hasil pada prosedur f2 adalah <b>pass by reference</b>.</p> <p>Untuk pemanggilan dengan notasi pseudocode masih sama dengan materi yang sudah dipelajari sebelumnya</p>
Notasi dalam bahasa Go	
<pre> package main import "fmt"  func f1(x,y int) float64 {     var hasil float64     hasil = float64(2*x) - 0.5*float64(y) + 3.0     return hasil }  func f2(x,y int, hasil *float64) {     *hasil = float64(2*x) - 0.5*float64(y) + 3.0 }  func main(){     var a,b int; var c float64     fmt.Scan(&amp;a,&amp;b)     f2(a,b,&amp;c)     output(c, f1(b,a)) endprogram </pre>	<p>x dan y pada fungsi f1 dan prosedur f2 adalah <b>pass by value</b>, sedangkan variabel hasil pada prosedur f2 adalah <b>pass by reference</b>.</p> <p>Karena variabel hasil adalah <b>pointer to float64</b>, maka untuk mengaksesnya menggunakan <b>simbol bintang (*)</b> pada variabelnya.</p> <p>Pada bahasa Go saat pemanggilan prosedur f2, maka <b>parameter aktual</b> untuk <b>pass by reference</b> harus diberi <b>ampersand "&amp;"</b>, contohnya <b>&amp;c</b></p>

## 5.6. Soal Latihan Modul 5

- 1) Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika diskrit untuk mempelajari kombinasi dan permutasi. Jonas salah seorang mahasiswa, iseng untuk mengimplementasikannya ke dalam suatu program. Oleh karena itu bersediakah kalian membantu Jonas? (tidak tentunya ya :p)

**Masukan** terdiri dari empat buah bilangan asli  $a, b, c$ , dan  $d$  yang dipisahkan oleh spasi, di mana  $a \geq c$  dan  $b \geq d$ .

**Keluaran** terdiri dari dua baris, di mana baris pertama adalah hasil permutasi dan kombinasi  $a$  terhadap  $c$ , sedangkan baris kedua adalah hasil permutasi dan kombinasi  $b$  terhadap  $d$ .

**Catatan:** permutasi ( $P$ ) dan kombinasi ( $C$ ) dari  $n$  terhadap  $r$  di mana ( $n \geq r$ ) dapat dihitung dengan menggunakan persamaan berikut!

$$P(n, r) = \frac{n!}{(n-r)!}, \text{ sedangkan } C(n, r) = \frac{n!}{r!(n-r)!}$$

### Contoh

No	Masukan	Keluaran	Penjelasan
1	5 10 3 10	60 10 3628800 1	$P(5, 3) = 5! / 2! = 120 / 2 = 60$ $C(5, 3) = 5! / (3! \times 2!) = 120 / 12 = 10$ $P(10, 10) = 10! / 0! = 3628800 / 1 = 3628800$ $C(10, 10) = 10! / (10! \times 0!) = 10! / 10! = 1$
2	8 0 2 0	56 28 1 1	

Selesaikan program tersebut dengan memanfaatkan prosedure yang diberikan berikut ini!

```
procedure factorial(in n: integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n
 F.S. hasil berisi nilai faktorial dari n}

procedure permutation(in n,r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, di mana n >= r
 F.S. hasil berisi nilai dari n permutasi r}

procedure combination(in n,r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, di mana n >= r
 F.S. hasil berisi nilai dari n kombinasi r}
```

- 2) 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 penggerjaan dibaca di dalam prosedur.

```
procedure 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.

No	Masukan	Keluaran
1	Astuti 20 50 301 301 61 71 75 10 Bertha 25 47 301 26 50 60 65 21 Selesai	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.

- 3) 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 selalu mencapai suku dengan nilai 1.

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

```
procedure 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.

No	Masukan	Keluaran
1	22	22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

## MODUL 6. REKURSIF

### 6.1. Pengantar Rekursif

Pada modul-modul sebelumnya sudah dijelaskan bahwa suatu subprogram baik fungsi atau prosedur bisa memanggil subprogram lainnya. Hal ini tidak menutup kemungkinan bahwa subprogram yang dipanggil adalah dirinya sendiri. Dalam pemrograman teknik ini dikenal dengan istilah rekursif.

Rekursif secara sederhana dapat diartikan sebagai cara menyelesaikan suatu masalah dengan cara menyelesaikan sub-masalah yang identik dari masalah utama. Sebagai contoh perhatikan prosedur cetak berikut ini!

	Notasi Algoritma	Notasi dalam bahasa GO
1	procedure <b>cetak</b> (in x:integer)	func cetak(x int){
2	algoritma	fmt.Println(x)
3	output(x)	cetak(x+1)
4	<b>cetak</b> (x+1)	}
5	endprocedure	

Apabila diperhatikan subprogram **cetak()** di atas, terlihat pada baris ke-4 terdapat pemanggilan subprogram **cetak()** kembali. Misalnya apabila kita eksekusi perintah **cetak(5)** maka akan menampilkan angka 5 6 7 8 9...dst tanpa henti. Artinya setiap pemanggilan subprogram **cetak()** nilai x akan selalu bertambah 1 (*increment by one*) secara **terus menerus tanpa henti**.

```
1 package main
2 import "fmt"
3 func main(){
4     cetak(5)
5 }
6 func cetak(x int){
7     fmt.Println(x)
8     cetak(x+1)
9 }
```

```
D:\DEV\DEMO>go build contoh.go
D:\DEV\DEMO>contoh.exe
5
6
7
8
9
10
11
12
13
...
```

Oleh karena itu biasanya ditambahkan struktur kontrol percabangan (if-then) untuk menghentikan proses rekursif ini. Kondisi ini disebut juga dengan **base-case**, artinya apabila kondisi base-case bernilai

true maka proses rekursif akan berhenti. Sebagai contoh misalnya base case adalah ketika x bernilai 10 atau  $x == 10$ , maka tidak perlu dilakukan rekursif.

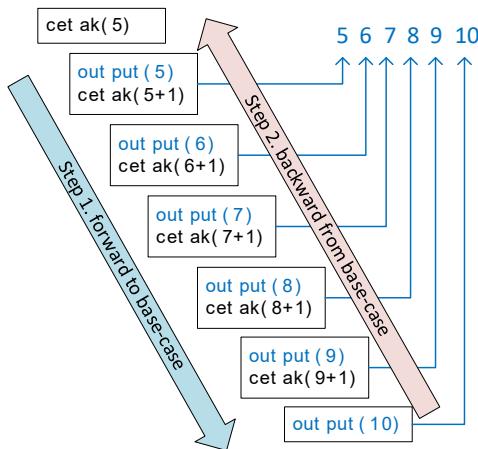
```
1 procedure cetak(in x:integer)
2 algoritma
3     if x == 10 then
4         output(x)
5     else
6         output(x)
7         cetak(x+1)
8     endif
9 endprocedure
```

Apabila diperhatikan pada baris ke-3 di Program di atas, kita telah menambahkan **base-case** seperti penjelasan sebelumnya. Selanjutnya pada bagian aksi dari else di baris ke-6 dan ke-7 kita namakan **recursive-case** atau bagian di mana pemanggilan dirinya sendiri tersebut terjadi. Kondisi dari **recursive-case** ini adalah negasi dari kondisi **base-case** atau ketika nilai  $x \neq 10$ .

```
1 package main
2 import "fmt"
3 func main(){
4     cetak(5)
5 }
6 func cetak(x int){
7     if x == 10 {
8         fmt.Println(x)
9     }else{
10        fmt.Println(x)
11        cetak(x+1)
12    }
13 }
```

```
D:\DEV\DEMO>go build contoh.go
D:\DEV\DEMO>contoh.exe
5
6
7
8
9
10
```

Apabila program di atas ini dijalankan maka akan tampil angka 5 6 7 8 9 10. Terlihat bahwa proses rekursif berhasil dihentikan ketika  $x == 10$ .



Gambar 1. Ilustrasi proses forward dan backward pada saat rekursif.

Pada Gambar 2 memperlihatkan saat subprogram dipanggil secara rekursif, maka subprogram akan terus melakukan pemanggilan (**forward**) hingga berhenti pada saat kondisi **base-case** terpenuhi atau **true**. Setelah itu akan terjadi proses **backward** atau kembali ke subprogram yang sebelumnya. Artinya setelah semua instruksi **cetak(10)** selesai dieksekusi, maka program akan kembali ke **cetak(9)** yang memanggil **cetak(10)** tersebut. Begitu seterusnya hingga kembali ke **cetak(5)**.

Perhatikan modifikasi program di atas dengan menukar posisi baris 10 dan 11, mengakibatkan ketika program dijalankan maka akan menampilkan 10 9 8 7 6 5. Kenapa bisa demikian? Pahami proses **backward** pada Gambar 2

```

1 package main
2 import "fmt"
3 func main(){
4     cetak(5)
5 }
6 func cetak(x int){
7     if x == 10 {
8         fmt.Println(x)
9     }else{
10        cetak(x+1)
11        fmt.Println(x)
12    }
13 }
```

```

D:\DEV\DEMO>go build contoh.go
D:\DEV\DEMO>contoh.exe
10
9
8
7
6
5
```

#### Catatan:

- Teknik rekursif ini merupakan salah satu alternatif untuk mengganti struktur kontrol perulangan dengan memanfaatkan subprogram (bisa fungsi ataupun prosedure).

- Untuk menghentikan proses rekursif digunakan percabangan (if-then).
- **Base-case** adalah kondisi di mana proses rekursif berhenti. **Base-case** merupakan hal terpenting dan **pertama yang harus diketahui ketika akan membuat program rekursif**. **Mustahil** membuat program rekursif tanpa mengetahui **base-case** terlebih dahulu.
- **Recursive-case** adalah kondisi dimana proses pemanggilan dirinya sendiri dilakukan. Kondisi **recursive-case** adalah **komplemen atau negasi** dari **base-case**.
- Setiap algoritma rekursif selalu memiliki padanan dalam bentuk algoritma interatif.

## 6.2. Komponen Rekursif

Algoritma rekursif terdiri dari dua komponen utama:

- **Base-case (Basis)**, yaitu bagian untuk menghentikan proses rekursif dan menjadi komponen terpenting di dalam sebuah rekursif.
- **Recursive-case**, yaitu bagian pemanggilan subprogramnya.

## 6.3. Contoh Program dengan menggunakan Rekursif

- 1) Membuat baris bilangan dari n hingga 1

Base-case: bilangan == 1

```

1 package main
2 import "fmt"
3 func main(){
4     var n int
5     fmt.Scan(&n)
6     baris(n)
7 }
8
9 func baris(bilangan int){
10    if bilangan == 1 {
11        fmt.Println(1)
12    }else{
13        fmt.Println(bilangan)
14        baris(bilangan - 1)
15    }
16 }
```

- 2) Menghitung hasil penjumlahan 1 hingga n

Base-case: n == 1

```

1 package main
2 import "fmt"
3 func main(){
4     var n int
5     fmt.Scan(&n)
6     fmt.Println(penjumlahan(n))
```

```
7 }
8
9 func penjumlahan(n int) int {
10    if n == 1 {
11        return 1
12    }else{
13        return n + penjumlahan(n-1)
14    }
15 }
```

- 3) Mencari dua pangkat n atau  $2^n$

Base-case:  $n == 0$

```
1 package main
2 import "fmt"
3 func main(){
4     var n int
5     fmt.Scan(&n)
6     fmt.Println(pangkat(n))
7 }
8
9 func pangkat(n int) int {
10    if n == 0 {
11        return 1
12    }else{
13        return 2 * pangkat(n-1)
14    }
15 }
```

- 4) Mencari nilai faktorial atau  $n!$

Base-case:  $n == 0$  atau  $n == 1$

```
1 package main
2 import "fmt"
3 func main(){
4     var n int
5     fmt.Scan(&n)
6     fmt.Println(faktorial(n))
7 }
8
9 func faktorial(n int) int {
10    if n == 0 || n == 1 {
11        return 1
12    }else{
13        return n * faktorial(n-1)
14    }
15 }
```

#### 6.4. Soal Latihan Modul 6

- 1) Deret fibonacci 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 fibonacci hingga suku ke-10. Buatlah program yang mengimplementasikan fungsi rekursif pada deret fibonacci tersebut.

$n$	0	1	2	3	4	5	6	7	8	9	10
$S_n$	0	1	1	2	3	5	8	13	21	34	55

- 2) Buatlah sebuah program yang digunakan untuk menampilkan pola bintang berikut ini dengan menggunakan fungsi rekursif. N adalah masukan dari user.

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	5	* ** *** **** *****
2	1	*
3	3	* ** ***

- 3) Buatlah program yang mengimplementasikan rekursif untuk menampilkan faktor bilangan dari suatu N, atau bilangan yang apa saja yang habis membagi N.

**Masukan** terdiri dari sebuah bilangan bulat positif N.

**Keluaran** terdiri dari barisan bilangan yang menjadi faktor dari N (terurut dari 1 hingga N ya).

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	5	1 5
2	12	1 2 3 4 6 12

- 4) Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan tertentu.

**Masukan** terdiri dari sebuah bilangan bulat positif N.

**Keluaran** terdiri dari barisan bilangan dari N hingga 1 dan kembali ke N.

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	5	5 4 3 2 1 2 3 4 5
2	9	9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9

- 5) Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan ganjil.

**Masukan** terdiri dari sebuah bilangan bulat positif N.

**Keluaran** terdiri dari barisan bilangan ganjil dari 1 hingga N.

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	5	1 3 5
2	20	1 3 5 7 9 11 13 15 17 19

- 6) Buatlah program yang mengimplementasikan rekursif untuk mencari hasil pangkat dari dua buah bilangan.

**Masukan** terdiri dari bilangan bulat x dan y.

**Keluaran** terdiri dari hasil x dipangkatkan y.

**Catatan:** diperbolehkan menggunakan asterik "\*", tapi dilarang menggunakan import "math".

**Contoh masukan dan keluaran:**

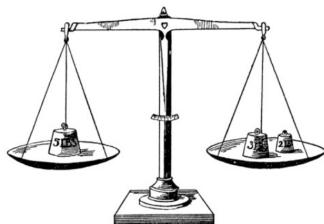
No	Masukan	Keluaran
1	2 2	4
2	5 3	125

## MODUL 7. ASESMEN PRAKTIKUM 1

### 7.1. Contoh Soal Asesmen Praktikum 1

Modul ini berisi contoh soal asesmen praktikum 1 dengan materi subprogram function dan procedure.

- 1) Sebuah timbangan disimulasikan dalam sebuah program. Timbangan memiliki wadah berbentuk tabung yang digunakan untuk zat cair yang akan ditimbang.



Program akan menampilkan selisih massa antara zat cair di wadah kiri dan kanan atau menampilkan "balance" jika massa di kiri dan kanan sama.

Lengkapi program di bawah ini.

```
package main
import "fmt"
const pi float64 = 3.14

func volume(r,t float64) float64{
/* mengembalikan volume tabung yang memiliki jari-jari lingkaran r dan tinggi t, yang mana volume adalah luas alas x tinggi tabung*/
    ...
}

func massa(r,t,p float64) float64{
/* mengembalikan massa tabung yang memiliki jari-jari lingkaran r dan tinggi t, serta massa jenis p, yang mana massa = volume x massa jenis*/
    ...
}

func display(m1, m2 float64) {
/* I.S. terdefinisi massa zat cair kiri m1 dan massa zat cair kanan m2 pada timbangan
F.S. menampilkan "BALANCE" apabila m1 dan m2 adalah sama, atau selisih positif apabila m1 dan m2 berbeda*/
    ...
}

func main(){
    var r float64                // jari-jari
    var tKiri, tKanan float64      // tinggi zat cair kiri dan kanan
    var mjKiri, mjKanan float64    // massa jenis zat cair
    var massaKiri, massaKanan float64 // massa zat cair kiri dan kanan
```

```

    // masukan jari-jari alas tabung

    // masukan tinggi zat cair di tabung kiri, beserta massa jenisnya
    // masukan tinggi zat cair di tabung kanan, beserta massa jenisnya

    // hitung massa zat cair di tabung kiri dan kanan

    // tampilkan hasil dari proses penimbangan

}

```

- 2) Buatlah program untuk menghitung biaya perjalanan yang harus dikeluarkan oleh Universitas Telkom (Tel-U) untuk study tour yang dilakukan oleh mahasiswa setiap tahunnya. Algoritma menerima input jumlah mahasiswa yang ikut, lama study tour (dalam satuan hari), dan tujuan study tour apakah domestik atau mancanegara. Ketentuan penghitungan biaya:
- Lama perjalanan yang menjadi tanggungan Tel-U maksimum 3 hari untuk tujuan domestik dan 8 hari untuk tujuan mancanegara. Kelebihan hari menjadi tanggungan pribadi masing-masing mahasiswa.
  - Biaya yang harus dialokasikan per mahasiswa per hari untuk tujuan domestik meliputi:
    - Biaya makan siang dan makan malam. Biaya 1 kali makan adalah Rp. 35.000.
    - Biaya penginapan sebesar Rp. 250.000 (sudah termasuk sarapan pagi)
    - Uang saku sebesar Rp. 300.000
  - Biaya yang harus dialokasikan per mahasiswa per hari untuk tujuan mancanegara adalah 1.5 kali biaya per hari untuk tujuan domestik.

Lengkapi potongan program berikut ini!

```

package main
import "fmt"

func tanggunganHari(jumlahHari int, tujuan string) int {
/* Mengembalikan jumlah hari maksimum yang biaya perjalanannya ditanggung oleh
   Tel-U berdasarkan lama study tour (jumlahHari) dan tujuan
   (domestik/mancanegara) */
    ...
}

func biayaPerHari(jumlahMhs int) int {
/* Menghitung biaya tour domestik per hari yang ditanggung oleh Tel-U untuk
   jumlah mahasiswa sebanyak jumlahMhs*/
    ...
}

func perhitunganBiaya(jumlahMhs, lamaPerjalanan int, tujuan string,
totalBiaya *float64) {
/* IS. Terdefinisi jumlah mahasiswa, lama, dan tujuan perjalanan
   FS. Telah dihitung biaya perjalanan yang ditanggung Tel-U */
}

```

```
// Panggil salah satu fungsi/prosedur untuk menghitung lama perjalanan
...
// Panggil fungsi/prosedur untuk menghitung biaya total tour domestik
seluruh mahasiswa
...
// Hitung biaya study tour seluruh mahasiswa jika tujuan domestik atau
mancanegara
...
}

func main(){
    var jumlah, lama int
    var tujuan string
    var biaya float64
    // lakukan proses masukan atau input di sini
    ...
    // hitung biaya perjalanan yang dikeluarkan Tel-U dengan memanggil
subprogram yang tepat
    ...
    // tampilkan biaya
    ...
}
```



## MODUL 8. REVIEW ASESMEN PRAKTIKUM 1

### 8.1. Review Soal Asesmen Praktikum 1

Modul ini berisi jawaban atas contoh soal asesmen praktikum 1 yang terdapat pada modul 7.

- 1) Sebuah timbangan disimulasikan dalam sebuah program. Timbangan memiliki wadah berbentuk tabung yang digunakan untuk zat cair yang akan ditimbang.



Program akan menampilkan selisih massa antara zat cair di wadah kiri dan kanan atau menampilkan "balance" jika massa di kiri dan kanan sama.

**Jawaban:**

```
1 package main
2 import "fmt"
3 const pi float64 = 3.14
4
5 func volume(r,t float64) float64{
6     /* mengembalikan volume tabung yang memiliki jari-jari lingkaran r dan tinggi
7     t, yang mana volume adalah luas alas x tinggi tabung*/
8     return pi * (r * r) * t
9 }
10
11 func massa(r,t,p float64) float64{
12     /* mengembalikan massa tabung yang memiliki jari-jari lingkaran r dan tinggi
13     t, serta massa jenis p, yang mana massa = volume x massa jenis*/
14     var vol float64 = volume(r,t)
15     return vol * p
16 }
17
18 func display(m1, m2 float64) {
19     /* I.S. terdefinisi massa zat cair kiri m1 dan massa zat cair kanan m2 pada
20     timbangan
21     F.S. menampilkan "BALANCE" apabila m1 dan m2 adalah sama, atau selisih
22     positif apabila m1 dan m2 berbeda*/
23     var selisih float64
24     if m1 == m2 {
25         fmt.Println("BALANCE")
26     }else{
27         selisih = m1 - m2
28         if selisih < 0 {
29             selisih = - selisih
30         }
31         fmt.Println(selisih)
32     }
```

```

33 }
34
35 func main(){
36     var r float64                      // jari-jari
37     var tKiri, tKanan float64           // tinggi zat cair kiri dan kanan
38     var mjKiri, mjKanan float64        // massa jenis zat cair
39     var massaKiri, massaKanan float64   // massa zat cair kiri dan kanan
40
41     // masukan jari-jari alas tabung
42     fmt.Scan(&r)
43     // masukan tinggi zat cair di tabung kiri, beserta massa jenisnya
44     fmt.Scan(&tKiri, &mjKiri)
45     // masukan tinggi zat cair di tabung kanan, beserta massa jenisnya
46     fmt.Scan(&tKanan, &mjKanan)
47     // hitung massa zat cair di tabung kiri dan kanan
48     massaKiri = massa(r,tKiri,mjKiri)
49     massaKanan = massa(r,tKanan,mjKanan)
50     // tampilkan hasil dari proses penimbangan
51     display(massaKiri,masaKanan)
52 }
```

- 2) Buatlah program untuk menghitung biaya perjalanan yang harus dikeluarkan oleh Universitas Telkom (Tel-U) untuk study tour yang dilakukan oleh mahasiswa setiap tahunnya. Algoritma menerima input jumlah mahasiswa yang ikut, lama study tour (dalam satuan hari), dan tujuan study tour apakah domestik atau mancanegara. Ketentuan penghitungan biaya:
- Lama perjalanan yang menjadi tanggungan Tel-U maksimum 3 hari untuk tujuan domestik dan 8 hari untuk tujuan mancanegara. Kelebihan hari menjadi tanggungan pribadi masing-masing mahasiswa.
  - Biaya yang harus dialokasikan per mahasiswa per hari untuk tujuan domestik meliputi:
    - Biaya makan siang dan makan malam. Biaya 1 kali makan adalah Rp. 35.000.
    - Biaya penginapan sebesar Rp. 250.000 (sudah termasuk sarapan pagi)
    - Uang saku sebesar Rp. 300.000
  - Biaya yang harus dialokasikan per mahasiswa per hari untuk tujuan mancanegara adalah 1.5 kali biaya per hari untuk tujuan domestik.

**Jawaban:**

```

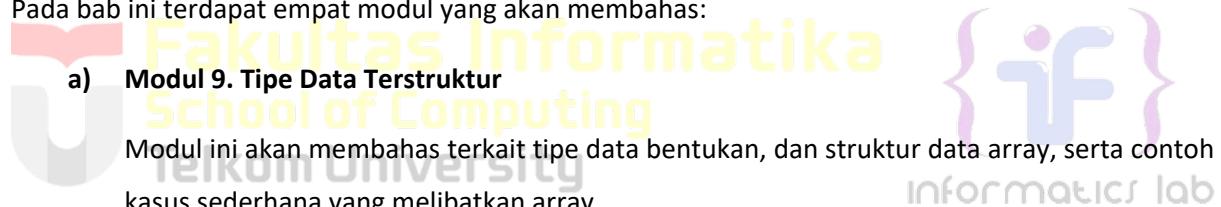
1 package main
2 import "fmt"
3
4 func tanggunganHari(jumlahHari int, tujuan string) int {
5     /* Mengembalikan jumlah hari maksimum yang biaya perjalannya ditanggung
6     oleh Tel-U berdasarkan lama study tour (jumlahHari) dan tujuan
7     (domestik/mancanegara) */
8     if tujuan == "domestik" && jumlahHari > 3 {
9         jumlahHari = 3
10    }else if tujuan == "mancanegara" && jumlahHari > 8 {
```

```
11         jumlahHari = 8
12     }
13     return jumlahHari
14 }
15
16 func biayaPerHari(jumlahMhs int) int {
17 /* Menghitung biaya tour domestik per hari yang ditanggung oleh Tel-U untuk
18 jumlah mahasiswa sebanyak jumlahMhs*/
19     return jumlahMhs * (70000+300000+250000)
20 }
21
22 func perhitunganBiaya(jumlahMhs, lamaPerjalanan int, tujuan string,
23 totalBiaya *float64) {
24 /* IS. Terdefinisi jumlah mahasiswa, lama, dan tujuan perjalanan FS. Telah
25 dihitung biaya perjalanan yang ditanggung Tel-U */
26     // Panggil salah satu fungsi/prosedur untuk menghitung lama perjalanan
27     var lama int = tanggunganHari(lamaPerjalanan, tujuan)
28     // Panggil fungsi/prosedur untuk menghitung biaya total tour domestik
29 seluruh mahasiswa
30     *totalBiaya = float64(biayaPerHari(jumlahMhs))
31     // Hitung biaya study tour seluruh mahasiswa jika tujuan domestik atau
32 mancanegara
33     *totalBiaya = *totalBiaya * float64(lama)
34     if tujuan == "mancanegara" {
35         *totalBiaya = 1.5 * *totalBiaya
36     }
37 }
38
39 func main(){
40     var jumlah, lama int
41     var tujuan string
42     var biaya float64
43     // lakukan proses masukan atau input di sini
44     fmt.Scan(&jumlah, &lama, &tujuan)
45     // hitung biaya perjalanan yang dikeluarkan Tel-U dengan memanggil
46 subprogram yang tepat
47     perhitunganBiaya(jumlah, lama, tujuan, &biaya)
48     // tampilkan biaya
49     fmt.Printf("Rp %.0f", biaya)
50 }
```

### BAB 3. PENCARIAN PADA DATA TERSTRUKTUR

Pada bab ini diperkenalkan beberapa ide dan algoritma untuk kasus-kasus pencarian suatu data pada sekumpulan data. Pertama akan dibahas terkait tipe data terstruktur seperti tipe bentukan dan array, yang mana dengan struktur ini memungkinkan pemrograman menyimpan data dengan struktur yang lebih kompleks dan dalam jumlah data yang banyak. Hal ini tentunya berbeda dengan tipe data dasar yang sudah dipelajari sebelumnya, yang mana setiap variabel dengan tipe data dasar hanya bisa menyimpan sebuah nilai saja. Sebagai contoh, melalui tipe bentukan memungkinkan pemrograman menyimpan data biodata mahasiswa yang berisi nama, nim, kelas, jurusan dan data lainnya dalam sebuah variabel, selain itu melalui array memungkinkan juga menyimpan data mahasiswa dalam jumlah yang sangat banyak hanya dengan sebuah variabel array. Kedua, bab ini akan membahas beberapa kasus-kasus pencarian yang melibatkan kumpulan data dengan tipe data dasar dan juga tipe bentukan, yaitu algoritma pencarian sekuensial untuk kumpulan data yang tersusun acak dan pencarian biner untuk kumpulan data yang tersusun secara terurut.

Pada bab ini terdapat empat modul yang akan membahas:



a) **Modul 9. Tipe Data Terstruktur**

Modul ini akan membahas terkait tipe data bentukan, dan struktur data array, serta contoh kasus sederhana yang melibatkan array.

b) **Modul 10. Pencarian Nilai Ekstrim**

Pada modul ini akan dibahas algoritma pencarian nilai terkecil atau terbesar pada sekumpulan data bertipe data dasar ataupun bentukan.

c) **Modul 11. Pencarian Nilai Tertentu**

Modul ini memaparkan algoritma pencarian nilai tertentu pada sekumpulan data yang tersusun acak ataupun tersusun terurut.

d) **Modul 12. Asesmen Praktikum 2**

Modul ini berisi contoh soal-soal asesmen praktikum dengan materi meliputi pada tipe data terstruktur dan kasus-kasus yang melipatkan pencarian data.

**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 update, pertukaran atau hapus data.

## MODUL 9. TIPE DATA TERSTRUKTUR

### 9.1. Tipe Bentukan

Tipe bentukan memungkinkan pemrograman untuk mendefinisikan suatu tipe data baru pada suatu bahasa pemrograman. Tipe bentukan ini dapat dibedakan atas dua jenis, yaitu Alias dan Structure.

#### 1) Alias (Type)

Bahasa pemrograman pada umumnya mengizinkan pemrograman untuk merubah nama suatu tipe data dengan nama baru yang lebih ringkas dan familiar. Sebagai contoh "**integer**" dapat dirubah dengan nama alias "**bilangan**". Caranya dengan menggunakan kata kunci "**type**".

	Notasi Algoritma	Notasi dalam bahasa Go
1	kamus	package main
2	type <nama alias> <tipe data>	type <nama alias> <tipe data>
3		
4	algoritma	func main(){
5	...	...
6		
7		}

Sebagai contoh perhatikan program Go berikut beserta hasil eksekusinya!

```
1 package main
2 import "fmt"
3 type bilangan int
4 type pecahan float64
5 func main(){
6     var a,b bilangan
7     var hasil pecahan
8     a = 9
9     b = 5
10    hasil = pecahan(a) / pecahan(b)
11    fmt.Println(hasil)
12 }
```

```
E:\DEV\GO>go build Demo.go
E:\DEV\GO> Demo.exe
1.8
```

## 2) Structure atau Record

Structure memungkinkan pemrograman untuk mengelompokkan beberapa data atau nilai yang memiliki relasi atau keterkaitan tertentu menjadi suatu kesatuan. Masing-masing nilai tersimpan dalam field dari structure tersebut.

	Notasi Algoritma	Notasi dalam bahasa Go
1	kamus	package main
2	type <nama structure> <	type <nama structure> struct {
3	<field 1> <tipe data>	<field 1> <tipe data>
4	<field 2> <tipe data>	<field 2> <tipe data>
5	<field 3> <tipe data>	<field 3> <tipe data>
6	>	}
7		

Berbeda dengan bahasa pemrograman lain, kesamaan tipe dari dua variabel berjenis structure 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 structure tersebut didefinisikan sebagai sebuah tipe baru, sehingga deklarasi structure tidak perlu lagi seluruh field-nya ditulis ulang berkali-kali.

```
1 package main
2 import "fmt"
3 type waktu struct {
4     jam, menit, detik int
5 }
6
7 func main(){
8     var wParkir, wPulang, durasi waktu
9     var dParkir, dPulang, lParkir int
10    fmt.Scan(&wParkir.jam, &wParkir.menit, &wParkir.detik)
11    fmt.Scan(&wPulang.jam, &wPulang.menit, &wPulang.detik)
12    dParkir = wParkir.detik + wParkir.menit*60 + wParkir.jam*3600
13    dPulang = wPulang.detik + wPulang.menit*60 + wPulang.jam*3600
14    lParkir = dPulang - dParkir
15    durasi.jam = lParkir / 3600
16    durasi.menit = lParkir % 3600 / 60
17    durasi.detik = lParkir % 3600 % 60
18    fmt.Printf("Lama parkir: %d jam %d menit %d detik",
19                durasi.jam, durasi.menit, durasi.detik)
20 }
```

```
E:\DEV\GO>go build Demo.go
E:\DEV\GO> Demo.exe
7 30 0
```

```
10 45 15
Lama parkir: 3 jam 15 menit 15 detik
```

## 9.2. Array

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

Notasi dalam bahasa Go	
1	var (
2	// array arr mempunyai 73 elemen, masing-masing bertipe CircType2
3	arr [73]CircType
4	
5	// array buf dengan 5 elemen, dengan nilai awal 7, 3, 5, 2, dan 11.
6	buf = [5]byte{7, 3, 5, 2, 11}
7	
8	// mhs adalah array dengan 2000 elemen bertipe NewType
9	mhs [2000]NewType
10	
11	// rec adalah array dari array, yaitu matriks, atau array berdimensi-2
12	rec [20][40]float64
13	)

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

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

Contoh:

```
1 // Mengganti isi elemen ke-0 dengan nilai dari elemen ke-7
2 arr[0] = arr[7]
3
4 // Mengambil data field x dari elemen ke-i
5 currX = arr[i].center.x
6
7 // Mengambil elemen terakhir
8 n := len(arr)
9 buf := arr[n-1]
```

### Slice (Array dinamik)

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

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

```
3
4 // declaring sl01 as a slice
5 var sl01 = []int{ 11, 2, 3, 5, 7, 13 }
```

Sebuah slice dapat diprealokasi menggunakan fungsi built-in **make**

```
1 // Preallocasi 10 elemen untuk sl02 dan sejumlah tempat tambahan
2 var sl02 []int = make([]int, 10, 20)
3
4 // Preallocasi 7 elemen untuk sl03 tanpa tempat tambahan
5 var sl03 []circType = make([]circType, 7)
```

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

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

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

```
1 /* 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. */
2 sl01 = append(sl01, 17)
3 sl01 = append(sl01, 19, 23)
```

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

```
1 // Ambil 3 elemen pertama dari suatu slice atau array
2 sl04 = arr[:4]
3
4 // Ambil beberapa elemen terakhir, dimulai dari indeks 5
5 sl05 = sl01[5:]
6
7 // Salin semua dari slice/array aslinya
8 sl06 = sl05[:]
9
10 // Salin element dari indeks 3 sampai, tapi tidak termasuk, 5.
11 // Jadi dalam contoh hanya 2 elemen sl06[3] dan sl06[4] yang disalin
12 sl07 = sl06[3:5]
```

## Map

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

```
1 // Deklarasi variabel dct sebagai map bilangan bulat dengan kunci string
2 var dct map[string]int
3
4 // Deklarasi map lain dct1 dari elemen string dengan kunci juga string
5 // Mempunyai nilai awal dct1["john"] = "hi", dct1["anne"] = "darling"
```

```

6 var dct1 = map[string]string{ "john":"hi", "anne":"darling" }
7
8 // Deklarasi dan prealokasi tempat untuk map dct2
9 var dct2 map[float64]int = make(map[float64]int, 10)
10
11 // Mengambil nilai yang tersimpan dengan kunci "john"
12 fmt.Println( dct1["john"] )
13
14 // Mengganti nilai yang tersimpan pada kunci "anne", dan
15 // Membuat entri baru dengan kunci "boy"
16 dct1["anne"] = "lovely"
17 dct1["boy"] = "runaround"
18
19 // Menghapus entri dengan kunci "john"
20 delete(dct1, "john")

```

### 9.3. Soal Latihan Modul 9

- 1) Suatu lingkaran didefinisikan dengan koordinat titik pusat  $(cx, cy)$  dengan radius  $r$ . Apabila diberikan dua buah lingkaran, maka tentukan posisi sebuah titik sembarang  $(x, y)$  berdasarkan dua lingkaran tersebut. **Gunakan tipe bentukan titik untuk menyimpan koordinat, dan tipe bentukan lingkaran untuk menyimpan titik pusat lingkaran dan radiusnya.**

**Masukan** terdiri dari beberapa tiga baris. Baris pertama dan kedua adalah koordinat titik pusat dan radius dari lingkaran 1 dan lingkaran 2, sedangkan baris ketiga adalah koordinat titik sembarang. Asumsi sumbu x dan y dari semua titik dan juga radius direpresentasikan dengan bilangan bulat.

**Keluaran** berupa string yang menyatakan posisi titik "**Titik di dalam lingkaran 1 dan 2**", "**Titik di dalam lingkaran 1**", "**Titik di dalam lingkaran 2**", atau "**Titik di luar lingkaran 1 dan 2**".

#### Contoh

No	Masukan	Keluaran
1	1 1 5 8 8 4 2 2	Titik di dalam lingkaran 1
2	1 2 3 4 5 6 7 8	Titik di dalam lingkaran 2
3	5 10 15 -15 4 20 0 0	Titik di dalam lingkaran 1 dan 2

4	1 1 5 8 8 4 15 20	Titik di luar lingkaran 1 dan 2
---	-------------------------	---------------------------------

Fungsi untuk menghitung jarak titik (a, b) dan (c, d) dimana rumus jarak adalah:

$$\text{jarak} = \sqrt{(a - c)^2 + (b - d)^2}$$

dan juga fungsi untuk menentukan posisi sebuah titik sembarang berada di dalam suatu lingkaran atau tidak.

```
function jarak(p, q : titik) -> real
{Mengembalikan jarak antara titik p(x,y) dan titik q(x,y)}

function didalam(c:lingkaran, p:titik) -> boolean
{Mengembalikan true apabila titik p(x,y) berada di dalam lingkaran c yang
memiliki titik pusat (cx,cy) dan radius r}
```

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

- 2) Sebuah array digunakan untuk menampung sekumpulan bilangan bulat. Buatlah program yang digunakan untuk mengisi array tersebut sebanyak N elemen nilai. Asumsikan array memiliki kapasitas penyimpanan data sejumlah elemen tertentu. Program dapat menampilkan beberapa informasi berikut:
- Menampilkan keseluruhan isi dari array.
  - Menampilkan elemen-elemen array dengan indeks ganjil saja.
  - Menampilkan elemen-elemen array dengan indeks genap saja (asumsi indek ke-0 adalah genap).
  - Menampilkan elemen-elemen array dengan indeks kelipatan bilangan x, di mana x bisa diperoleh dari masukan pengguna.
  - Menghapus elemen array pada indeks tertentu, asumsi indeks yang hapus selalu valid. Tampilkan keseluruhan isi dari arraynya, pastikan data yang dihapus tidak tampil
  - Menampilkan rata-rata dari bilangan yang ada di dalam array.
  - Menampilkan standar deviasi atau simpangan baku dari bilangan yang ada di dalam array tersebut.

- h. Menampilkan frekuensi dari suatu bilangan tertentu di dalam array yang telah diisi tersebut.
- 3) 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
```

- 4) 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!

```
package main
import "fmt"
const NMAX int = 127
type tabel [NMAX]rune
    tab : tabel
    m : integer

func isiArray(t *tabel, n *int)
/*I.S. Data tersedia dalam piranti masukan
 F.S. Array t berisi sejumlah n karakter yang dimasukkan user,
 Proses input selama karakter bukanlah TITIK dan n <= NMAX */

func cetakArray(t tabel, n int)
/*I.S. Terdefinisi array t yang berisi sejumlah n karakter
 F.S. n karakter dalam array muncul di layar */

func balikanArray(t *tabel, n int)
/*I.S. Terdefinisi array t yang berisi sejumlah n karakter
 F.S. Urutan isi array t terbalik */

func main(){
    var tab tabel
    var m int
    // si 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	I	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.

```
func palindrom(t tabel, n int) bool
/* Mengembalikan true apabila susunan karakter di dalam 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	I	A	K
Palindrom	?	true				

Teks Palindrom	:	S	E	N	A	N	G
		? false					



## MODUL 10. PENCARIAN NILAI EKSTRIM

### 10.1. Ide Pencarian Nilai Ekstrim

Pencarian adalah suatu proses yang lazim dilakukan di dalam kehidupan sehari-hari. Contoh penggunaannya dalam kehidupan nyata sangat beragam, misalnya pencarian file di dalam directory komputer, pencarian suatu teks di dalam sebuah dokumen, pencarian buku pada rak buku, dan contoh lainnya. Pertama pada modul ini akan dipelajari salah satu algoritma pencarian nilai terkecil atau terbesar pada sekumpulan data.

Ide algoritma sederhana sekali. Karena data harus diproses secara sekuisial, 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. Adapun algoritmanya secara umum adalah sebagai berikut:

- 1) Jadikan data pertama sebagai nilai ekstrim
- 2) Lakukan validasi nilai ekstrim dari data kedua hingga data terakhir.
  - Apabila nilai ekstrim tidak valid, maka update nilai ekstrims tersebut dengan data yang dicek.
- 3) Apabila semua data telah dicek, maka nilai ekstrim yang dimiliki adalah valid.

Berikut ini adalah notasi dalam pseudocode dan bahasa Go, misalnya untuk pencarian nilai terbesar atau maksimum:

	Notasi Algoritma	Notasi dalam bahasa Go
1	max ← 1	max = 0
2	i ← 2	i = 1
3	while i <= n do	for i < n {
4	if a[i] > a[max] then	if a[i] > a[max] {
5	max ← i	max = i
6	endif	}
7	i ← i + 1	i = i + 1
8	endwhile	}

### 10.2. Pencarian Nilai Ekstrim pada Array Bertipe Data Dasar

Misalnya terdefinisi sebuah array of integer dengan kapasitas 2023, dan array terisi sejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut. Perhatikan potongan program dalam bahasa Go berikut ini!

```

.. ...
5 type arrInt [2023]int
..
15
16 func terkecil_1(tabInt arrInt, n int) int {
17 /* mengembalikan nilai terkecil yang terdapat di dalam tabInt yang berisi n
18 bilangan bulat */
19     var min int = tabInt[0]           // min berisi data pertama
20     var j int = 1                   // pencarian dimulai dari data berikutnya
21     for j < n {
22         if min > tabInt[j] {       // pengecekan apakah nilai minimum valid
23             min = tabInt[j]        // update nilai minimum dengan yang valid
24         }
25         j = j + 1
26     }
27     return min                     // returnkan nilai minimumnya
28 }
```

Potongan program di atas sedikit berbeda dengan sebelumnya karena penggunaan indeks array pada bahasa Go di mulai dari nol atau "0" seperti penjelasan pada modul 9. Selanjutnya, pada penjelasan di awal bab 3 telah disampaikan bahwa pada pencarian yang terpenting adalah posisi atau indeks dari nilai yang dicari dalam kumpulan data atau array. Oleh karena itu modifikasi pada program di atas dapat dilihat pada potongan program berikut ini!

```

.. ...
5 type arrInt [2023]int
..
15
16 func terkecil_2(tabInt arrInt, n int) int {
17 /* mengembalikan indeks nilai terkecil yang terdapat di dalam tabInt yang berisi n
18 bilangan bulat */
19     var idx int = 0                 // idx berisi indeks data pertama
20     var j int = 1                   // pencarian dimulai dari data berikutnya
21     for j < n {
22         if tabInt[idx] > tabInt[j] { // pengecekan apakah nilai minimum valid
23             idx = j                  // update nilai minimum dengan yang valid
24         }
25         j = j + 1
26     }
27     return idx                     // returnkan indeks nilai minimumnya
28 }
```

### 10.3. Pencarian Nilai Ekstrim pada Array Bertipe Data Bentukan

Pada kasus yang lebih kompleks pencarian ekstrim dapat juga dilakukan, misalnya mencari data mahasiswa dengan nilai terbesar, mencari lagu dengan durasi terlama, mencari pembalap yang memiliki catatan waktu balap tercepat, dan sebagainya. Sebagai contoh misalnya terdapat array yang digunakan untuk menyimpan data mahasiswa, kemudian terdapat fungsi IPK yang digunakan untuk mencari data mahasiswa dengan IPK tertinggi.

```

.. ...
5 type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
.. ...
15
16 func IPK_1(T arrMhs, n int) float64 {
17 /* mengembalikan ipk terkecil yang dimiliki mahasiswa pada array T yang berisi
18 n mahasiswa */
19     var tertinggi float64 = T[0].ipk
20     var j int = 1
21     for j < n {
22         if tertinggi < T[j].ipk {
23             tertinggi = T[j].ipk
24         }
25         j = j + 1
26     }
27     return tertinggi
28 }
```

Apabila diperhatikan potongan program di atas, maka kita akan memperoleh ipk tertinggi, tetapi kita tidak memperoleh identitas mahasiswa dengan ipk tertinggi tersebut. Maka seperti penjelasan yang sudah diberikan sebelumnya, maka pencarian yang dilakukan bisa mengembalikan indeks mahasiswa dengan ipk tertinggi tersebut. Berikut ini adalah modifikasinya!

```

.. ...
5 type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
.. ...
15
16 func IPK_2(T arrMhs, n int) int {
17 /* mengembalikan indeks mahasiswa yang memiliki ipk tertinggi pada array T yang
18 berisi n mahasiswa */
19     var idx int = 0
20     var j int = 1
21     for j < n {
22         if T[idx].ipk < T[j].ipk {
23             idx = j
24         }
25         j = j + 1
26     }
27     return idx
28 }
```

Sehingga melalui algoritma di atas, identitas mahasiswa dapat diperoleh, misalnya `T[idx].nama`, `T[idx].nim`, `T[idx].kelas`, hingga `T[idx].jurusan`.

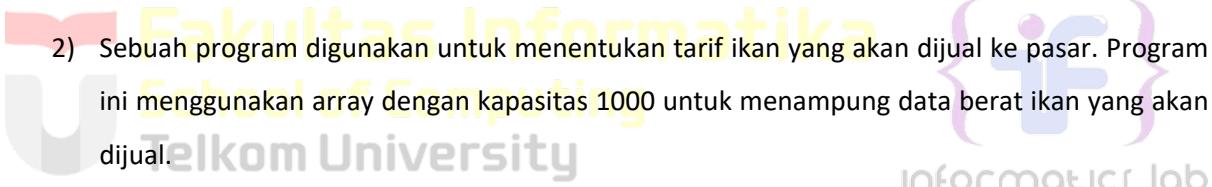
#### 10.4. Soal Latihan Modul 10

Berikut adalah contoh soal yang melibatkan pencarian nilai ekstrim pada tipe data dasar maupun terstruktur.

- 1) Sebuah program digunakan untuk mendata berat anak kelinci yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat anak kelinci yang akan dijual.

**Masukan** terdiri dari sekumpulan bilangan, yang mana bilangan pertama adalah bilangan bulat  $N$  yang menyatakan banyaknya anak kelinci yang akan ditimbang beratnya. Selanjutnya  $N$  bilangan riil berikutnya adalah berat dari anak kelinci yang akan dijual.

**Keluaran** terdiri dari dua buah bilangan riil yang menyatakan berat kelinci terkecil dan terbesar.

- 
- 2) Sebuah program digunakan untuk menentukan tarif ikan yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat ikan yang akan dijual.

**Masukan** terdiri dari dua baris, yang mana baris pertama terdiri dari dua bilangan bulat  $x$  dan  $y$ . Bilangan  $x$  menyatakan banyaknya ikan yang akan dijual, sedangkan  $y$  adalah banyaknya ikan yang akan dimasukan ke dalam wadah. Baris kedua terdiri dari sejumlah  $x$  bilangan riil yang menyatakan banyaknya ikan yang akan dijual.

**Keluaran** terdiri dari dua baris. Baris pertama adalah kumpulan bilangan riil yang menyatakan total berat ikan di setiap wadah (jumlah wadah tergantung pada nilai  $x$  dan  $y$ , urutan ikan yang dimasukan ke dalam wadah sesuai urutan pada masukan baris ke-2). Baris kedua adalah sebuah bilangan riil yang menyatakan berat rata-rata ikan di setiap wadah.

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

Buatlah program dengan spesifikasi subprogram sebagai berikut:

```
type arrBalita [100]float64
```

```
func hitungMinMax(arrBerat arrBalita; bMin, bMax *float64) {  
    /* 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 arrBalita) 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
```



## MODUL 11. PENCARIAN NILAI TERTENTU

Berikut ini merupakan materi lanjutan untuk kasus pencarian data. Selain pencarian nilai ekstrim, kita terkadang juga mencari suatu data yang lebih spesifik. Misalnya mencari mahasiswa dengan nama atau nim tertentu, mencari rumah dengan alamat tertentu, dan lainnya. Berbeda dengan pencarian nilai ekstrim, yang mana nilai yang dicari selalu ditemukan, maka pada kasus pencarian ini terdapat kemungkinan bahwa data yang dicari tidak ditemukan. Selain itu pada kasus pencarian ini akan diperkenalkan algoritma pencarian yang memanfaatkan keterurutan data.

### 11.1. 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 data 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 atau data terakhir.

- 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 T[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.

Berikut ini adalah contoh notasi algoritma **sekuensial search** sesuai dengan penjelasan di atas.

	Notasi Algoritma	Notasi dalam bahasa Go
1	found $\leftarrow$ false	found = false
2	i $\leftarrow$ 0	i = 0
3	while i < n and not found do	for i < n && !found {
4	found $\leftarrow$ T[i] == X	found = T[i] == X
5	i $\leftarrow$ i + 1	i = i + 1
6	endwhile	}

Adapun contoh potongan program pencarian sebuah string pada array of string adalah sebagai berikut ini:

```
.. ...
5 type arrStr [1234]string
..
15
16 func SeqSearch_1(T arrStr, n int, X string) bool {
17 /* mengembalikan true apabila X ditemukan di dalam array T yang berisi n buah
18 teks, atau false apabila X tidak ditemukan */
19 var found bool = false
20 var j int = 0
21 for j < n && !found {
22     found = T[j] == X
23     j = j + 1
24 }
25 return found
26 }
```

Seperti penjelasan yang sudah diberikan sebelumnya, bahwa pada pencarian indeks atau lokasi dari data yang dicari lebih penting dibandingkan data itu sendiri. Oleh karena itu algoritma pencarian di atas bisa dimodifikasi untuk menghasilkan indeks dari data yang dicari. Selain itu perlu dipersiapkan juga sebuah nilai untuk menyatakan apabila pencarian data tidak ditemukan, biasanya -1 atau bilangan di luar indeks dari array yang valid.

```
.. ...
5 type arrStr [1234]string
..
15
16 func SeqSearch_1(T arrStr, n int, X string) int {
17 /* mengembalikan indeks dari X apabila X ditemukan di dalam array T yang berisi
18 n buah teks, atau -1 apabila X tidak ditemukan */
19 var found int = -1
20 var j int = 0
21 for j < n && found == -1 {
22     if T[j] == X {
23         found = j
24     }
25     j = j + 1
26 }
27 return found
28 }
```

### Variasi yang lain

```
.. ...
5 type arrStr [1234]string
..
15
16 func SeqSearch_1(T arrStr, n int, X string) int {
```

```

18 /* mengembalikan indeks dari X apabila X ditemukan di dalam array T yang berisi
19 n buah teks, atau -1 apabila X tidak ditemukan */
20 var j int = 0
21 for j < n-1 && T[j] != X {
22     j = j + 1
23 }
24 if T[j] == X {
25     return j
26 }else{
27     return -1
28 }
29 }

```

## 11.2. Pencarian Biner/Belah Tengah

Ide algoritma adalah: (dengan asumsi data terurut dari kecil membesar (*ascending*), 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 di bawah menggunakan rentang dari kiri ***kr*** s.d. kanan ***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.

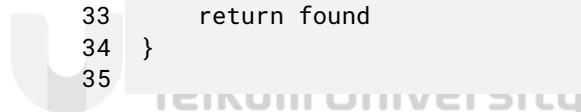
Algoritma ini dikenal dengan nama ***Binary Search***. Berikut ini adalah contoh notasi algoritma *binary search* sesuai dengan penjelasan di atas.

	Notasi Algoritma	Notasi dalam bahasa Go
1	<code>kr &lt; 0</code>	<code>kr = 0</code>
2	<code>kn &lt; n-1</code>	<code>kn = n-1</code>
3	<code>found &lt; false</code>	<code>found = false</code>
4	<code>while kr &lt;= kn and not found do</code>	<code>for kr &lt;= kn &amp;&amp; !found {</code>
5	<code>    med &lt;- (kr+kn) div 2</code>	<code>    med = (kr+kn) / 2</code>
6	<code>    if a[med] &lt; X then</code>	<code>    if a[med] &lt; X {</code>
7	<code>        kr &lt;- med + 1</code>	<code>        kr = med + 1</code>
8	<code>    else if a[med] &gt; X then</code>	<code>    } else if a[med] &gt; X {</code>
9	<code>        kn &lt;- med - 1</code>	<code>        kn = med</code>
10	<code>    else</code>	<code>    } else {</code>
11	<code>        found &lt;- true</code>	<code>        found = true</code>
12	<code>    endif</code>	<code>    }</code>
13	<code>endwhile</code>	

Sebagai catatan algoritma *binary search* untuk array terurut membesar (*ascending*) akan berbeda dengan terurut mengecil (*descending*), sehingga algoritma ini tidak akan berjalan apabila terbalik.

Misalnya array terurut secara membesar, tetapi algoritma binary search yang digunakan untuk array terurut mengecil. Hal ini mengakibatkan pencarian binary search tidak akan berhasil. Selanjutnya bagaimana contoh algoritma pencarian data apabila dituliskan ke dalam suatu fungsi pencarian dan diketahui array terurut secara mengecil atau descending.

```
.. ...
5 type arrInt [4321]int
..
15 func BinarySearch_1(T arrInt, n int, X string) bool {
16 /* mengembalikan true apabila X ditemukan di dalam array T yang berisi n buah
17 bilangan bulat terurut secara descending/mengecil, atau false apabila X tidak
18 ditemukan */
19     var found bool = false
20     var med int
21     var kr int = 0
22     var kn int = n - 1
23     for kr <= kn && !found {
24         med = (kr + kn) / 2
25         if X > T[med] {
26             kn = med - 1
27         }else if X < T[med] {
28             kr = med + 1
29         }else{
30             found = true
31         }
32     }
33     return found
34 }
```



informatics lab

Adapun variasi lain yang mengembalikan indeks dari data yang dicari adalah sebagai berikut:

```
.. ...
5 type arrInt [4321]int
..
15 func BinarySearch_2(T arrInt, n int, X string) int {
16 /* mengembalikan indeks dari X apabila X ditemukan di dalam array T yang berisi
17 n buah bilangan bulat terurut secara descending/mengecil, atau -1 apabila X
18 tidak ditemukan */
19     var found int = -1
20     var med int
21     var kr int = 0
22     var kn int = n - 1
23     for kr <= kn && found == -1 {
24         med = (kr + kn) / 2
25         if X > T[med] {
26             kn = med - 1
27         }else if X < T[med] {
28             kr = med + 1
29         }else{
30             found = med
31         }
32     }
33     return found
34 }
```

Proses belah tengah akan berakhir apabila nilai kr > kn (data tidak ditemukan) atau found sudah tidak bernilai false atau -1 (data ditemukan).

### 11.3. Pencarian pada Array Bertipe Data Bentukan

Algoritma pencarian pada array bertipe data bentukan tidak berbeda jauh dengan tipe data dasar, kita hanya perlu menambahkan field kategori dari pencarian yang akan dilakukan. Khusus untuk algoritma *binary search*, maka keterurutan array harus sama dengan field kategori dari pencariannya. Misalnya apabila pencarian berdasarkan nim mahasiswa, maka algoritma binary search hanya akan bisa digunakan apabila array terurut berdasarkan nim. Algoritma binary search tidak akan berjalan apabila array terurut berdasarkan nim mahasiswa, tetapi pencarian yang dilakukan berdasarkan filed nama atau kategori yang lain selain nim. Solusinya adalah array harus diurutkan berdasarkan nim terlebih dahulu, atau gunakan algoritma *sequential search*.

Barikut adalah contoh apabila array mahasiswa terurut berdasarkan nim secara membesar (ascending) dan dilakukan pencarian menggunakan algoritma sekuensial dan biner.

```
.. ...
5 type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
..
15 func SeqSearch_3(T arrMhs, n int, X string) int {
16 /* mengembalikan indeks mahasiswa dengan nama X, atau -1 apabila tidak ditemukan
17 pada array T yang berisi n data mahasiswa */
18     var found int = -1
19     var j int = 0
20     for j < n && found == -1 {
21         if T[j].nama == X {
22             found = j
23         }
24         j = j + 1
25     }
26     return found
27 }
28
29 func BinarySearch_3(T arrMhs, n int, X string) int {
30 /* mengembalikan indeks mahasiswa dengan nim X, atau -1 apabila tidak ditemukan
31 pada array T yang berisi n data mahasiswa dan terurut membesar berdasarkan nim
32 */
33     var found int = -1
34     var med int
35     var kr int = 0
36     var kn int = n - 1
37     for kr <= kn && found == -1 {
38         med = (kr + kn) / 2
39         if X < T[med].nim {
40             kn = med - 1
41         }else if X > T[med].nim {
```

```

42         kr = med + 1
43     }else{
44         found = med
45     }
46 }
47 return found
48 }
```

#### 11.4. Soal Latihan Modul 11

- 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 di luar yang tersedia, sehingga data juga harus divalidasi. Tugas Anda untuk membuat program mencari siapa yang memenangkan pemilihan ketua RT.

Buatlah program **pilkart** 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 di antara 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.

No	Masukan	Keluaran
1	7 19 3 2 78 3 1 -3 18 19 0	Suara masuk: 10 Suara sah: 8 1: 1 2: 1 3: 2 7: 1 18: 1 19: 2

- 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 di antara 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.

No	Masukan	Keluaran
1	7 19 3 2 78 3 1 -3 18 19 0	Suara masuk: 10 Suara sah: 8 Ketua RT: 3 Wakil ketua: 19

- 3) 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".

**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.

```
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){
/* I.S. terdefinisi integer n, dan sejumlah n data sudah siap pada piranti
masukan.
   F.S. 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 */
}

```

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran	Penjelasan
1	12 534 1 3 8 16 32 123 323 323 534 543 823 999	8	Data 534 berada pada posisi ke-8 dihitung dari awal data.
2	12 535 1 3 8 16 32 123 323 323 534 543 823 999	TIDAK ADA	



## MODUL 12. ASESMEN PRAKTIKUM 2

### 12.1. Contoh Soal Asesmen Praktikum 2

Modul ini berisi contoh soal asesmen praktikum 2 dengan materi pencarian data pada tipe data terstruktur.

- 1) Sebuah program yang digunakan untuk mencari sebuah irisan himpunan.

**Masukan** terdiri dari dua baris, di mana setiap barisnya yang berisi sekumpulan bilangan.

Masukan disetiap barisnya akan berakhir apabila bilangan yang diberikan sudah pernah diberikan pada baris tersebut (atau duplikat). Catatan: anggota suatu himpunan tidak boleh duplikat.

**Keluaran** adalah sekumpulan bilangan yang menyatakan irisan dari himpunan pada baris pertama dan baris kedua pada masukan.

**Contoh masukan dan keluaran:**

No.	Masukan	Keluaran	Penjelasan
1	11 28 33 64 95 16 100 15 64 3 11 7 28 33 6 28	11 28 33	64 dan 28 duplikat sehingga masukan berhenti dan tidak perlu dimasukkan ke dalam array
2	1 1 1 1	1	
3	1 2 3 4 3 9 8 7 9		

```
package main
import "fmt"

type set [2022]int

func exist(T set, n int, val int) bool
/* mengembalikan true apabila bilangan val ada di dalam array T yang berisi sejumlah n bilangan bulat */

func inputSet(T *set, n *int)
/* I.S. data himpunan telah siap pada piranti masukan
   F.S. array T berisi sejumlah n bilangan bulat yang berasal dari masukan (masukan berakhir apabila bilangan ada yang duplikat, atau array penuh)
   Catatan: Panggil function exist di sini untuk membantu pengecekan */

func findIntersection(T1,T2 set, n,m int, T3 *set, h *int)
/* I.S. terdefinisi himpunan T1 dan T2 yang berisi sejumlah n dan m anggota himpunan
   F.S. himpunan T3 berisi sejumlah h bilangan bulat yang merupakan irisan dari himpunan T1 dan T2
   Catatan: Panggil function exist di sini untuk membantu pengecekan */

func printSet(T set, n int)
```

```

/* I.S. terdefinisi sebuah himpunan T yang berisi sejumlah n bilangan bulat
F.S. menampilkan isi array T secara horizontal (dipisahkan oleh spasi) */

func main(){
    var s1,s2,s3 set
    var n1,n2,n3 int
    inputSet(&s1,&n1)
    inputSet(&s2,&n2)
    findIntersection(s1,s2,n1,n2,&s3,&n3)
    printSet(s3,n3)
}

```

- 2) Suatu tabel digunakan untuk mencatat data mahasiswa. Mahasiswa memiliki atribut NIM, nama, dan nilai. Setiap data baru akan selalu ditambahkan ke dalam tabel di indeks N+1, di mana N adalah jumlah data saat ini di dalam array. Sehingga pada tabel mungkin terdapat beberapa data untuk seorang mahasiswa. Contoh isi tabel sebagai berikut:

114, Nana, 97	113, Jojo, 70	118, Rere, 88	116, Koko, 40	117, Keke, 90	116, Koko, 60	113, Jojo, 50	113, Jojo, 80	118, Rere, 88	119, Roro, 100
---------------------	---------------------	---------------------	---------------------	---------------------	---------------------	---------------------	---------------------	---------------------	----------------------

Pada contoh di atas, data Jojo ada tiga dengan 70 sebagai nilai pertama, kemudian 50, dan 80 sebagai nilai terakhir.



Definiskan tipe bentukan dan array berikut:

```

constant nMax: integer = 51
type mahasiswa <NIM: string, nama:string, nilai:integer>
type arrayMahasiswa: array [1..nMax] of mahasiswa

```

Kemudian buatlah program dengan spesifikasi berikut:

- Menerima masukan sejumlah N data mahasiswa kemudian menyimpannya ke dalam array. N diperoleh dari masukan.
  - Fungsi untuk mencari nilai pertama seorang mahasiswa dengan NIM tertentu.
  - Fungsi untuk mencari nilai terbesar seorang mahasiswa dengan NIM tertentu.
  - Menampilkan hasil pencarian pada poin a dan b.
- 3) Sebuah program digunakan untuk mengolah data nama provinsi, populasi, dan angka pertumbuhan penduduk provinsi di Indonesia pada tahun 2018.

**Masukan** terdiri dari 35 baris, yang mana masing-masing barisnya terdiri dari tiga nilai yang menyatakan nama provinsi, jumlah populasi provinsi (bilangan bulat), dan angka pertumbuhan (riil) provinsi tersebut. Pada baris terakhir hanya sebuah string yang menyatakan nama provinsi yang akan dicari.

**Keluaran** terdiri dari 36 baris. Baris pertama adalah nama provinsi dengan angka pertumbuhan tercepat. Baris kedua adalah indeks provinsi yang dicari sesuai dengan nama provinsi yang ditulis pada masukan baris terakhir. Terakhir terdiri dari 34 baris yang menampilkan nama provinsi beserta prediksi jumlah penduduk pada provinsi tersebut di tahun depannya, khusus yang memiliki pertumbuhan di atas 2%.

Lengkapi program berikut sesuai dengan spesifikasi dari subprogram yang diberikan.

```
program Provinsi
kamus
    const nProv : integer = 34
    type NamaProv = array [1..nProv] of string
    type PopProv = array [1..nProv] of integer
    type TumbuhProv = array [1..nProv] of real
algoritma
    ...
endprogram

procedure InputData(in/out prov:NamaProv, pop:PopProv, tumbuh:TumbuhProv )
{I.S. Data-data provinsi tersedia pada input device
 F.S. Array prov, pop, dan tumbuh berisi data yang diberikan.}

function ProvinsiTercepat( tumbuh : TumbuhProv ) → integer
{Mengembalikan indeks array tumbuh dengan pertumbuhan penduduk tercepat}

procedure Prediksi (in prov:NamaProv, in pop:PopProv, in tumbuh:TumbuhProv)
{I.S. Tabel prov, pop, dan tumbuh berisi data-data provinsi
 F.S. Tampilan seluruh nama provinsi dan prediksi jumlah penduduknya di tahun depan dengan pertumbuhan diatas 0.02 (atau diatas 2%) }

function IndeksProvinsi(prov:NamaProvinsi, nama:string )→ integer
{Mengembalikan indeks array prov untuk provinsi dengan nama tersebut, dan -1 jika tidak ada provinsi dengan nama tersebut }
```

## 12.2. Review Asesmen Praktikum 2

Modul ini berisi jawaban atas contoh soal asesmen praktikum 2.

- 1) Sebuah program yang digunakan untuk mencari sebuah irisan himpunan.

**Masukan** terdiri dari dua baris, di mana setiap barisnya yang berisi sekumpulan bilangan.

Masukan disetiap barisnya akan berakhir apabila bilangan yang diberikan sudah pernah

diberikan pada baris tersebut (atau duplikat). Catatan: anggota suatu himpunan tidak boleh duplikat.

**Keluaran** adalah sekumpulan bilangan yang menyatakan irisan dari himpunan pada baris pertama dan baris kedua pada masukan.

**Contoh masukan dan keluaran:**

No.	Masukan	Keluaran	Penjelasan
1	11 28 33 64 95 16 100 15 64 3 11 7 28 33 6 28	11 28 33	64 dan 28 duplikat sehingga masukan berhenti dan tidak perlu dimasukkan ke dalam array
2	1 1 1 1	1	
3	1 2 3 4 3 9 8 7 9		

**Jawaban:**

```
1 package main
2 import "fmt"
3
4 type set [2022]int
5
6 func exist(T set, n int, val int) bool {
7     /* mengembalikan true apabila bilangan val ada di dalam array T yang
8      berisi sejumlah n bilangan bulat */
9     var i int = 0
10    var status bool = false
11    for i < n && !status {
12        status = T[i] == val
13        i++
14    }
15    return status
16 }
17
18 func inputSet(T *set, n *int){
19     /* I.S. data himpunan telah siap pada piranti masukan
20      F.S. array T berisi sejumlah n bilangan bulat yang berasal dari
21      masukan (masukan berakhir apabila bilangan ada yang duplikat atau array
22      penuh)
23      Catatan: Panggil function exist di sini untuk membantu pengecekan */
24     *n = 0
25     var bilangan int
26     fmt.Scan(&bilangan)
27     for *n < 2022 && !exist(*T,*n,bilangan) {
28         T[*n] = bilangan
29         *n++
30         fmt.Scan(&bilangan)
31     }
32 }
33
34 func findIntersection(T1,T2 set, n,m int, T3 *set, h *int){
35     /* I.S. terdefinisi himpunan T1 dan T2 yang berisi sejumlah n dan m
36      anggota himpunan
37 
```

```

38     F.S. himpunan T3 berisi sejumlah h bilangan bulat yang merupakan
39     irisan dari himpunan T1 dan T2
40     Catatan: Panggil function exist di sini untuk membantu pengecekan */
41     var j int = 0
42     *h = 0
43     for j < n {
44         if exist(T2,m, T1[j]) {
45             T3[*h] = T1[j]
46             *h++
47         }
48         j++
49     }
50 }
51
52 func printSet(T set, n int){
53 /* I.S. terdefinisi sebuah himpunan T yang berisi sejumlah n bilangan
54 bulat
55     F.S. menampilkan isi array T secara horizontal (dipisahkan oleh spasi)
56 */
57     var i int = 0
58     for i < n {
59         fmt.Print(T[i], " ")
60         i++
61     }
62 }
63
64 func main(){
65     var s1,s2,s3 set
66     var n1,n2,n3 int
67     inputSet(&s1,&n1)
68     inputSet(&s2,&n2)
69     findIntersection(s1,s2,n1,n2,&s3,&n3)
70     printSet(s3,n3)
71 }
72 }
```

- 2) Suatu tabel digunakan untuk mencatat data mahasiswa. Mahasiswa memiliki atribut NIM, nama, dan nilai. Setiap data baru akan selalu ditambahkan ke dalam tabel di indeks N+1, di mana N adalah jumlah data saat ini di dalam array. Sehingga pada tabel mungkin terdapat beberapa data untuk seorang mahasiswa. Contoh isi tabel sebagai berikut:

114, Nana, 97	113, Jojo, 70	118, Rere, 88	116, Koko, 40	117, Keke, 90	116, Koko, 60	113, Jojo, 50	113, Jojo, 80	118, Rere, 88	119, Roro, 100
---------------------	---------------------	---------------------	---------------------	---------------------	---------------------	---------------------	---------------------	---------------------	----------------------

Pada contoh di atas, data Jojo ada tiga dengan 70 sebagai nilai pertama, kemudian 50, dan 80 sebagai nilai terakhir.

Definiskan tipe bentukan dan array berikut:

```

constant nMax: integer = 51
type mahasiswa <NIM: string, nama:string, nilai:integer>
```

```
type arrayMahasiswa: array [1..nMax] of mahasiswa
```

Kemudian buatlah program dengan spesifikasi berikut:

- a. Menerima masukan sejumlah N data mahasiswa kemudian menyimpannya ke dalam array. N diperoleh dari masukan.
- b. Fungsi untuk mencari nilai pertama seorang mahasiswa dengan NIM tertentu.
- c. Fungsi untuk mencari nilai terbesar seorang mahasiswa dengan NIM tertentu.
- d. Menampilkan hasil pencarian pada poin a dan b.

**Jawaban:**

```
1 package main
2 import "fmt"
3
4 const nMax int = 51
5 type mahasiswa struct {
6     NIM string
7     nama string
8     nilai int
9 }
10 type arrayMahasiswa [nMax]mahasiswa
11
12 func main(){
13     var A arrayMahasiswa
14     var M, idx1, idx2 int
15     var NIM string
16     inputMhs(&A, &M)
17     NIM = "113"
18     idx1 = findFirstScore(A,M,NIM)
19     idx2 = findMaxScore(A,M,NIM)
20     fmt.Println("Nilai pertama dari NIM",NIM,"adalah",A[idx1].nilai)
21     fmt.Println("Nilai terbesar dari NIM",NIM,"adalah",A[idx2].nilai)
22 }
23
24 func inputMhs(T *arrayMahasiswa, N *int){
25     /*I.S. Sejumlah N data mahasiswa telah siap pada piranti masukan
26     F.S. N beisi sebuah nilai dan T berisi sejumlah N data mahasiswa*/
27     fmt.Scan(N)
28     var i int = 0
29     for i < *N {
30         fmt.Scan(&T[i].NIM, &T[i].nama, &T[i].nilai)
31         i++
32     }
33 }
34
35 func findFirstScore(T arrayMahasiswa, N int, nim string) int {
36     var i, found int
37     found = -1
38     i = 0
39     for i < N && found == -1 {
```

```

40         if T[i].NIM == nim {
41             found = i
42         }
43         i++
44     }
45     return found
46 }
47
48 func findMaxScore(T arrayMahasiswa, N int, nim string) int {
49     var found int = findFirstScore(T,N,nim)
50     var i, idxMax int
51     if found != -1 {
52         idxMax = found
53         for i = found; i < N; i++ {
54             if T[idxMax].nilai < T[i].nilai && T[i].NIM == nim {
55                 idxMax = i
56             }
57         }
58         return idxMax
59     }else{
60         return found
61     }
62 }
```

- 3) Sebuah program digunakan untuk mengolah data nama provinsi, populasi, dan angka pertumbuhan penduduk provinsi di Indonesia pada tahun 2018.

**Masukan** terdiri dari 35 baris, yang mana masing-masing barisnya terdiri dari tiga nilai yang menyatakan nama provinsi, jumlah populasi provinsi (bilangan bulat), dan angka pertumbuhan (riil) provinsi tersebut. Pada baris terakhir hanya sebuah string yang menyatakan nama provinsi yang akan dicari.

**Keluaran** terdiri dari 36 baris. Baris pertama adalah nama provinsi dengan angka pertumbuhan tercepat. Baris kedua adalah indeks provinsi yang dicari sesuai dengan nama provinsi yang ditulis pada masukan baris terakhir. Terakhir terdiri dari 34 baris yang menampilkan nama provinsi beserta prediksi jumlah penduduk pada provinsi tersebut di tahun depannya, khusus yang memiliki pertumbuhan di atas 2%.

**Jawaban:**

```

1 package main
2 import "fmt"
3 const nProv int = 34
4 type NamaProv [nProv] string
5 type PopProv [nProv] int
6 type TumbuhProv [nProv]float64
7
8 func InputData(prov *NamaProv, pop *PopProv, tumbuh *TumbuhProv ){
9     /*I.S. Data-data provinsi tersedia pada input device
10    F.S. Array prov, pop, dan tumbuh berisi data yang diberikan. */
```

```

11     var i int
12     for i = 0 ; i < nProv; i++{
13         fmt.Scan(&prov[i],&pop[i],&tumbuh[i])
14     }
15 }
16
17 func ProvinsiTercepat(tumbuh TumbuhProv ) int {
18 // Mengembalikan indeks array tumbuh dengan pertumbuhan penduduk tercepat
19     var idx int = 0
20     var i int
21     for i = 1; i < nProv; i++ {
22         if tumbuh[idx] < tumbuh[i] {
23             idx = i
24         }
25     }
26     return idx
27 }
28
29 func Prediksi (prov NamaProv, pop PopProv, tumbuh TumbuhProv){
30 /* I.S. Tabel prov, pop, dan tumbuh berisi data-data provinsi
31 F.S. Tampilan seluruh nama provinsi dan prediksi jumlah penduduknya
32 di tahun depan dengan pertumbuhan di atas 0.02 (atau diatas 2%) */
33     var i int
34     var result float64
35     for i = 0; i < nProv; i++{
36         if tumbuh[i] > 0.02 {
37             result = (tumbuh[i] + 1) * float64(pop[i])
38             fmt.Println(prov[i], result)
39         }
40     }
41 }
42
43 func IndeksProvinsi(prov NamaProv, nama string) int {
44 /* Mengembalikan indeks array prov untuk provinsi dengan nama tersebut,
45 dan -1 jika tidak ada provinsi dengan nama tersebut */
46     var found int = -1
47     var i int = 0
48     for i < nProv && found == -1 {
49         if prov[i] == nama {
50             found = i
51         }
52         i++
53     }
54     return found
55 }
56
57 func main(){
58     var TProvinsi NamaProv
59     var TPopulasi PopProv
60     var TPertumbuhan TumbuhProv
61     var cari string
62     var idxTercepat, idxProvinsi int
63
64     InputData(&TProvinsi, &TPopulasi, &TPertumbuhan)
65     fmt.Scan(&cari)
66
67     idxTercepat = ProvinsiTercepat(TPertumbuhan)
68     fmt.Println(TProvinsi[idxTercepat])

```



```
69  
70     idxProvinsi = IndeksProvinsi(TProvinsi,cari)  
71     fmt.Println(TProvinsi[idxProvinsi])  
72  
73     Prediksi(TProvinsi, TPopulasi, TPertumbuhan)  
74 }  
75
```



## BAB 4. PENGURUTAN PADA DATA TERSTRUKTUR

Pada bab sebelumnya telah dipelajari tipe data terstruktur beserta contoh penggunaannya pada kasus pencarian data, baik itu pencarian nilai ekstrim, pencarian data tertentu pada kumpulan data yang tersusun acak dan juga terurut. Selanjutnya pada bab ini akan dipelajari studi kasus lanjutan yaitu bagaimana cara mengurutkan data baik pada array bertipe data dasar ataupun bentukan. Algoritma pengurutan yang akan diperkenalkan akan memanfaatkan algoritma pencarian yang sudah dipelajari sebelumnya. Terdapat dua algoritma pengurutan yang akan dipelajari, yaitu: (a) Pengurutan Seleksi atau *Selection Sort*; dan (b) Pengurutan Inversi atau *Insertion Sort*.

Pada bab ini terdapat empat modul yang akan membahas:

a) **Modul 13. Pengurutan Data Secara Seleksi**

Modul ini akan membahas terkait ide pengurutan, algoritma pengurutan seleksi, contoh soal dan juga soal latihannya.

b) **Modul 14. Pengurutan Data Secara Inversi**

Pada modul ini akan dibahas ide pengurutan, algoritma pengurutan seleksi, contoh soal dan juga soal latihannya.

c) **Modul 15. Asesmen Praktikum 3**

Modul ini berisi contoh soal asesmen yang berkaitan dengan pengurutan pada data terstruktur.

d) **Modul 16. Review Asesmen Praktikum 3**

Modul ini berisi pembahasan dari contoh soal asesmen praktikum 3.

## MODUL 13. PENGURUTAN DATA SECARA SELEKSI

### 13.1. Ide Algoritma Pencarian Seleksi

Pengurutan secara seleksi ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian meletakkan pada posisi yang seharusnya. Pada penjelasan berikut ini data akan diurut membesar (*ascending*), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

- 1) Cari nilai terkecil di dalam rentang data tersisa
- 2) Pindahkan/tukar tempat dengan data yang berada pada posisi paling kiri pada rentang data tersisa tersebut.
- 3) Ulangi proses ini sampai tersisa hanya satu data saja.

Algoritma ini dikenal juga dengan nama **Selection Sort**, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian indeks nilai ekstrim dan proses pertukaran dua nilai atau *swap*.

	Notasi Algoritma	Notasi dalam bahasa Go
1	i ← 1	i = 1
2	while i <= n-1 do	for i <= n-1 {
3	idx_min ← i - 1	idx_min = i - 1
4	j ← i	j = i
5	while j < n do	for j < n {
6	if a[idx_min] > a[j] then	if a[idx_min] > a[j] {
7	idx_min ← j	idx_min = j
8	endif	}
9	j ← j + 1	j = j + 1
10	endwhile	}
11	t ← a[idx_min]	t = a[idx_min]
12	a[idx_min] ← a[i-1]	a[idx_min] = a[i-1]
13	a[i-1] ← t	a[i-1] = t
14	i ← i + 1	i = i + 1
15	endwhile	}

### 13.2. Algoritma Selection Sort

Adapun algoritma *selection sort* pada untuk mengurutkan array bertipe data bilangan bulat secara membesar atau *ascending* adalah sebagai berikut ini!

```
.. ...
5  type arrInt [4321]int
.. ...
15 func selectionSort1(T *arrInt, n int){
16 /* I.S. terdefinisi array T yang berisi n bilangan bulat
17   F.S. array T terurut secara asceding atau membesar dengan SELECTION SORT */
18   var t, i, j, idx_min int
```

```

19     i = 1
20     for i <= n-1 {
21         idx_min = i - 1
22         j = i
23         for j < n {
24             if T[idx_min] > T[j] {
25                 idx_min = j
26             }
27             j = j + 1
28         }
29         t = T[idx_min]
30         T[idx_min] = T[i-1]
31         T[i-1] = t
32         i = i + 1
33     }
34 }
35 }
```

Sama halnya apabila array yang akan diurutkan adalah bertipe data bentukan, maka tambahkan field pada saat proses perbandingan nilai ekstrim, kemudian tipe data dari variabel *t* sama dengan tipe bentukan dari arraynya.

```

.. ...
5  type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
.. ...
15 func selectionSort2(T * arrMhs, n int){
16 /* I.S. terdefinisi array T yang berisi n data mahasiswa
17 F.S. array T terurut secara ascending atau membesar berdasarkan ipk dengan
18 menggunakan algoritma SELECTION SORT */
19     var i, j, idx_min int
20     var t mahasiswa
21     i = 1
22     for i <= n-1 {
23         idx_min = i - 1
24         j = i
25         for j < n {
26             if T[idx_min].ipk > T[j].ipk {
27                 idx_min = j
28             }
29             j = j + 1
30         }
31         t = T[idx_min]
32         T[idx_min] = T[i-1]
33         T[i-1] = t
34         i = i + 1
35     }
36 }
```

### 13.3. Soal Latihan Modul 13

- 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 di suatu 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 di mana 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) Belakangan diketahui ternyata Hercules itu tidak berani menyeberang jalan, maka selalu diusahakan agar hanya menyeberang jalan sesedikit mungkin, hanya diujung jalan. Karena nomor rumah sisi kiri jalan selalu ganjil dan sisi kanan jalan selalu genap, maka buatlah program **kerabat dekat** 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 7 9 13 6 189 15 27 39 75 133 3 4 9 1	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:**

- Waktu pembacaan data, bilangan ganjil dan genap dipisahkan ke dalam dua array yang berbeda, untuk kemudian masing-masing diurutkan tersendiri.
  - 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) 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 ke bawah."

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 -5313.

**Keluaran** adalah median yang diminta, satu data per baris.

No	Masukan	Keluaran
1	7 23 11 0 5 19 2 29 3 13 17 0 -5313	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 metode insertion sort dan ambil mediannya.



## MODUL 14. PENGURUTAN DATA SECARA INSERSI

### 14.1. Ide Algoritma Pencarian Inversi

Pengurutan secara insersi ini idenya adalah menyisipkan suatu nilai pada posisi yang seharusnya. Berbeda dengan pengurutan seleksi, yang mana pada pengurutan ini tidak dilakukan pencarian nilai ekstrim terlebih dahulu, cukup memilih suatu nilai tertentu kemudian mencari posisinya secara *sequential search*. Pada penjelasan berikut ini data akan diurut mengecil (*descending*), 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.



Algoritma ini dikenal juga dengan nama *Insertion Sort*, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian sekuensial dan penyisipan.

	Notasi Algoritma	Notasi dalam bahasa Go
1	i ← 1	i = 1
2	while i <= n-1 do	for i <= n-1 {
3	j ← i	j = i
4	temp ← a[j]	temp = a[j]
5	while j > 0 and temp > a[j-1] do	for j > 0 && temp > a[j-1] {
6	a[j] ← a[j-1]	a[j] = a[j-1]
7	j ← j - 1	j = j - 1
8	endwhile	}
9	a[j] ← temp	a[j] = temp
10	i ← i + 1	i = i + 1
11	endwhile	}

### 14.2. Algoritma Insertion Sort

Adapun algoritma *insertion sort* pada untuk mengurutkan array bertipe data bilangan bulat secara mengecil atau *descending* adalah sebagai berikut ini!

```

.. ...
5 type arrInt [4321]int
..
15 func insertionSort1(T *arrInt, n int){
16 /* I.S. terdefinisi array T yang berisi n bilangan bulat
17   F.S. array T terurut secara mengecil atau descending dengan INSERTION SORT*/
18   var temp, i, j int
19   i = 1
20   for i <= n-1 {
21       j = i
22       temp = T[j]
23       for j > 0 && temp > T[j-1] {
24           T[j] = T[j-1]
25           j = j - 1
26       }
27       T[j] = temp
28       i = i + 1
29   }
30 }
```

Sama halnya apabila array yang akan diurutkan adalah bertipe data bentukan, maka tambahkan field pada saat proses perbandingan dalam pencarian posisi, kemudian tipe data dari variabel **temp** sama dengan tipe bentukan dari arraynya.

```

.. ...
5 type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
..
15 func insertionSort2(T * arrMhs, n int){
16 /* I.S. terdefinisi array T yang berisi n data mahasiswa
17   F.S. array T terurut secara mengecil atau descending berdasarkan nama dengan
18 menggunakan algoritma INSERTION SORT */
19   var temp i, j int
20   var temp mahasiswa
21   i = 1
22   for i <= n-1 {
23       j = i
24       temp = T[j]
25       for j > 0 && temp.nama > T[j-1].nama {
26           T[j] = T[j-1]
27           j = j - 1
28       }
29       T[j] = temp
30       i = i + 1
31   }
32 }
```

### 14.3. Soal Latihan Modul 14

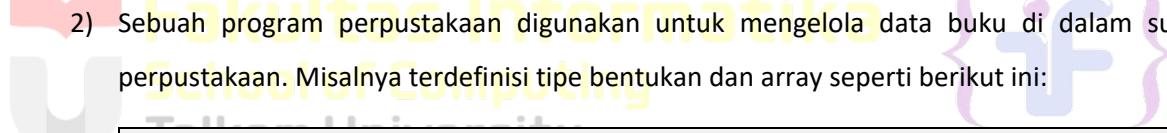
- Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metoda *insertion sort*), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.

**Masukan** terdiri dari sekumpulan bilangan bulat yang diakhiri oleh bilangan negatif. Hanya bilangan non negatif saja yang disimpan ke dalam array.

**Keluaran** terdiri dari dua baris. Baris pertama adalah isi dari array setelah dilakukan pengurutan, sedangkan baris kedua adalah status jarak setiap bilangan yang ada di dalam array. "Data berjarak x" atau "data berjarak tidak tetap".

Contoh masukan dan keluaran

No	Masukan	Keluaran
1	31 13 25 43 1 7 19 37 -5	1 7 13 19 25 31 37 43 Data berjarak 6
2	4 40 14 8 26 1 38 2 32 -31	1 2 4 8 14 26 32 38 40 Data berjarak tidak tetap

- 
- Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Misalnya terdefinisi tipe bentukan dan array seperti berikut ini:

```
const nMax : integer = 7919
type Buku = <
    id, judul, penulis, penerbit : string
    eksemplar, tahun, rating : integer >

type DaftarBuku = array [ 1..nMax] of Buku
Pustaka : DaftarBuku
nPustaka: integer
```

**Masukan** terdiri dari beberapa baris. Baris pertama adalah bilangan bulat N yang menyatakan banyaknya data buku yang ada di dalam perpustakaan. N baris berikutnya, masing-masingnya adalah data buku sesuai dengan atribut atau field pada tipe bentukan. Baris terakhir adalah bilangan bulat yang menyatakan rating buku yang akan dicari.

**Keluaran** terdiri dari beberapa baris. Baris pertama adalah data buku terfavorit, baris kedua adalah lima judul buku dengan rating tertinggi, selanjutnya baris terakhir adalah data buku yang dicari sesuai rating yang diberikan pada masukan baris terakhir.

Lengkapi subprogram-subprogram dibawah ini, sesuai dengan I.S. dan F.S yang diberikan.

```
procedure DaftarkanBuku(in/out pustaka : DaftarBuku, n : integer)
{I.S. sejumlah n data buku telah siap para piranti masukan
 F.S. n berisi sebuah nilai, dan pustaka berisi sejumlah n data buku}

procedure CetakTerfavorit(in pustaka : DaftarBuku, in n : integer)
{I.S. array pustaka berisi n buah data buku dan belum terurut
 F.S. Tampilan data buku (judul, penulis, penerbit, tahun)
 terfavorit, yaitu memiliki rating tertinggi}

procedure UrutBuku( in/out pustaka : DaftarBuku, n : integer )
{I.S. Array pustaka berisi n data buku
 F.S. Array pustaka terurut menurun/mengecil terhadap rating.
 Catatan: Gunakan metoda Insertion sort}

procedure Cetak5Terbaru( in pustaka : DaftarBuku, n integer )
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
 F.S. Laporan 5 judul buku dengan rating tertinggi
 Catatan: Isi pustaka mungkin saja kurang dari 5}

procedure CariBuku(in pustaka : DaftarBuku, n : integer, r : integer )
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
 F.S. Laporan salah satu buku (judul, penulis, penerbit, tahun,
 eksemplar, rating) dengan rating yang diberikan. Jika tidak ada buku
 dengan rating yang ditanyakan, cukup tuliskan "Tidak ada buku dengan
 rating seperti itu". Catatan: Gunakan pencarian biner/belah dua.}
```



School of Computing  
Telkom University



## MODUL 15. ASESMEN PRAKTIKUM 3

### 15.1. Contoh Soal Asesmen Praktikum 3

Modul ini berisi contoh soal asesmen praktikum 3 dengan materi pengurutan data pada tipe data terstruktur.

- 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 menyelesaiakannya, 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.

**Petunjuk:**

- a. Untuk setiap data bukan 0 (dan bukan marker -5313541) simpan ke dalam array.
- b. dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metoda selection sort dan ambil mediannya.

**Contoh masukan dan keluaran**

	<b>Masukan</b>	7 23 11 <b>0</b> 5 19 2 29 3 13 17 <b>0</b> -5313541
	<b>Keluaran</b>	11 12
1	<b>Penjelasan</b>	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 $(11+13)/2=12$ .

	<b>Masukan</b>	23 12 24 26 20 10 25 8 <b>0</b> -5313541
	<b>Keluaran</b>	21.5
2	<b>Penjelasan</b>	Sampai bilangan 0 yang pertama, data terbaca adalah 23 12 24 26 20 10 25 8, setelah tersusun: 8 10 12 20 23 24 25 26, maka median saat itu adalah $(20+23)/2=21.5$ .

```

1 package main
2 import "fmt"
3
4 const NMAX = ...
5 type arrInt [...]int
6
7 func sorting(T *arrInt, n int){
8     /* I.S. terdefinisi array T yang berisi sejumlah n bilangan bulat
9      F.S. array T terurut secara membesar berdasarkan algoritma selection
10     sort */
11     ...
12 }
13
14 func median(T arrInt, n int) float64 {
15     /* mengembalikan median dari array T yang berisi sejumlah n bilangan bulat
16     yang telah terurut membesar */
17     ...
18 }
19
20 func main(){
21     ...
22 }
```

- 2) Di masa pandemi covid-19 ini English Premier League sudah memasuki minggu ke-11. Tanpa kehadiran langsung fans fanatic di stadion, gairah pemain menunjukkan kemampuan mereka secara optimal jauh berkurang. Tidak ada gol-gol spektakular atau permainan menggocek bola yang cantik.

Untuk mendorong gairah pemain, penyelenggara menampilkan peringkat pencetak gol terbanyak sementara. Data itu ingin disusun secara **menurun terhadap perolehan gol dan assist**. Atribut data adalah **nama**, **jumlah gol**, dan **jumlah assist**. Jika ada dua orang pemain atau lebih memiliki jumlah gol yang sama, maka assist yang lebih banyak akan meletakkan pemain tersebut pada peringkat lebih atas.

**Masukan** terdiri atas **n** (tipe data integer) yang menyatakan jumlah pencetak gol, diikuti sebanyak **n** baris berikutnya berupa nama (dua tipe data string), jumlah gol (tipe data integer), dan jumlah assist (tipe data integer).

**Keluaran** terdiri atas data pemain (nama, jumlah goal, jumlah assist) yang tersusun menurun sesuai dengan kriteria di atas. Untuk kebutuhan pembuatan program, algoritma sorting yang

boleh digunakan adalah Selection sort atau Insertion sort. Selain itu anggap bahwa jumlah pemain yang diperhitungkan dalam peringkat tidak lebih dari 1001 orang pemain saja.

**Contoh masukan dan keluaran:**

No	Masukan	Keluaran
1	9 Bruno Fernandes 7 3 Jamie Vardy 8 1 Dominic Calvert-Lewin 10 2 Son Heung-Min 9 2 Harry Kane 7 2 Ollie Watkins 6 1 Patrick Bamford 7 1 Callum Wilson 7 0 Mohamed Salah 8 2	Dominic Calvert-Lewin 10 2 Son Heung-Min 9 2 Mohamed Salah 8 2 Jamie Vardy 8 1 Bruno Fernandes 7 3 Harry Kane 7 2 Patrick Bamford 7 1 Callum Wilson 7 0 Ollie Watkins 6 1
2	7 Bruno Fernandes 7 3 Dominic Calvert-Lewin 7 12 Son Heung-Min 7 2 Harry Kane 5 2 Patrick Bamford 5 1 Callum Wilson 5 0 Mohamed Salah 3 2	DominicCalvert-Lewin 7 12 BrunoFernandes 7 3 SonHeung-Min 7 2 HarryKane 5 2 PatrickBamford 5 1 CallumWilson 5 0 MohamedSalah 3 2

- 3) Sebuah program digunakan untuk menghitung perolehan suara dari berbagai partai politik dalam sebuah pemilihan umum calon legislatif. Program akan menampilkan data partai terurut berdasarkan perolehan suara terurut. Nama partai hanya disimbolkan dari angka 1 hingga N, di mana  $1 \leq N \leq 1000000$ .

**Masukan** berupa beberapa nilai yang dipisahkan oleh spasi. Masing-masing nilai menyatakan nama partai (1 hingga N) yang dipilih. Proses input ini diakhiri dengan nilai -1.

**Keluaran** berupa daftar partai dan peroleh suaranya yang terurut descending atau mengecil dengan format <partai>(<suara>). Perhatikan contoh masukan dan keluaran yang diberikan.

**Petunjuk :** gunakan tipe bentukan partai yang berisi nama dan suara. Data perolehan suara disimpan pada array of partai (kapasitas 1000000). Array tersebutlah yang diurutkan.

**Contoh masukan dan keluaran:**

1	<b>Masukan</b>	5 1 1 1 1 1 1 1 3 3 3 3 3 2 2 5 5 5 5 5 4 3 2 2 2 2 -1
	<b>Keluaran</b>	1(7) 5(6) 3(6) 2(6) 4(1)
2	<b>Masukan</b>	5 8 8 5 6 8 8 7 6 5 8 7 5 6 7 5 8 6 7 8 8 7 7 8 6 7 7 6 8 6 8 8 5 5 6 6 6 7 7 6 7 8 8 8 5 7 6 6 8 6 5 5 8 7 5 5 6 8 7 6 5 5 8 6 6 7 8 -1 8 6 7 6 6 5 7 8 7 6 6 8 7 7 8 6 5 5 7 7 6 5 7 8 8 6 8 8 6 7 8
	<b>Keluaran</b>	8(30) 6(28) 7(24) 5(18)

3	<b>Masukan</b>	8 8 8 8 8 8 8 8 8 8 8 8 8 8 -1
	<b>Keluaran</b>	8(15)
4	<b>Masukan</b>	10 1 7 8 10 1 4 8 8 5 -1
	<b>Keluaran</b>	8(3) 10(2) 1(2) 7(1) 4(1) 5(1)
5	<b>Masukan</b>	14 10 13 13 14 10 11 13 13 12 15 11 10 -1
	<b>Keluaran</b>	13(4) 10(3) 14(2) 11(2) 12(1) 15(1)
6	<b>Masukan</b>	-1
	<b>Keluaran</b>	

### Lengkapi program berikut ini

```

1 package main
2 import "fmt"
3 const NMAX = 1000000
4 // tipe bentukan partai
5 ...
6 // tipe tabPartai: array of partai dengan kapasitas NMAX
7 ...
8 func main(){
9     // deklarasi variabel
10    ...
11    // lakukan proses input suara secara berulang di sini, simpan ke
12    dalam                                array
13    p, sehingga terdapat array p yang berisi hasil peroleh suara n
14    partai.
15    ...
16    // lakukan proses pengurutan dengan insertion sort descending
17    berdasarkan
18        jumlah suara yang diperoleh
19    ...
20    // tampilkan array p
21    ...
22 }
23
24 func posisi(t tabPartai, n int, nama int) int {
25 /* mengembalikan indeks partai yang memiliki nama yang dicari pada array
26 t yang berisi n partai atau -1 apabila tidak ditemukan , gunakan pencarian
27 sekuensial */
28    ...
29 }
```

## MODUL 16. REVIEW ASESMEN PRAKTIKUM 3

### 16.1. Review Soal Asesmen Praktikum 3

Modul ini berisi jawaban atas contoh soal asesmen praktikum 3 yang terdapat pada modul 15.

- 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 menyelesaiakannya, 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.

**Petunjuk:**

- a. Untuk setiap data bukan 0 (dan bukan marker -5313541) simpan ke dalam array.
- b. dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metoda selection sort dan ambil mediannya.

**Contoh masukan dan keluaran**

	<b>Masukan</b>	7 23 11 <b>0</b> 5 19 2 29 3 13 17 <b>0</b> -5313541
	<b>Keluaran</b>	11 12
1	<b>Penjelasan</b>	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 $(11+13)/2=12$ .

	<b>Masukan</b>	23 12 24 26 20 10 25 8 <b>0</b> -5313541
2	<b>Keluaran</b>	21.5
	<b>Penjelasan</b>	Sampai bilangan 0 yang pertama, data terbaca adalah 23 12 24 26 20 10 25 8, setelah tersusun: 8 10 12 20 23 24 25 26, maka median saat itu adalah $(20+23)/2=21.5$ .

### Jawaban:

```

1 package main
2 import "fmt"
3
4 const NMAX = 1000000
5 type arrInt [NMAX]int
6
7 func sorting(T *arrInt, n int){
8     /* I.S. terdefinisi array T yang berisi sejumlah n bilangan bulat
9        F.S. array T terurut secara membesar berdasarkan algoritma selection
10       sort */
11    var pass, idx_min, i, temp int
12    for pass = 1; pass <= n-1; pass++ {
13        idx_min = pass -1
14        for i = pass; i <= n-1; i++ {
15            if T[idx_min] > T[i] {
16                idx_min = i
17            }
18        }
19        temp = T[idx_min]
20        T[idx_min] = T[pass-1]
21        T[pass-1] = temp
22    }
23}
24
25 func median(T arrInt, n int) float64 {
26     /* mengembalikan median dari array T yang berisi sejumlah n bilangan
27        bulat yang telah terurut membesar */
28     var mid int = n / 2
29     if n % 2 == 0 {
30         return float64(T[mid-1] + T[mid]) / 2.0
31     }else{
32         return float64(T[mid])
33     }
34 }
35
36 func main(){
37     var A arrInt
38     var x,n int
39     fmt.Scan(&x)
40     n = 0
41     for x != -5313541 && n < NMAX {
42         if x == 0 {
43             sorting(&A,n)
44             fmt.Println(median(A,n))
45         }else{
46             A[n] = x
47             n++
48         }
49         fmt.Scan(&x)

```

```
50 }  
51 }
```

- 2) Di masa pandemi covid-19 ini English Premier League sudah memasuki minggu ke-11. Tanpa kehadiran langsung fans fanatik di stadion, gairah pemain menunjukkan kemampuan mereka secara optimal jauh berkurang. Tidak ada gol-gol spektakular atau permainan menggocek bola yang cantik.

Untuk mendorong gairah pemain, penyelenggara menampilkan peringkat pencetak gol terbanyak sementara. Data itu ingin disusun **secara menurun terhadap perolehan gol dan assist**. Atribut data adalah **nama**, **jumlah gol**, dan **jumlah assist**. Jika ada dua orang pemain atau lebih memiliki jumlah gol yang sama, maka assist yang lebih banyak akan meletakkan pemain tersebut pada peringkat lebih atas.

**Masukan** terdiri atas **n** (tipe data integer) yang menyatakan jumlah pencetak gol, diikuti sebanyak **n** baris berikutnya berupa nama (dua tipe data string), jumlah gol (tipe data integer), dan jumlah assist (tipe data integer).

**Keluaran** terdiri atas data pemain (nama, jumlah goal, jumlah assist) yang tersusun menurun sesuai dengan kriteria di atas. Untuk kebutuhan pembuatan program, algoritma sorting yang boleh digunakan adalah Selection sort atau Insertion sort. Selain itu anggap bahwa jumlah pemain yang diperhitungkan dalam peringkat tidak lebih dari 1001 orang pemain saja.

#### Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	9 Bruno Fernandes 7 3 Jamie Vardy 8 1 Dominic Calvert-Lewin 10 2 Son Heung-Min 9 2 Harry Kane 7 2 Ollie Watkins 6 1 Patrick Bamford 7 1 Callum Wilson 7 0 Mohamed Salah 8 2	Dominic Calvert-Lewin 10 2 Son Heung-Min 9 2 Mohamed Salah 8 2 Jamie Vardy 8 1 Bruno Fernandes 7 3 Harry Kane 7 2 Patrick Bamford 7 1 Callum Wilson 7 0 Ollie Watkins 6 1
2	7 Bruno Fernandes 7 3 Dominic Calvert-Lewin 7 12 Son Heung-Min 7 2 Harry Kane 5 2 Patrick Bamford 5 1 Callum Wilson 5 0 Mohamed Salah 3 2	DominicCalvert-Lewin 7 12 BrunoFernandes 7 3 SonHeung-Min 7 2 HarryKane 5 2 PatrickBamford 5 1 CallumWilson 5 0 MohamedSalah 3 2

#### Jawaban:

```
1 package main  
2 import "fmt"
```

```

3
4 type pemain struct{
5     name string
6     gol,assist int
7 }
8 const NMAX = 1000
9 type arrPemain [NMAX]pemain
10
11 func main(){
12     var T arrPemain
13     var i, n int
14     var fname, lname string
15     // INPUT
16     fmt.Scan(&n)
17     for i = 0; i < n && i < NMAX; i++ {
18         fmt.Scan(&fname,&lname, &T[i].gol, &T[i].assist)
19         T[i].name = fname+" "+lname
20     }
21     // PENGURUTAN DG SELECTION SORT
22     var pass,idx_max int
23     var temp pemain
24     for pass = 1; pass <= n-1; pass++ {
25         idx_max = pass-1
26         for i = pass; i < n; i++ {
27             if T[idx_max].gol < T[i].gol || (T[idx_max].gol == T[i].gol
28             && T[idx_max].assist <= T[i].assist) {
29                 idx_max = i
30             }
31         }
32         temp = T[idx_max]
33         T[idx_max] = T[pass-1]
34         T[pass-1] = temp
35     }
36     // MENAMPILKAN
37     for i = 0; i < n ; i++{
38         fmt.Println(T[i].name, T[i].gol, T[i].assist)
39     }
40 }
```

- 3) Sebuah program digunakan untuk menghitung perolehan suara dari berbagai partai politik dalam sebuah pemilihan umum calon legislatif. Program akan menampilkan data partai terurut berdasarkan perolehan suara terurut. Nama partai hanya disimbolkan dari angka 1 hingga N, di mana  $1 \leq N \leq 1000000$ .

**Masukan** berupa beberapa nilai yang dipisahkan oleh spasi. Masing-masing nilai menyatakan nama partai (1 hingga N) yang dipilih. Proses input ini diakhiri dengan nilai -1.

**Keluaran** berupa daftar partai dan peroleh suaranya yang terurut descending atau mengecil dengan format <partai>(<suara>). Perhatikan contoh masukan dan keluaran yang diberikan.

**Petunjuk** : gunakan tipe bentukan partai yang berisi nama dan suara. Data perolehan suara disimpan pada array of partai (kapasitas 1000000). Array tersebutlah yang diurutkan.

**Contoh masukan dan keluaran:**

1	<b>Masukan</b>	5 1 1 1 1 1 1 1 3 3 3 3 3 2 2 5 5 5 5 5 4 3 2 2 2 2 -1
	<b>Keluaran</b>	1(7) 5(6) 3(6) 2(6) 4(1)
2	<b>Masukan</b>	5 8 8 5 6 8 8 7 6 5 8 7 5 6 7 5 8 6 7 8 8 7 7 8 6 7 7 6 8 6 8 8 5 5 6 6 6 7 7 6 7 8 8 8 5 7 6 6 8 6 5 5 8 7 5 5 6 8 7 6 5 5 8 6 6 7 8 8 8 6 7 6 6 5 7 8 7 6 6 8 7 7 8 6 5 5 7 7 6 5 7 8 8 6 8 8 6 7 8 -1
	<b>Keluaran</b>	8(30) 6(28) 7(24) 5(18)
3	<b>Masukan</b>	8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 -1
	<b>Keluaran</b>	8(15)
4	<b>Masukan</b>	10 1 7 8 10 1 4 8 8 5 -1
	<b>Keluaran</b>	8(3) 10(2) 1(2) 7(1) 4(1) 5(1)
5	<b>Masukan</b>	14 10 13 13 14 10 11 13 13 12 15 11 10 -1
	<b>Keluaran</b>	13(4) 10(3) 14(2) 11(2) 12(1) 15(1)
6	<b>Masukan</b>	-1
	<b>Keluaran</b>	

**Jawaban:**

```

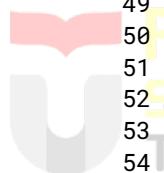
1 package main
2 import "fmt"
3 const NMAX = 1000000
4 // tipe bentukan partai
5 type partai struct {
6     nama,suara int
7 }
8 // tipe tabPartai: array of partai dengan kapasitas NMAX
9 type tabPartai [NMAX]partai
10
11 func main(){
12     // deklarasi variabel
13     var p tabPartai
14     var n,x,idx int
15     // lakukan proses input suara secara berulang di sini, simpan ke
16     dalam array p, sehingga terdapat array p yang berisi hasil peroleh suara
17     n partai.
18     n = 0
19     fmt.Scan(&x)
20     for x != -1 {
21         idx = posisi(p, n, x)
22         if idx == -1 {

```

```

23         p[n].nama = x
24         p[n].suara = 1
25         n++
26     }else{
27         p[idx].suara++
28     }
29     fmt.Scan(&x)
30 }
31 // lakukan proses pengurutan dengan insertion sort descending
32 berdasarkan jumlah suara yang diperoleh
33 var pass, k int
34 var temp partai
35 for pass = 1; pass <= n-1; pass++ {
36     k = pass
37     temp = p[k]
38     for k > 0 && temp.suara > p[k-1].suara {
39         p[k] = p[k-1]
40         k--
41     }
42     p[k] = temp
43 }
44 // tampilkan array p
45 for k = 0; k < n; k++ {
46     fmt.Printf("%v(%v) ",p[k].nama,p[k].suara)
47 }
48 }

49 func posisi(t tabPartai, n int, nama int) int {
50 /* mengembalikan indeks partai yang memiliki nama yang dicari pada array
51 t yang berisi n partai atau -1 apabila tidak ditemukan , gunakan
52 pencarian sekuisial */
53     var i, ketemu int
54     i = 0
55     ketemu = -1
56     for i < n && ketemu == -1 {
57         if t[i].nama == nama {
58             ketemu = i
59         }
60         i++
61     }
62 }
63 return ketemu
64 }
```



## BAB 5. PENGAYAAN

Bab lima ini berisi dua modul materi dan soal pengayaan, praktikan bisa melakukan eksplorasi secara mandiri di luar jam praktikum. Praktikan bisa bertanya kepada dosen terkait materi dan soal pada modul ini apabila terdapat hal-hal yang kurang dimengerti.

Adapun modul yang akan di bahas adalah

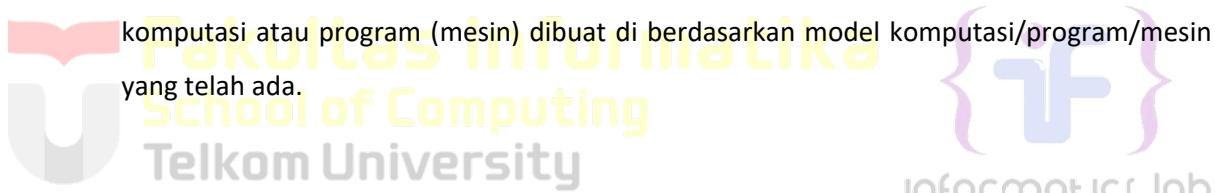
a) **Modul 17. Skema Pemrosesan Sekuensial**

Pada modul ini diperkenalkan skema pemrosesan sekuensial yang bisa digunakan untuk memperkuat pemahaman struktur kontrol perulangan. Penggunaan struktur kontrol memungkinkan praktikan untuk bisa menyelesaikan kasus-kasus perulangan berdasarkan pola-pola yang diperkenalkan pada skema pemrosesan sekuensial.

b) **Modul 18. Mesin Abstract**

Modul 18 ini memperkenal penggunaan konsep mesin abstract, di mana suatu model

komputasi atau program (mesin) dibuat di berdasarkan model komputasi/program/mesin yang telah ada.



## MODUL 17. SKEMA PEMROSESAN SEKUENSIAL

### 17.1. Pengantar Skema Pemrosesan Sekuensial

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).

### 17.2. Pembacaan Data Tanpa Marker pada Akhir Rangkaian Data

Pola ini memperlihatkan bahwa semua data yang diberikan pada masukan adalah data yang harus diproses. Berikut contoh polanya

	Notasi Algoritma	Notasi dalam bahasa Go
1	input(n)	fmt.Scanln(&n)
2	i ← 1	i = 0
3	while i <= n do	for i < n {
4	input(dat)	fmt.Scan(&dat)
5	{ kode untuk memproses dat }	// kode untuk memproses dat
6	i ← i + 1	i = i + 1
7	endwhile	}

Pada pola di atas, terlihat bahwa variabel **dat** yang diperoleh pada baris ke-4 adalah data valid yang pasti diproses pada baris ke-5. Baris ke-5 bisa berisi potongan kode apapun yang digunakan untuk memproses variabel **dat** tersebut.

Berikut contohnya pada potongan program pencarian nilai maksimum:

	Notasi Algoritma	Notasi dalam bahasa Go
1	input(n)	fmt.Scan(&n)
2	max ← {BILANGAN_KECIL}	max = // BILANGAN_KECIL
3	i ← 1	i = 1
4	while i <= n do	for i <= n {
5	input(dat)	fmt.Scan(&dat)
6	if dat > max then	if dat > max {
7	max ← dat	max = dat
8	endif	}
9	i ← i + 1	i = i + 1
10	endwhile	}
11	output("Data terbesar", max)	fmt.Println("Data terbesar", max)

### 17.3. Pembacaan Data Dengan Marker pada Akhir Rangkaian Data

Pada pola dengan marker, terdapat data yang dipersiapkan khusus untuk menghentikan perulangan.

Artinya semua data yang diberikan pada masukan adalah data yang valid, kecuali data yang terakhir, karena digunakan untuk menghentikan perulangan. Berikut contoh polanya:

	<b>Notasi Algoritma</b>	<b>Notasi dalam bahasa Go</b>
1	input(dat) 2   while <b>dat != MARKER</b> do 3     {kode untuk memproses dat} 4     input(dat) 5 endwhile	fmt.Scanln(&dat) for <b>dat != MARKER</b> { // kode untuk memproses dat fmt.Scanln(&dat) }

Apabila kita memperhatikan pola dengan marker di atas, terlihat bahwa selalu ada pengkondisian pada while yang akan selalu menyecek nilai **dat**. Semua nilai pada variabel **dat**, baik yang diperoleh pada baris ke-1 ataupun baris ke-4 akan selalu dicek pada baris ke-2. Apabila variabel **dat** tidak berisi **marker**, maka **dat** dapat diproses pada baris ke-3, sebaliknya apabila **dat** adalah **marker**, maka perulangan akan berhenti, dan **dat** tidak akan diproses pada baris ke-3.

Perhatikan contoh potongan program mencari nilai maksimum berikut ini:

	<b>Notasi Algoritma</b>	<b>Notasi dalam bahasa Go</b>
1	max ← {BILANGAN_KECIL} 2 input(dat) 3 while <b>dat != MARKER</b> do 4   if <b>dat &gt; max</b> then 5     max ← <b>dat</b> 6   endif 7   input(dat) 8 endwhile 9 output("Data terbesar", max)	max = // BILANGAN_KECIL fmt.Scan(&dat) for <b>dat != MARKER</b> { if <b>dat &gt; max</b> { max = <b>dat</b> } fmt.Scan(&dat) } fmt.Println("Data terbesar", max)

Nilai **marker** bisa nilai berapapun, biasanya diberikan pada soal atau kita biasanya bisa memberikan nilai berdasarkan asumsi.

#### 17.4. Kemungkinan Rangkaian Data Kosong, Hanya Ada Marker

Pola dengan marker pada contoh di atas memungkinkan terjadi bahwa data pertama yang diberikan pada masukan adalah marker, artinya tidak ada satu datapun yang valid. Kemungkinan ini disebut juga rangkaian data kosong atau kasus kosong.

	<b>Notasi Algoritma</b>	<b>Notasi dalam bahasa Go</b>
1	input(dat) 2 if <b>dat == MARKER</b> then 3   {kode untuk data kosong} 4 else 5    while <b>dat != MARKER</b> do 6       {kode untuk memproses dat} 7       input(dat)	fmt.Scanln(&dat) if <b>dat == MARKER</b> { // kode untuk data kosong } else { for <b>dat != MARKER</b> { // kode untuk memproses dat fmt.Scanln(&dat)

8	endwhile	}
9	endif	

Pada pola di atas ditambahkan struktur kontrol pengkondisian atau percabangan apabila ada kasus kosong yang terjadi. Berikut ini adalah contoh kode untuk mencari nilai maksimum

	Notasi Algoritma	Notasi dalam bahasa Go
1	input(dat)	fmt.Scanln(&dat)
2	if dat == MARKER then	if dat == MARKER {
3	output("tidak ada data")	fmt.Println("tidak ada data")
4	else	}else{
5	max < {BILANGAN KECIL }	max = // BILANGAN KECIL
6	while dat != MARKER do	for dat != MARKER {
7	if dat > max then	if dat > max {
8	max < dat	max = dat
9	endif	}
10	input(dat)	fmt.Scanln(&dat)
11	endwhile	}
12	output("Data terbesar", max)	fmt.Println("Data terbesar", max)
13	endif	}

### 17.5. Elemen Pertama Perlu Diproses Tersendiri/Kasus Khusus

Pada pola ini data pertama diproses terlebih dahulu secara khusus sebelum perulangan dilakukan. Apabila melihat contoh pencarian nilai maksimum di atas, terlihat bahwa nilai variabel **max** selalu diinisialisasi oleh sebuah nilai **BILANGAN KECIL** berapapun. Kekurangan dari pendekatan ini adalah kita harus mengetahui secara pasti nilai-nilai yang mungkin ada pada variabel **dat**, yang mana nilai pada variabel **dat** tersebut **TIDAK BOLEH** lebih kecil dibandingkan nilai dari **BILANGAN KECIL** yang digunakan saat inisialisasi. Sebagai contoh mencari nilai temperatur atau suhu maksimum. Pada kasus ini, berapa nilai dari **BILANGAN KECIL** yang akan digunakan? Tentu kita akan kesulitan menentukannya, karena temperatur bisa bernilai negatif. Oleh karena itu, dengan menggunakan pola Kasus Khusus ini, penentuan **BILANGAN KECIL** tersebut data dapat terselesaikan.

	Notasi Algoritma	Notasi dalam bahasa Go
1	input(dat)	fmt.Scanln(&dat)
2	if dat == MARKER then	if dat == MARKER {
3	{kode untuk data kosong}	// kode untuk data kosong
4	else	}else{
5	{proses data pertama}	{proses data pertama}
6	input(dat)	fmt.Scan(&dat)
7	while dat != MARKER do	for dat != MARKER {
8	{kode untuk memproses dat}	// kode untuk memproses dat
9	input(dat)	fmt.Scanln(&dat)
10	endwhile	}
11	endif	

		}
--	--	---

Pada pola tersebut, terlihat data atau elemen pertama yang diperoleh pada baris ke-1 diproses pada baris ke-5, selanjutnya data ke-2 didapat pada baris ke-6.

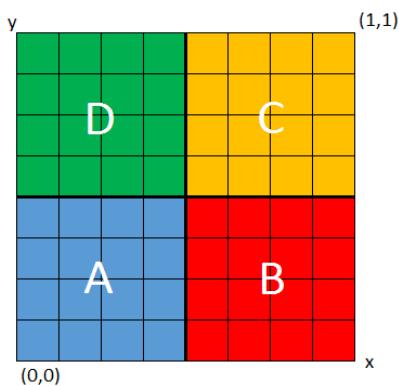
Berikut contoh potongan program untuk mencari nilai maksimum:

	<b>Notasi Algoritma</b>	<b>Notasi dalam bahasa Go</b>
1	input(dat)	fmt.Scanln(&dat)
2	if dat == MARKER then	if dat == MARKER {
3	output("tidak ada data")	fmt.Println("tidak ada data")
4	else	}else{
5	max <- dat	max = dat
6	input(dat)	fmt.Scan(&dat)
7	while dat != MARKER do	for dat != MARKER {
8	if dat > max then	if dat > max {
9	max <- dat	max = dat
10	endif	}
11	input(dat)	fmt.Scanln(&dat)
12	endwhile	}
13	output("Data terbesar", max)	fmt.Println("Data terbesar", max)
14	endif	}

## Fakultas Informatika Soal Latihan Modul 17

### 17.6. Soal Latihan Modul 17

- 1) Diberikan sejumlah bilangan real yang diakhiri dengan marker 9999, cari rerata dari bilangan-bilangan tersebut.
- 2) Diberikan string x dan n buah string, di mana 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 ke berapa 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 2. Ilustrasi denah daerah yang digunakan untuk mengukur curah hujan

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].

Berikut contoh masukan dan keluarannya:

No	Masukan	Keluaran
1	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  $\pi$  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):

N suku pertama: 1000000

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, terpikirkan oleh Monti cara menghitung berapa **banyak topping yang dia butuhkan**, dan cara menghitung **nilai  $\pi$** .

Ilustrasi seperti gambar yang diberikan di bawah, topping adalah lingkaran-lingkaran kecil. Ada yang tepat berada di atas pizza, dan ada yang jatuh di dalam kotak tetapi berada di luar pizza.

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 di dalam lingkaran apabila memenuhi ketidaksamaan:

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

Pada ilustrasi 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 di atas.

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 di atas 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 di atas, maka nilai konstanta  $\pi$  dapat dihitung.

Modifikasi program di atas sehingga dapat menghitung dan menampilkan nilai konstanta  $\pi$ .

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 18. MESIN ABSTRAK

### 18.1. Pengantar Mesin Abstrak

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

### 18.2. Contoh Kasus Mesin Domino

Tipe data dan operasi dasar mesin domino:

- 1) 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
- 2) 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, di mana 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 di mana salah satu gambar (suit)-nya sama dengan kartu pertama.

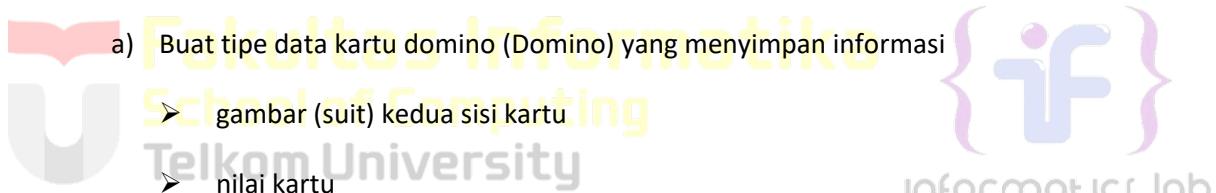
- Uji apakah kartu yang dipegang adalah kartu balak, yaitu kartu di mana kedua sisinya mempunya nilai yang sama.
- Diberikan sepasang kartu, apakah nilai kedua kartu tersebut 12?

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.

### 18.3. Soal Latihan Modul 18

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
  - prosedur **kocokKartu**(Dominoes)
  - fungsi **ambilKartu**(Dominoes) → Domino
  - fungsi **gambarKartu**(Domino,suit int) → int
  - fungsi **nilaiKartu**(Domino) → int
- 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

- b) fungsi **sepasangKartu**(Domino,Domino) → Boolean; yang memberikan nilai **true** jika total nilai kartu adalah 12 dan **false** jika tidak.
- 3) Implementasi salah satu permainan domino. Lihat lampiran untuk deskripsi permainan Gapleh.
- 4) Implementasi mesin abstrak karakter yang bekerja terhadap untaian karakter (yang diakhiri dengan penanda titik (".")) dan mempunyai sejumlah operasi dasar.
- a) Operasi dasar mesin karakter:
- Prosedur **start()**; yang menyiapkan mesin karakter di awal 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 di atas 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://go.dev/tour/welcome/1>
2. Dokumentasi GoLang: <https://go.dev/>
3. Daftar paket standar yang tersedia dalam bahasa Go: <https://pkg.go.dev/std>
4. Situs tutorial dari pihak ketiga:
  - a. <https://gobyexample.com>,
  - b. <https://golangbot.com>,
  - c. <https://golangtutorial.com>,
  - d. <https://www.tutorialspoint.com/go/index.htm>,
5. Situs latihan pemrograman:
  - a. <https://hackerrank.com>,
  - b. <https://spoj.com>



## 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:

- a. Fungsi **Modf(x)(b,p)** mengembalikan bagian bilangan bulat b dan bilangan pecahan p dari suatu bilangan riil x.
- b. Fungsi **Pow(x,y)** menghitung x pangkat y ( $x^y$ )
- c. 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:

- a. Prosedure **Seed(s)** untuk menginisialisasi awal nilai acak.
- b. Fungsi **Int()** menghasilkan nilai acak integer dalam rentang [-231 s.d. 231].
- c. Fungsi **Intn(n)** menghasilkan nilai integer acak dalam rentang [0 .. n), yaitu antara 0 s.d. n-1.
- d. 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.Print( 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:

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

f. Metoda **t.UnixNano()** untuk mengambil data nanosekon dari data **Time t**.

g. 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.Print( 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)
g. Fungsi Open membuka file yang sudah ada.

func Open(n string) (*File, error)
h. Fungsi Close menutup file yang sudah dibuka oleh Create/Open.

func (f *File)Close() error
i. Metoda Read membaca data dalam bytes dengan mengembalikan jumlah data
yang berhasil dibaca.

func (f *File)Read(b []byte) (int, error)ALGORITMA PEMROGRAMAN
86

j. 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.

- a. 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)
```

- b. 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)
```

- c. 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)
```

- d. Menulis teks ke **os.Stdout**, output standar yang biasanya layar monitor atau dapat dialihoutput 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)
```

- e. 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)
```

- f. Menkonversi data ke bentuk string dengan proses **Print**.

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

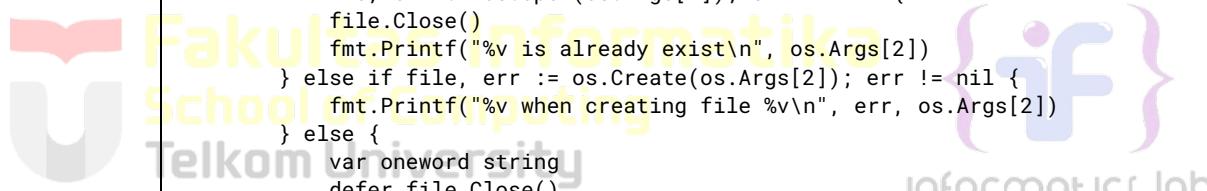
Berikut ini beberapa kode pemformat yang dapat digunakan dalam string format:

- **%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  $x.y\pm z$ ).
- **%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:

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

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

- b. **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{})
```

- c. **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:

- a. 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)
```

- b. Untuk membaca file. Bandingkan dengan perintah **os.Read** sebelumnya

```
func ReadAtLeast(r File, buf []byte, min int) (n int, err error)
```

- c. Untuk menuliskan string **s** kedalam file.

```
func WriteString(w File, s string) (n int, err error)
```

Contoh

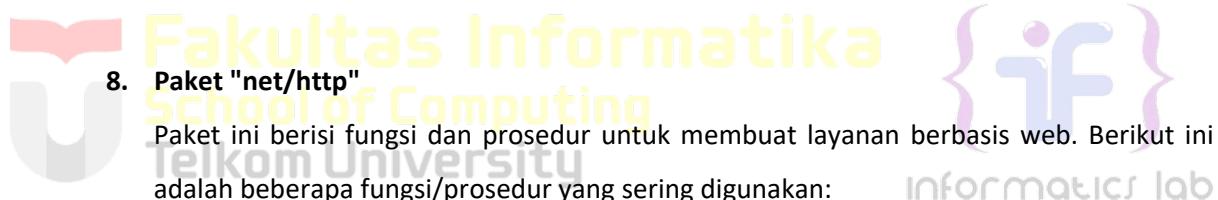
```
package main  
/* Ini hanya contoh  
Program yang baik seharusnya memeriksa berbagai kemungkinan error:  
Argumen yang salah,
```

```

Ada/tidaknya file masukan,
Output akan menimpah 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 jimmmyt jimmmyt 336 Jul 2 15:35 sumber.dat
-rw-rw-r--. 1 jimmmyt jimmmyt 336 Jul 2 17:13 tujuan.dat
$ ./contohio sumber.dat duplikat.dat
$ ls -l
-rw-rw-r--. 1 jimmmyt jimmmyt 336 Jul 2 15:35 sumber.dat
-rw-rw-r--. 1 jimmmyt jimmmyt 336 Jul 2 17:13 tujuan.dat
-rw-rw-r--. 1 jimmmyt jimmmyt 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
```

- e. 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 mengkonversi data string dari/ke tipe yang lain, yang tidak dapat dilakukan dengan cara casting.

- a. 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)
```

- b. 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)
```

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

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

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

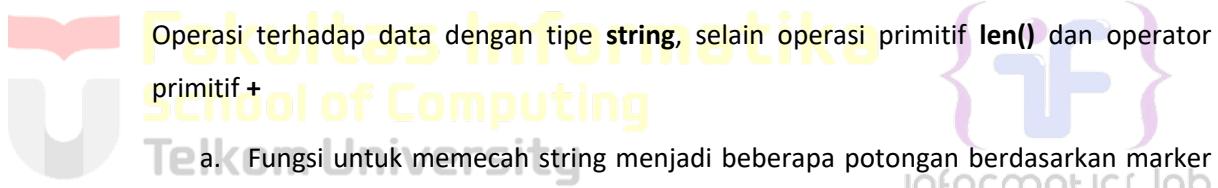
```
func Itoa(i int) string
```

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/indeks 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
```

- e. Menghapus spasi dibagian depan dan belakang suatu string.

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

Contoh

```
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.



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

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

- b. 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
```

- c. 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.

Contoh

```
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**.

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

```
func TestXXX(t *testing.T)
```

- b. Uji tolok-ukur dilakukan dalam fungsi bernama yang diawali dengan **Benchmark** dan mempunyai parameter **\*testing.B**

```
func BenchmarkXxx(b *testing.B)
```

- c. Uji keluaran dilakukan dalam fungsi bernama yang diawali dengan Example dan tanpa parameter.

```
func ExampleXXX()
```

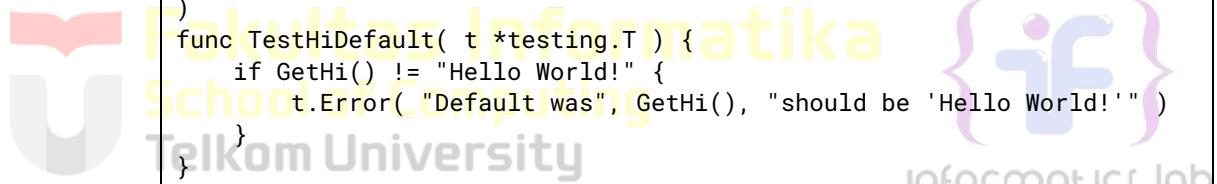
Pengujian dapat melaporkan melalui metoda berikut, yang ada dalam antarmuka T dan B

- 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.

Contoh

```
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**

1. **go version**: Menampilkan versi dan platform implementasi perkakas Go
2. **go help**: Menampilkan ringkasan informasi penggunaan perkakas Go
3. **go env GOPATH**: Menampilkan lokasi folder untuk menyimpan program Go
  - a. Program sumber di **\$GOPATH/src**
  - b. Program biner/eksekutabel di **\$GOPATH/bin**
  - c. Pustaka paket program di **\$GOPATH/pkg**
4. **go env GOROOT**: Menampilkan lokasi instalasi perkakas Go
5. **go build**: Melakukan proses kompilasi dalam folder *current*. Jika sukses, akan menghasilkan program eksekutabel. (Yaitu file dengan ekstensi .exe untuk os Windows)
6. **go clean**: Membersihkan sisa-sisa hasil kompilasi sebelumnya
7. **go run filename.go**: Mengkompilasi dan langsung mengeksekusi program tersebut. Program eksekutabel tidak disimpan, sehingga memberikan ilusi seolah Go adalah sebuah interpreter,
8. **go fmt**: Me-reformat program sumber yang ada di folder *current*, sehingga memenuhi standar format penulisan program sumber Go
9. **go test**: Melakukan uji (tahap unit), baik uji logik maupun tolok-ukur (*benchmark*).  
Program  
sumber uji harus mempunyai ekstensi **\_test.go**.
10. **go install**: Untuk menginstall program hasil kompilasi pada lokasi yang dituju, biasanya di  
**\$GOPATH/bin/**
11. **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/**.
  - a. **go get golang.org/dl/go1.12.5**: Untuk mengunduh Go versi 1.12.5

- b. **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 diinstall 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()
}

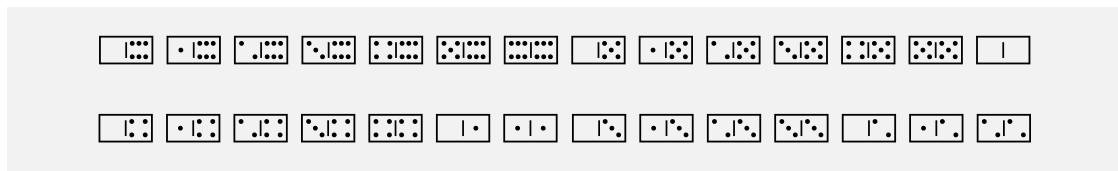
func main() {
    err := ui.Main(gui)
    if err != nil {
        panic(err)
    }
}

```



#### E. Contoh Program

(Permainan Solitaire Gapleh) Kartu domino yang biasa digunakan bermain gapleh berisi 28 kartu seperti yang ditampilkan dalam gambar di bawah 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.



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) di mana -1 .. -7 untuk meletakkan kartu diujung kiri rantai, 1..7 untuk ujung kanan rantai, 0 untuk selesai ronde, 9 untuk selesai permainan

**Contoh masukan dan keluaran (teks bergaris bawah adalah input/masukan):**

```
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? +1                                     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)                                    The train (0,4) (4,3) (3,6) (6,5) (5,5)
Decision? +4                                     4th tile=(2,5)
Train is (0,2)                                    The train (0,4) (4,3) (3,6) (6,5) (5,5)
(5,2)
Decision? +6                                     6th tile=(2,3)
Train is (0,3)                                    The train
(0,4)(4,3)(3,6)(6,5)(5,5)(5,2)(2,3)
Decision? 0                                       No more legal move
You earn 6 points
Your score is 6/1
Dealing ...                                         Shuffle the tiles for a new game
Your tiles: (3,5) (1,4) (0,1) (3,6) (4,4) (0,4) (0,6)
Starting train is (3,4)
Decision? +5                                     5th tile=(4,4) put on right end
Train is (3,4)                                    The train (3,4) (4,4)
Decision? -4                                     4th tile=(3,6) put on left end
Train is (6,4)                                    The train (6,3) (3,4) (4,4)
Decision? -7                                     7th tile=(0,6)
```

```
Train is (0,4)          The train (0,6) (6,3) (3,4) (4,4)
Decision? -6          6th tile=(0,4)
Train is (4,4)          The train (4,0) (0,6) (6,3) (3,4) (4,4)
Decision? -2          2nd tile=(1,4)
Train is (1,4)          The train (1,4) (4,0) (0,6) (6,3) (3,4)
(4,4)
Decision? -3          3rd tile=(0,1)
Train is (0,4)          The train
(0,1)(1,4)(4,0)(0,6)(6,3)(3,4)(4,4)
Decision? 0            no more legal move
You earn 6 points
Your score is 12/2
Dealing ...             Shuffle the tiles for a new game
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)
```





**Kontak Kami :**

-  @dky2921g
-  @informaticslab
-  @informaticslab\_telu
-  informaticslab@telkomuniversity.ac.id
-  informatics.labs.telkomuniversity.ac.id