

Modular Programming

Menggunakan Subprogram (Fungsi/Prosedur)

Tim Pengajar IF1210

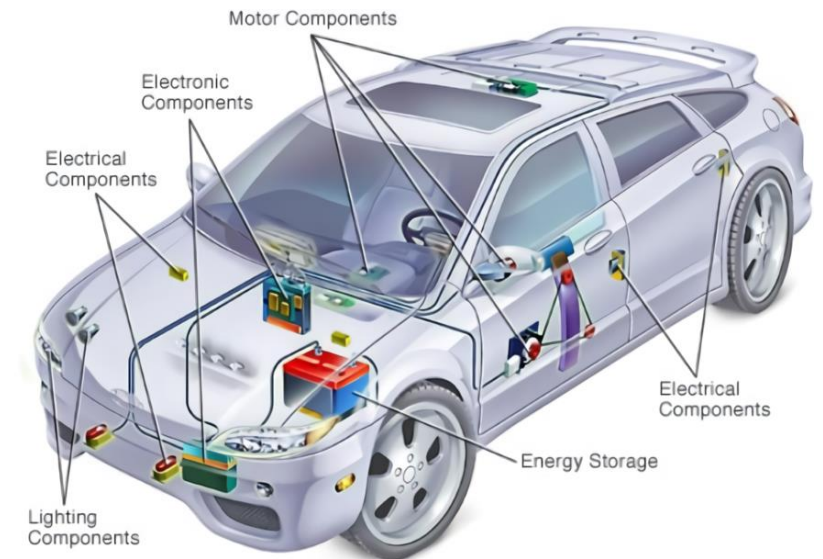
Sekolah Teknik Elektro dan Informatika

Sumber Utama

Diktat “Dasar Pemrograman, Bag.
Pemrograman Prosedural” oleh Inggriani
Liem

Pendahuluan

- Sesuatu yang “besar” biasanya terdiri atas komponen-komponen “kecil”
- Contoh:
 - Sebuah **mobil listrik** terdiri atas berbagai komponen yang masing-masing dapat difabrikasi secara terpisah dan selanjutnya dirakit menjadi sebuah mobil
 - Sebuah **peralatan elektronik canggih** di dalamnya terdiri atas komponen elektronik standar yang dirakit
 - Sebuah **gedung besar** memiliki komponen seperti pintu, jendela, ubin yang dapat difabrikasi secara terpisah sebelum dipasang.
- Dengan demikian, ada **produsen** komponen dan ada **perakit** pemakai komponen



Komponen mobil listrik

Pendahuluan

- Pendekatan “merakit” ini juga dilakukan dalam pembangunan perangkat lunak
- Dengan demikian, ada dua kategori programmer: (1) penyedia modul/komponen dan (2) pemakai modul komponen
- Skala “komponen” juga berkembang, dari komponen setara “suku cadang” dalam industri perakitan mobil, sampai setara “engine” dari produk keseluruhan.
 - Dalam *software engineering*, muncul istilah *library*, *framework*, *platform*, dsb.

Modular Programming

- **Modular programming** *is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect or "concern" of the desired functionality.*
- Salah satu implementasi **dekomposisi** dalam pemrograman prosedural adalah melalui *modular programming*

Implementasi Program Modular

Subprogram

- Fungsi
- Prosedur

Abstract Data Type

Modul/paket/ unit/library

Kelas

- Pemrograman berorientasi objek

Tidak
dibahas
lebih lanjut
di kuliah ini

Keuntungan Modular Programming

- Memudahkan programmer bekerja secara *team work*
- Modifikasi program lebih mudah dilakukan karena perubahan dapat diisolasi pada modul tertentu
- Modul dapat dites dan di-*debug* secara independen
- Modul-modul program dapat dikemas dalam bentuk *library* yang dapat digunakan oleh program lain

Subprogram

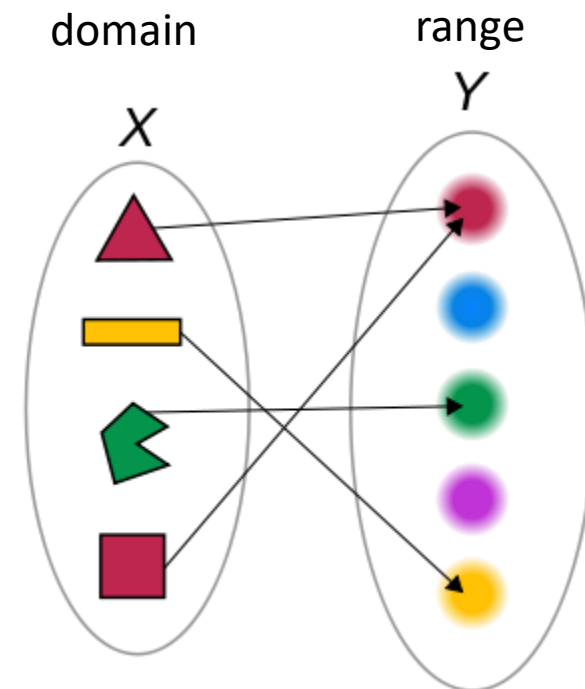
- *A set of instructions designed to perform a frequently used operation within a program*
- 2 (dua) jenis subprogram:
 - **Fungsi:** pemetaan suatu nilai domain (input) ke range (output)
 - Hasil dari fungsi dinyatakan dalam sebuah type data yang eksplisit
 - **Prosedur:** deretan instruksi yang jelas initial state dan final state-nya → mirip seperti program secara umum, namun dalam scope yang lebih kecil
- Tahap penggunaan fungsi/prosedur dalam program
 - **Mendefinisikan** fungsi/prosedur: nama, spesifikasi, parameter formal
 - **Merealisasikan** fungsi/prosedur
 - **Memanggil/memakai** fungsi/prosedur

Kegunaan Subprogram

- Program dapat didekomposisi menjadi sub-sub bagian
 - Tiap sub bagian dapat didefinisikan sebagai fungsi/prosedur yang tinggal dipanggil sebagai 1 baris atau ekspresi dalam program utama
- *Code reuse instead of code rewriting*
 - Jika task yang harus dikerjakan fungsi/prosedur banyak dipakai di program, memrogram menjadi jauh lebih sederhana jika task tersebut dibuat dalam bentuk fungsi/prosedur
 - Contoh: fungsi untuk menghitung akar kuadrat (**sqrt**) sangat berguna untuk berbagai jenis persoalan → bayangkan kalau setiap kali Anda harus menulis programnya 😊
- Setiap fungsi/prosedur dapat dites secara mandiri dan tidak tergantung pada bagian program yang lain
 - Jika program besar dan harus dikerjakan oleh lebih dari 1 programmer, hal ini memudahkan pembagian kerja

Fungsi (1)

- Konsep fungsi di pemrograman didasari oleh konsep pemetaan dan **fungsi** di matematika
 - Setiap elemen pada himpunan domain dipetakan **tepat satu** ke sebuah elemen pada himpunan range
- **Fungsi**: pemetaan suatu **domain** ke **range** berdomain tertentu
 - Secara algoritmik, sebuah fungsi akan menerima suatu harga yang diberikan lewat parameter formal bertipe tertentu (jika ada) dan menghasilkan suatu nilai sesuai domain yang didefinisikan dalam spesifikasi fungsi
- Contoh: $f(X) = X^2$
 - fungsi untuk menghitung kuadrat dari suatu bilangan
 - Domain: bilangan bulat
 - Range: bilangan bulat (0 atau positif)



$f : X \rightarrow Y$

Fungsi dalam Notasi Algoritmik

function *nama_fungsi* (*param1: type1*, *param2: type2*, ..., *paramn: typen*) → *type_hasil*
 { Spesifikasi fungsi }

KAMUS LOKAL

{ Konstanta, variabel, fungsi/prosedur yang dideklarasikan lokal di function }

ALGORITMA

{ Langkah-langkah algoritmik untuk fungsi }
 { Return hasil menggunakan notasi : → *hasil* }
 → *hasil*

Contoh

function Kuadrat (*x* : integer) → integer
 { Menghasilkan pangkat2 dari *x* }

KAMUS LOKAL

ALGORITMA

→ *x* * *x*

Parameter **formal**: parameter yang digunakan dalam definisi fungsi/prosedur

Fungsi (2)

- Nama parameter yang dituliskan pada definisi/spesifikasi fungsi disebut sebagai parameter **formal**
- Daftar parameter input pada fungsi *param1: **type1***, *param2: **type2***, ..., *paramn: **typen*** boleh kosong
- Instruksi “terakhir” yang harus ada pada fungsi harus merupakan pengiriman harga yang dihasilkan oleh fungsi (notasi algoritmik: → hasil)
 - Instruksi “terakhir” belum tentu dituliskan di baris terakhir, misalnya jika hasil merupakan nilai yang dihasilkan berdasarkan analisis kasus
- Type data pada list parameter input dan type hasil boleh merupakan type dasar maupun type bentukan

Prosedur (1)

- **Prosedur:**
 - Sederetan instruksi algoritmik yang diberi nama dan akan menghasilkan efek neto yang terdefinisi
 - Harus jelas *initial state* (I.S.) dan *final state* (F.S.) prosedur → I.S. dan F.S. dari prosedur menjamin eksekusi program mencapai efek neto yang diharapkan
- **Jenis prosedur:**
 - **Prosedur tanpa parameter:** memanfaatkan nilai dari nama-nama yang terdefinisi pada kamus global
 - **Prosedur berparameter:** dirancang agar sepotong kode yang sama Ketika eksekusi dilakukan dapat dipakai dengan nama-nama parameter [aktual] yang berbeda

Prosedur dalam Notasi Algoritmik

procedure *nama_proc* ([input/output] *param1:type1*, ..., [input/output] *paramn:typen*)
 { Spesifikasi prosedur }

KAMUS LOKAL

{ Konstanta, variabel, fungsi/prosedur yang dideklarasikan local di prosedur }

ALGORITMA

{ Langkah-langkah algoritmik untuk prosedur }

Contoh

procedure CetakNama (input *nama* : *string*)
 { I.S.: *nama* terdefinisi; F.S.: tercetak "Hello +
nama" di layar }

KAMUS LOKAL

ALGORITMA

output ("Hello ", *nama*)

Parameter **formal**:
 parameter yang
 digunakan dalam definisi
 fungsi/prosedur

Prosedur (2)

- Parameter **formal**: nama-nama parameter yang dipakai dalam mendefinisikan prosedur dan membuat prosedur tersebut dapat dieksekusi dengan nama-nama yang berbeda saat dipanggil
- Jenis parameter formal pada prosedur:
 - **input**: parameter yang diperlukan prosedur sebagai masukan untuk melakukan aksi efektif
 - **output**: parameter yang nilainya **akan** dihasilkan oleh prosedur
 - **input/output**: parameter yang nilainya diperlukan prosedur sebagai masukan untuk melakukan aksi, dan pada akhir prosedur akan dihasilkan nilai baru

Pemanggilan Fungsi / Prosedur (1)

Program *Judul*

{ Spesifikasi Program }

KAMUS

{ Variabel, konstanta, definisi fungsi/prosedur }

{ Definisi dan spesifikasi fungsi/prosedur }

function ...

{ spesifikasi function }

procedure ...

{ spesifikasi procedure }

ALGORITMA PROGRAM UTAMA

{ Algoritma program mengandung pemanggilan
fungsi/prosedur }

{ **REALISASI FUNGSI/PROSEDUR** }
{ lihat slide berikutnya }

Pemanggilan Fungsi/Prosedur (2)

```
{ REALISASI FUNGSI/PROSEDUR}  
function ...  
{ spesifikasi function }  
KAMUS LOKAL  
  
...  
ALGORITMA  
  
...  
procedure ...  
{ spesifikasi procedure }  
KAMUS LOKAL  
  
...  
ALGORITMA  
  
...
```

Program CetakKuadrat

```
{ Input: masukan integer dan
  mencetak nilai kuadratnya ke layar }
```

KAMUS

```
bil : integer
```

```
function Kuadrat (x : integer) → integer
```

```
{ menghasilkan kuadrat x }
```

```
procedure CetakInt (input x : integer)
```

```
{ I.S. x terdefinisi; F.S. x tercetak ke layar }
```

ALGORITMA PROGRAM UTAMA

```
input(bil) { baca dari keyboard }
```

```
CetakInt (Kuadrat(bil))
```

{ REALISASI FUNGSI/PROSEDUR }

```
function Kuadrat (x : integer) → integer
```

```
{ menghasilkan kuadrat x }
```

KAMUS LOKAL**ALGORITMA**

```
→ x * x
```

```
procedure CetakInt (input x : integer)
```

```
{ I.S. x terdefinisi; F.S. x tercetak ke layar }
```

KAMUS LOKAL**ALGORITMA**

```
output(x)
```

Contoh Pemanggilan Fungsi/Prosedur di Notasi Algoritmik

bil adalah contoh parameter aktual

Parameter **aktual**: parameter yang digunakan dalam pemanggilan fungsi/prosedur

Pemanggilan Fungsi/Prosedur (3)

- Memakai/memanggil prosedur/fungsi adalah menuliskan nama fungsi/prosedur yang pernah didefinisikan dan memberikan harga-harga yang dibutuhkan oleh fungsi/prosedur
 - Fungsi dipanggil sebagai bagian dari ekspresi
 - Prosedur dipanggil sebagai sebuah instruksi
- Parameter **aktual**: nama-nama informasi yang dipakai ketika fungsi/prosedur dipakai (“dipanggil”)
- Saat pemanggilan terjadi terjadi asosiasi antara parameter formal dengan parameter aktual sesuai urutan penulisan dalam daftar parameter input
 - Daftar pada parameter aktual harus sama/kompatibel jumlah (arity), urutan, dan type data dengan daftar parameter formal fungsi/prosedur → asosiasi “by position”

Pemanggilan Fungsi/Prosedur (4)

- Parameter aktual pada **fungsi**:
 - dapat berupa nama variabel/konstanta yang terdefinisi pada kamus, atau suatu harga tertentu, atau harga yang dihasilkan dari sebuah ekspresi atau fungsi
- Parameter aktual pada **prosedur**:
 - Parameter **input** harus sudah terdefinisi nilainya (karena dibutuhkan prosedur untuk melakukan aksinya)
 - Dapat berupa suatu nama variable/konstanta yang sudah terdefinisi nilainya, dapat berupa nilai/ekspresi tanpa menggunakan nama
 - Parameter **output** tidak perlu terdefinisi nilainya karena justru nilainya akan dihasilkan oleh prosedur. Setelah pemanggilan prosedur, nilai yang dihasilkan akan dimanfaatkan oleh deretan instruksi berikutnya.
 - harus ditampung dalam suatu variabel
 - Parameter **input/output** harus terdefinisi nilainya dan nilai baru yang diperoleh karena eksekusi prosedur akan dimanfaatkan oleh deretan instruksi berikutnya.

Studi Kasus

Studi Kasus

- Buatlah sebuah program yang digunakan untuk melakukan operasi matriks dengan elemen bertipe integer dengan ukuran 3 x 3.
- Program diawali dengan membaca masukan elemen 2 buah matriks 3 x 3, yaitu matriks A dan matriks B dari pengguna.
- Selanjutnya, diberikan pilihan menu (integer) yang bisa digunakan untuk memilih operasi matriks yang akan dilakukan, yaitu:
 - pilihan 1 untuk operasi penjumlahan kedua matriks,
 - pilihan 2 untuk operasi pengurangan matriks A dengan matriks B, dan
 - pilihan 3 adalah operasi untuk memeriksa apakah kedua matriks adalah matriks satuan atau tidak.Matriks satuan adalah matriks yang elemen-elemennya hanya terdiri atas angka 0 dan/atau 1.
- Jika dimasukkan pilihan selain 1, 2, 3, maka dituliskan pesan kesalahan “Bukan pilihan yang benar”. Tidak perlu ada validasi masukan pilihan dengan pengulangan.
- Jika pilihan 1, 2, atau 3, program akan menghasilkan output hasil operasi (tergantung pilihan menu).

Studi Kasus

Input			Output (Tampilan di Layar)
Matriks A	Matriks B	Pilihan menu	
1 2 3 4 2 3 4 2 3	1 2 3 2 3 4 2 4 2	1	2 4 6 6 5 7 6 6 5
1 2 3 4 2 3 4 2 3	1 2 3 2 3 4 2 4 2	2	0 0 0 2 -1 -1 2 -2 1
1 2 3 4 2 3 4 2 3	1 2 3 2 3 4 2 4 2	3	Matriks A bukan matriks satuan Matriks B bukan matriks satuan
1 2 3 4 2 3 4 2 3	1 0 0 0 1 0 0 0 0	3	Matriks A bukan matriks satuan Matriks B adalah matriks satuan
1 2 3 4 2 3 4 2 3	1 0 0 0 1 0 0 0 0	0	Bukan pilihan yang benar

Sekilas tentang Matriks

- A **matrix** is a rectangular array or table of numbers, symbols, or expressions, with elements or entries arranged in rows and columns, which is used to represent a mathematical object or property of such an object. ([https://en.wikipedia.org/wiki/Matrix_\(mathematics\)](https://en.wikipedia.org/wiki/Matrix_(mathematics)))
- Contoh:
$$\begin{bmatrix} 1 & 9 & -13 \\ 20 & 5 & -6 \end{bmatrix}$$
 - Matriks 2x3 yaitu matriks dengan 2 baris (*row*) dan 2 kolom (*column*)
- Implementasi dalam pemrograman: sebagai array 2 dimensi (array of array)

Studi Kasus: Analisis I.S. dan F.S.

I.S.: Matriks A, matriks B, dan pilihan menu terdefinisi berdasarkan masukan pengguna

F.S.: Hasil penjumlahan/pengurangan matriks A dengan matriks B atau pernyataan apakah matriks A dan matriks B adalah matriks satuan atau bukan, atau pesan kesalahan, ditampilkan di layar tergantung pada pilihan menu

Studi Kasus: Struktur Data

- Struktur data **matriks**:
 - menggunakan array of array of [type elemen]
- Deklarasi variabel matriks (contoh):

M: array [1..3] of array [1..4] of integer

- Matriks bernama **M** dengan setiap elemen bertipe **integer**, dengan **ukuran baris = 3** dan **ukuran kolom = 4**; dengan alamat setiap elemen diakses melalui **indeks baris 1 s.d. 3** dan **indeks kolom 1 s.d. 4**

Solusi 1

Banyaknya bagian program yang berulang

Semakin besar dan kompleks program, akan semakin tidak efisien

```
{ Baca pilihan menu dan lakukan operasi sesuai pilihan menu }
output("Masukkan pilihan menu (1,2,3) = ") { Format bebas }
input(pilihan)
depend on (pilihan)
  pilihan = 1 :
    { Menjumlahkan kedua matriks dan mencetak hasilnya ke layar }
    i traversal [1..3]
      j traversal [1..3]
        MHasil[i][j] ← MA[i][j] + MB[i][j]
    output("Hasil penjumlahan matriks A dan B =")
    i traversal [1..3]
      j traversal [1..3]
        output(MHasil[i][j], " ")
  pilihan = 2 :
    { Mengurangkan kedua matriks dan mencetak hasilnya ke layar }
    i traversal [1..3]
      j traversal [1..3]
        MHasil[i][j] ← MA[i][j] - MB[i][j]
    output("Hasil pengurangan matriks A dengan B =")
    i traversal [1..3]
      j traversal [1..3]
        output(MHasil[i][j], " ")
  pilihan = 3 :
    { Cek apakah kedua matriks adalah matriks satuan/bukan }
    { Mengecek apakah MA adalah matriks satuan/bukan }
    count ← 0
    i traversal [1..3]
      j traversal [1..3]
        if (MA[i][j] ≠ 0) and (MA[i][j] ≠ 1) then
          count ← count + 1
    if (count = 0) then { count = 0, berarti tidak ada elemen bukan 0/1 }
      output("Matriks A adalah matriks satuan")
    else
      output("Matriks A bukan matriks satuan")
    { Mengecek apakah MB adalah matriks satuan/bukan }
    count ← 0
    i traversal [1..3]
```

Solusi 2

Bagian-bagian yang kompleks dipisahkan dalam **fungsi/prosedur** sendiri
Sketsa algoritma (dalam notasi algoritmik):

InputMatriksA

InputMatriksB

input (pilihan)

depend on pilihan

pilihan = 1 :

JumlahMatriksAB

pilihan = 2 :

KurangMatriksAB

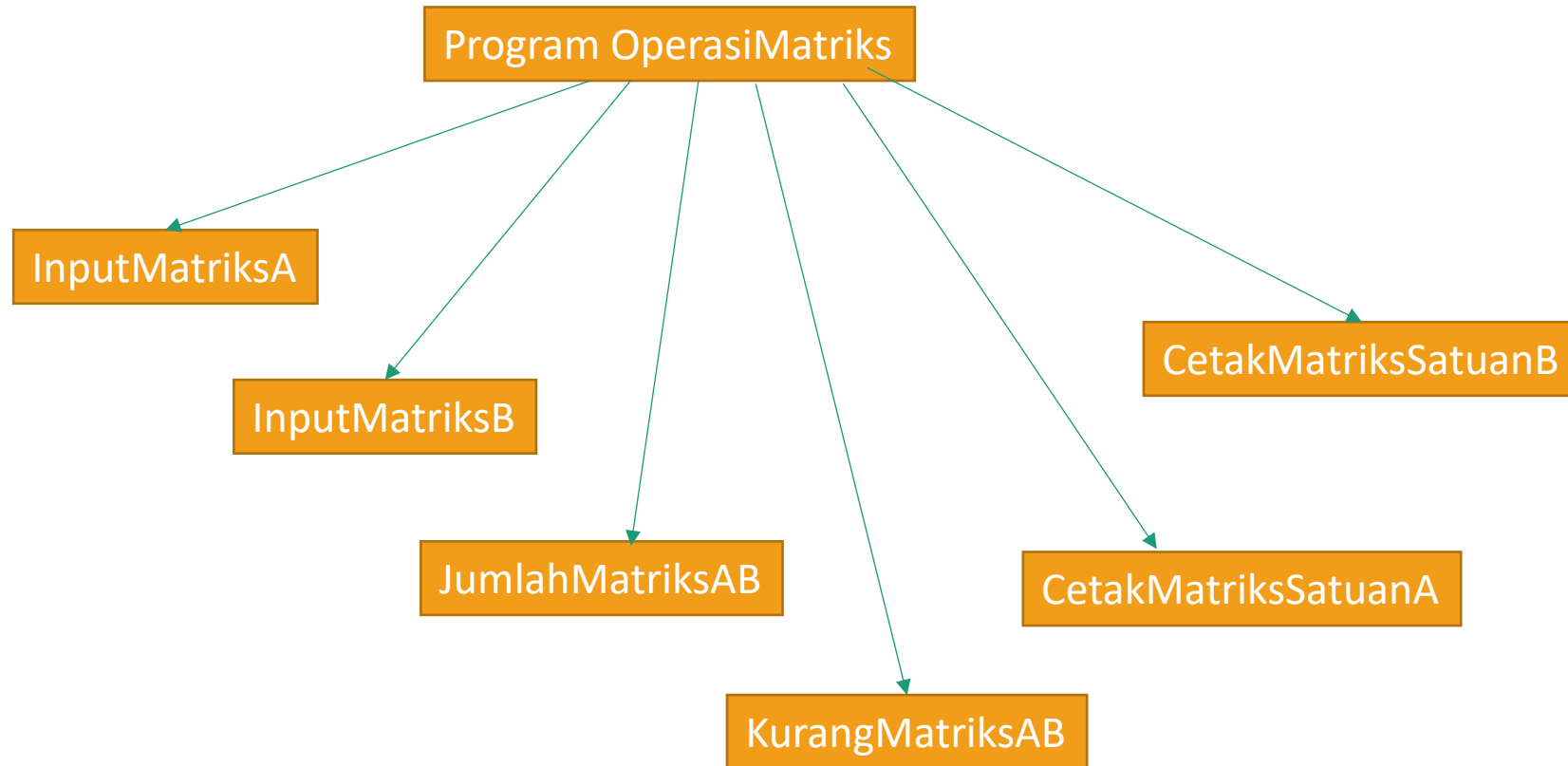
pilihan = 3 :

CetakMatriksSatuanA

CetakMatriksSatuanB

else : output (“Bukan pilihan yang benar”)

Dekomposisi Fungsional – Solusi 2



Solusi 2 – Input Matriks A

procedure InputMatriksA

{ I.S. MA sudah didefinisikan sebagai variabel global sebagai
array [1..3] of array [1..3] of integer }
{ F.S. MA berisi data berdasarkan masukan dari user }

KAMUS LOKAL

i, j : integer

ALGORITMA

i traversal [1..3]
 j traversal [1..3]
 input (MA[i][j])

Solusi 2 – Input Matriks B

procedure InputMatriksB

{ I.S. MB sudah didefinisikan sebagai variabel global sebagai
array [1..3] of array [1..3] of integer }
{ F.S. MB berisi data berdasarkan masukan dari user }

KAMUS LOKAL

i, j : integer

ALGORITMA

i traversal [1..3]
 j traversal [1..3]
 input (MB[i][j])

Solusi 2 – Jumlah Matriks A B

procedure JumlahMatriksAB

{ I.S. MA dan MB sudah didefinisikan sebagai variabel global
sebagai array [1..3] of array [1..3] of integer dan
sudah terisi }

{ F.S. MHasil (terdefinisi sebagai variabel global)
berisi hasil penjumlahan matriks A dan B }

KAMUS LOKAL

i, j : integer

ALGORITMA

i traversal [1..3]

j traversal [1..3]

MHasil[i][j] ← MA[i][j] + MB[i][j]

i traversal [1..3]

j traversal [1..3]

output(MHasil[i][j])

Solusi 2 – Kurangi Matriks A dengan B

procedure KurangMatriksAB

```
{ I.S. MA dan MB sudah didefinisikan sebagai variabel global
    sebagai array [1..3] of array [1..3] of integer dan
    sudah terisi }
{ F.S. MHasil (terdefinisi sebagai variabel global)
    berisi hasil pengurangan matriks A dengan B }
```

KAMUS LOKAL

i, j : integer

ALGORITMA

```
i traversal [1..3]
    j traversal [1..3]
        MHasil[i][j] ← MA[i][j] - MB[i][j]
i traversal [1..3]
    j traversal [1..3]
        output(MHasil[i][j])
```

Solusi 2 – Cetak Matriks Satuan A

procedure CetakMatriksSatuanA

{ I.S. MA sudah didefinisikan sebagai variabel global sebagai
array [1..3] of array [1..3] of integer dan sudah terisi }
{ F.S. Tercetak ke layar apakah MA matriks satuan atau bukan }

KAMUS LOKAL

i, j, count : integer

ALGORITMA

count \leftarrow 0

i traversal [1..3]

i traversal [1..3]

if (MA[i][j] \neq 0) and (MA[i][j] \neq 1) then
 count \leftarrow count + 1

if (count = 0) then { jika count = 0, berarti tidak ada elemen bukan 0/1 }
 output ("Matriks A adalah matriks satuan")

else

output ("Matriks A bukan matriks satuan")

Solusi 2 – Cetak Matriks Satuan B

procedure CetakMatriksSatuanB

{ I.S. MB sudah didefinisikan sebagai variabel global sebagai
array [1..3] of array [1..3] of integer dan sudah terisi }
{ F.S. Tercetak ke layar apakah MB matriks satuan atau bukan }

KAMUS LOKAL

i, j, count : integer

ALGORITMA

count \leftarrow 0

i traversal [1..3]

i traversal [1..3]

if (MB[i][j] \neq 0) and (MB[i][j] \neq 1) then
 count \leftarrow count + 1

if (count = 0) then { jika count = 0, berarti tidak ada elemen bukan 0/1 }
 output ("Matriks B adalah matriks satuan")

else

output ("Matriks B bukan matriks satuan")

Solusi 2: Diskusi

- Perhatikan: masih banyak bagian kode yang “mirip” yang ditulis berulang-ulang, contoh:
 - InputMatriksA, InputMatriksB
 - CetakMatriksSatuanA, CetakMatriksSatuanB
 - Di dalamnya ada bagian memeriksa apakah suatu matriks merupakan matriks satuan/bukan
 - JumlahMatriksAB, KurangMatriksAB
 - Di dalamnya ada bagian mencetak matriks
- Dibanding kode solusi-1 sebenarnya tidak terlalu berbeda

Solusi 3

Manfaatkan abstraksi dan generalisasi pola untuk menangkap parameter-parameter fungsi/prosedur

```
InputMatriks(MA)
```

```
InputMatriks(MB)
```

```
input(pilihan)
```

```
depend on pilihan
```

```
pilihan = 1 : Operasi2Matriks(MA, MB, "+", MHasil)  
              CetakMatriks(MHasil)
```

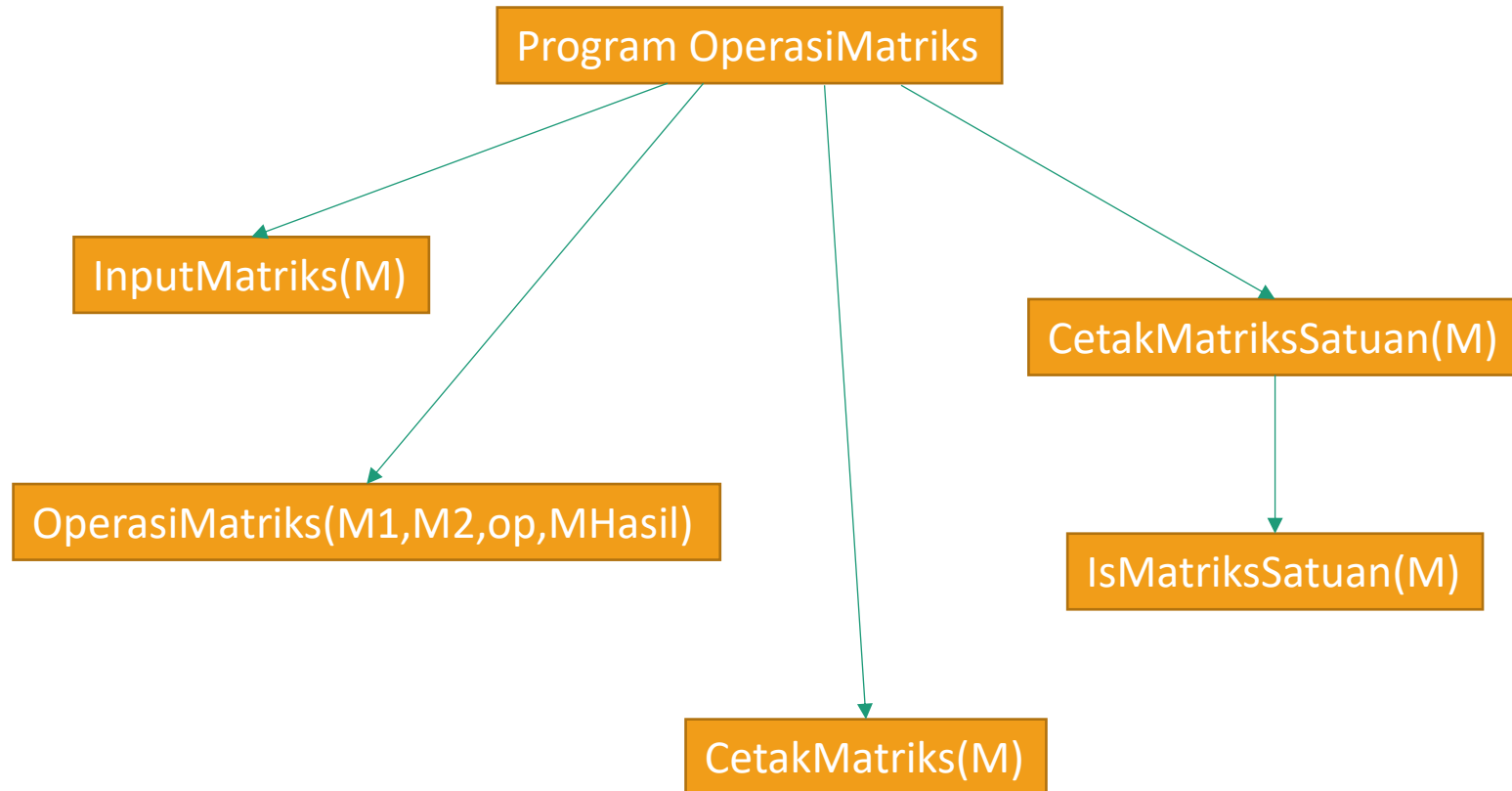
```
pilihan = 2 : Operasi2Matriks(MA, MB, "-", MHasil)  
              CetakMatriks(MHasil)
```

```
pilihan = 3 : CetakMatriksSatuan(MA)  
              CetakMatriksSatuan(MB)
```

```
else : output ("Bukan pilihan yang benar");
```

Call function:
IsMatriksSatuan(MA)
IsMatriksSatuan(MB)

Dekomposisi Fungsional – Solusi 3



Solusi 3 – Input Matriks

```
procedure InputMatriks  
    (output M : array [1..3] of array [1..3] of integer)  
{ I.S. M sembarang }  
{ F.S. M berisi data berdasarkan masukan dari user }
```

KAMUS LOKAL

i, j : integer

ALGORITMA

```
    i traversal [1..3]  
      j traversal [1..3]  
        input(M[i][j])
```

Solusi 3 – Operasi 2 Matriks

procedure Operasi2Matriks

 (input M1 : array [1..3] of array [1..3] of integer;

input M2 : array [1..3] of array [1..3] of integer;

input op : character;

output MHasil : array [1..3] of array [1..3] of integer)

{ I.S. M1 dan M2 sudah terdefinisi dan terisi, op terdefinisi dengan nilai "+" atau "-" }

{ F.S. MHasil berisi hasil operasi M1 dengan M2 tergantung op }

KAMUS LOKAL

 i, j : integer

ALGORITMA

 i traversal [1..3]

 j traversal [1..3]

if (op = "+") then

 MHasil[i][j] ← M1[i][j] + M2[i][j]

else { op = "-" }

 MHasil[i][j] ← M1[i][j] - M2[i][j]

Solusi 3 – Cetak Matriks

```
procedure CetakMatriks  
    (input M : array [1..3] of array [1..3] of integer)  
{ I.S.: M terdefinisi }  
{ F.S.: M tercetak ke layar dalam bentuk matriks 3x3 }
```

KAMUS LOKAL

i, j : integer

ALGORITMA

```
    i traversal [1..3]  
        j traversal [1..3]  
            output (M[i][j], " ");
```

Solusi 3 – Cetak Matriks Satuan

procedure CetakMatriksSatuan
 (input M : array [1..3] of array [1..3] of integer)
{ I.S. M sudah terdefinisi }
{ F.S. Tercetak ke layar apakah MA matriks satuan atau bukan }
KAMUS LOKAL

ALGORITMA

if (IsMatriksSatuan(M)) then
 output(" adalah matriks satuan")
 else
 output(" bukan matriks satuan")

Solusi 3 – Cek Matriks Satuan

function IsMatriksSatuan
 (M : array [1..3] of array [1..3] of integer) → boolean
 { menghasilkan true jika M adalah matriks satuan }

KAMUS LOKAL

i, j, count : integer

ALGORITMA

```
count ← 0
i traversal [1..3]
  j traversal [1..3]
    if (M[i][j] ≠ 0) and (M[i][j] ≠ 1) then
      count ← count + 1
  → (count = 0)
```

Potongan Program Utama – Solusi 3

```
InputMatriks(MA)
InputMatriks(MB)
 (pilihan)
depend on (pilihan)
    pilihan = 1 : Operasi2Matriks(MA, MB, '+', MHasil)
                  CetakMatriks(MHasil)
    pilihan = 2 : Operasi2Matriks(MA, MB, '-', MHasil)
                  CetakMatriks(MHasil)
    pilihan = 3 : output ("Matriks A")
                  CetakMatriksSatuan(MA)
                  output ("Matriks B")
                  CetakMatriksSatuan(MB)
else : output ("Bukan pilihan yang benar")
```

Solusi 3 - Diskusi

- Dibandingkan dengan solusi 1 dan 2, kode yang berulang lebih sedikit: lebih memudahkan *maintenance dan debugging*
- Dekomposisi dalam bentuk fungsi/prosedur:
 - Fungsionalitas tertentu “diserahkan” pada fungsi/prosedur tertentu
 - *Readability* program meningkat
 - Pembagian tugas *programmer* lebih mudah
 - Harus didasarkan pada abstraksi dan generalisasi pola yang baik
 - Interdependensi antar fungsi/prosedur dan program utama
- Penggunaan variabel/parameter bertipe array [1..3] of array [1..3] of integer

Abstraksi Data Matriks (1)

Untuk semua solusi sebelumnya digunakan variabel/parameter bertipe array [1..3] of array [1..3] of integer:

```
{ KAMUS }  
  MA, MB : array [1..3] of array [1..3] of integer  
  MHasil : array [1..3] of array [1..3] of integer  
  
{ Definisi Prosedur, contoh }  
procedure InputMatriks(output M : array [1..3] of array [1..3] of integer)
```

Abstraksi Data Matriks (2)

Gunakan type bentukan untuk menggantikan array of array yang merepresentasikan matriks

```
{ KAMUS }  
type Matriks : array [1..3] of array [1..3] of integer  
MA, MB : Matriks  
MHasil : Matriks  
  
{ Definisi Prosedur, contoh }  
procedure InputMatriks (output M : Matriks)
```

Solusi 4 – Input Matriks

```
procedure InputMatriks (output M : Matriks)  
{ I.S.: M sembarang }  
{ F.S.: Setiap elemen M terdefinisi berdasarkan pembacaan dari  
      keyboard }
```

KAMUS LOKAL

i, j : integer

ALGORITMA

```
  i traversal [1..3]  
    j traversal [1..3]  
      input (M[i][j])
```

**BUAT BAGIAN PROGRAM YANG LAIN
SEBAGAI LATIHAN**

Abstraksi Lebih Lanjut

Struktur Data Matriks

Bagaimana jika ukuran matriks bukan 3x3, tetapi NBrEff x NKolEff ??

```
{ KAMUS }
constant NMax : integer = 10
type Matriks :
    < T : array [1..NMax] of array [1..NMax] of integer,
        NBrEff : integer,
        NKolEff : integer >
MA, MB : Matriks
MHasil : Matriks

{ Definisi Prosedur, contoh }
procedure InputMatriks (output M : Matriks)
```

Abstraksi Lebih Lanjut: ADT Matriks

- Struktur data Matriks dibungkus dalam ADT
- Akan dibahas pada perkuliahan di paruh semester kedua

Translasi Fungsi dan Prosedur dalam Bahasa C

Fungsi (Bahasa C)

```
type-hasil NAMA FUNGSI (type1 param1, type2 param2, ...) {  
    /* Spesifikasi fungsi */  
  
    /* KAMUS LOKAL */  
    /* Semua nama yang dipakai dalam algoritma dari fungsi */  
  
    /* ALGORITMA */  
    /* Deretan fungsi algoritmik: pemberian harga, input, output, analisis kasus,  
       pengulangan */  
  
    /* Pengiriman harga di akhir fungsi, harus sesuai  
       dengan type-hasil */  
    return hasil;  
}
```

Pemanggilan fungsi (Bahasa C)

```
/* Program POKOKPERSOALAN */
/* Spesifikasi: Input, Proses, Output */
/* Prototype Fungsi */
type-hasil NMAFUNGSI ([list <type nama> input]);
/* Spesifikasi Fungsi */
int main () {
    /* KAMUS */
    /* Semua nama yang dipakai dalam algoritma */
    /* ALGORITMA */
    /* Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan yang memakai fungsi */
    /* Harga yang dihasilkan fungsi juga dapat dipakai dalam ekspresi */
    nama = NMAF ([list parameter aktual]);
    printf ("[format]", NMAF ([list parameter aktual]));
    /* Harga yang dihasilkan fungsi juga dapat dipakai dalam ekspresi */

    return 0;
}
/* Body/Realisasi Fungsi diletakkan setelah program utama */
type-hasil NMAFUNGSI ([list <type nama> input]) {
    /* Spesifikasi Fungsi */
}
```

Prosedur (Bahasa C)

```
void NAMAPROSEDUR (type1 nama1, type2 *nama2, type3 *nama3)
/* Spesifikasi, Initial State, Final State */
{
    /* KAMUS LOKAL */
    /* Semua nama yang dipakai dalam BADAN PROSEDUR */

    /* ALGORITMA */
    /* Deretan instruksi pemberian harga, input, output, analisis kasus,
       pengulangan, atau prosedur */

}
```

Pemanggilan Prosedur (Bahasa C)

```
/* Program POKOKPERSOALAN */
/* Spesifikasi: Input, Proses, Output */

/* Prototype prosedur */
void NAMAP (type1 nama1, type2 *nama2, type3 *nama3);
/* Spesifikasi prosedur, initial state, final state */

int main () {
    /* KAMUS */
    /* semua nama yang dipakai dalam algoritma */

    /* ALGORITMA */
    /* Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan */
    NAMAP(paramaktual1, &paramaktual2, &paramaktual3);

    return 0;
}

/* Body/Realisasi Prosedur diletakkan setelah program utama */
```

Translasi parameter formal/aktual prosedur

NOTASI ALGORITMIK	BAHASA C
Definisi/deklarasi Prosedur (parameter formal)	
<pre> <u>procedure</u> NAMAP (<u>input</u> nama1: type1, <u>input/output</u> nama2: type2, <u>output</u> nama3: type3) </pre>	<pre> void NAMAP (type1 nama1, type2 *nama2, type3 *nama3); </pre>
Pemanggilan prosedur (parameter aktual)	
<pre> NAMAP (namaaktual1, namaaktual2, namaaktual3) </pre>	<pre> NAMAP (namaaktual1, &namaaktual2, &namaaktual3); </pre>

Latihan Soal

Semua soal latihan berikut dikerjakan dalam notasi algoritmik, kecuali dinyatakan lain.

Latihan Soal 1

- Tuliskan fungsi **Max2** yang menerima masukan 2 buah bilangan real dan menghasilkan sebuah bilangan real, yaitu salah satu dari kedua bilangan tersebut yang terbesar.
- Kemudian dengan menggunakan fungsi Max2, tuliskan fungsi lain **Max3** yang menghasilkan nilai terbesar dari 3 buah bilangan real
- Contoh:
 - $\text{Max2}(1,2) \rightarrow 2$
 - $\text{Max2}(10,2) \rightarrow 10$
 - $\text{Max3}(1,2,3) \rightarrow 3$
 - $\text{Max3}(10,2,3) \rightarrow 10$

Latihan Soal 2

- Didefinisikan type bentukan Tanggal untuk mewakili hari seperti pada kalender yang kita pakai sehari-hari

```
type Tanggal : < DD : integer[1..31], { hari }  
                MM : integer[1..12], { bulan }  
                YY : integer > 0      { tahun }  >
```

- Contoh konstanta: < 5, 12, 2024> adalah 5 Desember 2024
- Buatlah fungsi **NextDay** yang menerima masukan sebuah Tanggal dan menghasilkan tanggal keesokan harinya.
- Contoh:
 - NextDay(<13,4,2000>) → <14,4,2000>
 - NextDay(<30,4,2024>) → <1,5,2024>

Latihan Soal 3

- Buatlah prosedur **Tukar** yang menukar dua nilai integer yang tersimpan dalam dua buah nama, misalnya a dan b.
 - I.S. diberikan $a = A$ dan $b = B$
 - F.S. $a = B$ dan $b = A$
- Menggunakan prosedur Tukar, buatlah prosedur **Putar3Bil** yang digunakan untuk “memutar” 3 buah nama integer.
 - Contoh: Jika $a = 1, b = 2, c = 3$, maka $a = 3, b = 1, c = 2$

Latihan Soal 4 - 1

- Buatlah program lengkap yang menerima masukan tanggal hari ini bertipe Tanggal (lihat definisi type pada soal 2) dan menuliskan tanggal hari ini dan keesokan harinya ke layar.
- Beberapa hal yang harus dilakukan:
 - Input data Tanggal harus dibuat dalam suatu prosedur bernama **BacaTanggal**. BacaTanggal yang menghasilkan sebuah tanggal yang valid (apa definisi tanggal valid?). Gunakan skema validasi dengan pengulangan (skema validasi II) untuk mendapatkan Tanggal valid.
 - Buat fungsi **IsTanggalValid** yang menerima masukan 3 buah integer yang merepresentasikan masukan untuk hari, bulan, dan tahun dan menghasilkan true jika masukan hari, bulan, tahun dapat menghasilkan bulan valid.

Latihan Soal 4 - 2

- Beberapa hal yang harus dilakukan:
 - Menuliskan data Tanggal ke layar harus menggunakan sebuah prosedur bernama **TulisTanggal** yang menerima masukan bertipe Tanggal dan menuliskan tanggal ke layar dalam bentuk <hari> <nama bulan> <tahun>.
 - Contoh: Tanggal <13,4,2024> akan dicetak sebagai 13 April 2024.
 - Buatlah fungsi **NamaBulan** yang digunakan untuk mengkonversi bulan (integer[1..12]) menjadi nama bulan (string) dalam Bahasa Indonesia Januari, Februari, ..., dst.
 - Gunakan fungsi **NextDay** yang telah dibuat pada latihan soal 2 untuk mendapatkan keesokan hari suatu Tanggal.

Latihan Soal 5

- Translasikan semua hasil latihan soal 1 s.d. 4 dalam Bahasa C.