IF1210 Dasar Pemrograman

Dekomposisi, Abstraksi dan Generalisasi Pola dalam Konteks Pemrograman Prosedural

Tim Pengajar IF1210

Sekolah Teknik Elektro dan Informatika

Sumber Utama

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

Problem Decomposition

Functional decomposition:

suatu fungsionalitas dipecah menjadi fungsionalitas-fungsionalitas yang lebih kecil yang jika dikomposisi/direkonstruksi ulang akan membentuk kembali fungsionalitas asal

Algorithmic decomposition:

sebuah proses dipecah menjadi langkah-langkah yang terdefinisi

Object-oriented decomposition:

sebuah sistem dipecah menjadi kelas-kelas objek yang lebih kecil yang bertanggung jawab terhadap suatu bagian tertentu dari sistem

Tidak dibahas lebih lanjut di kuliah ini

Studi Kasus 1

Mengupas kentang untuk mempersiapkan makan malam

Beberapa pertanyaan:

- Apakah kentang sudah ada di dapur atau harus dibeli dulu?
- Yang dimaksud mengupas kentang untuk makan malam apakah berarti sampai kentang terhidang dalam bentuk masakan atau hanya sampai terkupas?
- Bagaimana menentukan jumlah kentang yang dikupas sudah cukup?

Proses [sekuensial]

- terdiri atas beberapa **aksi** yang terjadi **berurut-urutan**

Aksi

- kejadian dalam **selang waktu terbatas**
- menghasilkan **efek neto** yang telah **terdefinisi** dengan baik dan direncanakan
- Jelas **initial state** dan **final state**

Dekomposisi **Algoritmik** dalam Konteks Pemrograman Prosedural

PROSES MENYIAPKAN MAKAN MALAM

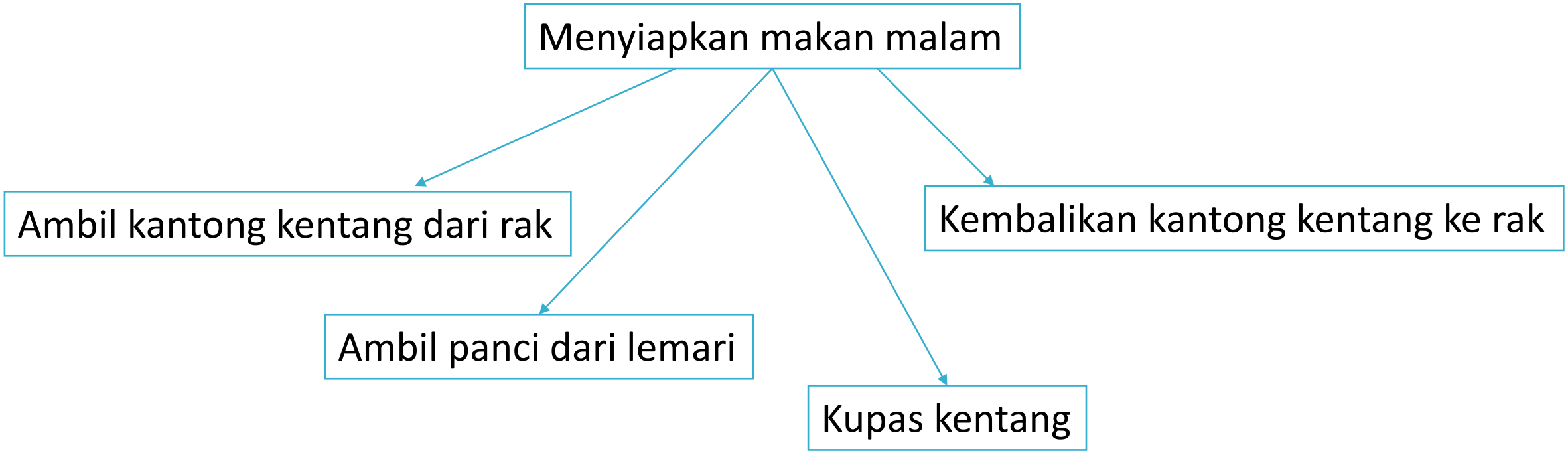
{ I.S. : kentang dalam kantong, diletakkan di rak di dapur
Batasan: diasumsikan bahwa tersedia cukup kentang untuk makan malam}

Ambil kantong kentang dari rak
{ F.S./I.S. : kantong kentang terletak di atas meja dapur, kentang siap dikupas }
Ambil panci dari lemari
{ F.S./I.S.: kantong kentang dan panci terletak di atas meja dapur, siap dipakai menampung kentang yang sudah terkupas }
Kupas kentang
{ F.S/I.S.: kentang dalam keadaan terkupas dan siap dimasak, diletakkan dalam panci }
Kembalikan kantong kentang ke rak

Urut-urutan aksi

{ F.S. : kentang dalam keadaan terkupas dan siap dimasak, diletakkan di dalam panci dan kantong kentang dikembalikan lagi ke rak }

Dekomposisi **Fungsional** dalam Konteks Pemrograman Prosedural



Pola [Tingkah Laku]

Proses/aksi disusun mengikuti suatu **pola [tingkah laku]**

I.S. dan F.S. ditentukan sepenuhnya oleh pola

Pola didapatkan dari "pengamatan" pada beberapa kejadian

Asumsikan selalu tersedia kentang dalam jumlah cukup dalam kantong

PENGAMATAN HARI 1

Ibu Tati mengambil kantong kentang dari rak
Ibu Tati mengambil panci dari lemari
Ibu Tati memakai celemek
Ibu Tati mengupas 1 kentang
Ibu Tati mengupas 1 kentang
Ibu Tati mengupas 1 kentang
Ibu Tati mengupas 1 kentang
Ibu Tati mengupas 1 kentang
Ibu Tati mengembalikan kantong kentang ke rak

PENGAMATAN HARI 2

Ibu Tati mengambil kantong kentang dari rak
Ibu Tati mengambil panci dari lemari
Ibu Tati mengupas 1 kentang
Ibu Tati mengupas 1 kentang
Ibu Tati mengupas 1 kentang
Ibu Tati mengupas 1 kentang
Ibu Tati mengupas 1 kentang
Ibu Tati mengembalikan kantong kentang ke rak

PENGAMATAN HARI 3

Ina mengambil kantong kentang dari rak
Ina mengambil panci dari almari
Ina memakai celemek
Ina mengupas 1 kentang
... } 100 kali
Ina mengupas 1 kentang
Ina mengembalikan kantong kentang ke rak

PENGAMATAN HARI 4

Aida mengambil kantong kentang dari rak
Aida mengambil panci dari almari
Aida mengupas 1 kentang
... } 50 kali
Aida mengupas 1 kentang
Aida mengembalikan kantong kentang ke rak

Apa yang **sama**?

{ I.S. : kentang dalam kantong, diletakkan di rak di dapur }

Ambil kantong kentang dari rak
Ambil panci dari lemari
Kupas 1 kentang
Kembalikan kantong kentang ke rak

{ F.S. : kentang dalam keadaan terkupas dan siap dimasak, diletakkan di dalam panci dan kantong kentang dikembalikan lagi ke rak }

Apa yang **berbeda**?

- **Orang** yang melakukannya
  - **Informasi ini dipandang tidak penting**
- **Memakai celemek atau tidak**
  - Berdasarkan pengamatan lebih lanjut, didapat bahwa seseorang memakai celemek kalau sedang memakai baju berwarna cerah
- Berapa **banyak kentang** yang dikupas?
  - Berdasarkan pengamatan lebih lanjut, didapat bahwa banyaknya kentang yang dikupas adalah berdasarkan jumlah orang yang makan (kalau sampai ratusan biasanya ada pesta)

Generalisasi & Abstraksi Pola

{ I.S. : kentang dalam kantong, diletakkan di rak di dapur }

Ambil kantong kentang dari rak

Ambil panci dari lemari

if <warna baju> adalah cerah then pakai celemek

repeat <jumlah peserta makan malam> times

    Kupas 1 kentang

Kembalikan kantong kentang ke rak

{ F.S. : kentang dalam keadaan terkupas dan siap dimasak, diletakkan di dalam panci dan kantong kentang dikembalikan lagi ke rak }

Algoritma

deskripsi dari suatu pola tingkah laku,

dinyatakan dalam **primitif**, yaitu: **aksi-aksi** yang didefinisikan sebelumnya dan diberi nama, dan diasumsikan sebelumnya bahwa aksi-aksi tersebut dapat dikerjakan sehingga dapat menyebabkan kejadian yang dapat diamati

Studi Kasus 2

Sebuah program mengelola dua buah bilangan bulat, misalnya diberi nama A dan B, dan mencetak sesuatu di layar tergantung nilai A dan B. Beberapa contoh kejadian:

A	B	Tercetak di layar
2	10	30
2	2	2
9	2	20
0	2	2
0	0	0
-1	5	Tidak bisa dihitung
5	0	6
-6	-5	Tidak bisa dihitung
-4	5	Tidak bisa dihitung
1	1	0

Analisis Kejadian

- Jika A >= 0 dan B >= 0 dan A < B, tercetak hasil berupa angka integer yang merupakan hasil penjumlahan nilai semua bilangan integer genap di antara A dengan B (A dan B termasuk).
  - Jika B < A, maka yang tercetak adalah penjumlahan semua integer genap antara B dengan A (A dan B termasuk)
- Jika A < 0 atau B < 0, maka tercetak “Tidak dapat dihitung”.

I.S. A dan B terdefinisi (masing-masing memiliki nilai dari domain bilangan bulat)
F.S. Jika A >= 0 dan B >= 0, tercetak hasil penjumlahan nilai semua bilangan integer genap di antara A dengan B (jika A <= B) atau B dengan A (jika B < A). Jika A < 0 atau B < 0, maka tercetak "Tidak dapat dihitung".

Algoritma (dalam notasi algoritmik):

```
input (A, B) { I.S. }
if A >= 0 and B >= 0 then
    if A <= B then
        hitung hasil penjumlahan integer genap antara A dan B, simpan dalam sum
    else
     hitung hasil penjumlahan integer genap antara B dan A, simpan dalam sum
    output (sum) { F.S. }
else { A < 0 or B < 0 }
        output ("Tidak dapat dihitung") { F.S. }
```

Bagaimana menghitung hasil penjumlahan integer genap antara X dan Y?

Alternatif:
1. Menggunakan pengulangan
2. Perhitungan secara langsung menggunakan rumus matematika

Alternatif-1 (salah satu algoritma yang mungkin):

```
i ← X
sum ← 0
while i <= Y do
    if i mod 2 = 0 then
        sum ← sum + i
        i ← i + 2
    else
        i ← i + 1
{ i > Y }
```

Bagaimana persisnya melakukan dekomposisi aksi-aksi yang dibutuhkan untuk algoritma diserahkan kepada mahasiswa untuk dipikirkan

Alternatif-2:

```
sum ← ((Y div 2)^2 + (Y div 2)) - (((X-1) div 2)^2 + ((X-1) div 2))
```

Salah satu implementasi **dekomposisi** dalam pemrograman prosedural adalah melalui *modular programming*

Modular programming

- teknis mendesain program di mana fungsionalitas program dipisah-pisahkan ke dalam modul-modul independen, masing-masing berisi hal-hal yang dibutuhkan untuk mengeksekusi “satu” aspek dari fungsionalitas keseluruhan

Implementasi Program Modular

Subprogram
- Fungsi
- Prosedur

Abstract Data Type

Modul/paket/ unit

Kelas
- Pemrograman berorientasi objek

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

Studi Kasus

Studi Kasus

- Buatlah sebuah program yang digunakan untuk melakukan operasi matriks dengan elemen bertype 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

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 bertype 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

```
Program Matriks3x3
{ Input: 2 matriks of integer 3x3, mis. matriks A dan B pilihan menu (integer) }
{ Output: jika pilihan menu = 1, dituliskan hasil perkalian kedua matriks
          jika pilihan menu = 2, dituliskan hasil penjumlahan kedua matriks
          jika pilihan menu = 3, dituliskan apakah kedua matriks adalah matriks
          satuan atau bukan
          jika pilihan menu lain, dituliskan "Bukan pilihan yang benar" }

{ ALTERNATIF PROGRAM TIDAK MENGGUNAKAN TYPE BENTUKAN DAN FUNGSI/PROSEDUR }
```

```
KAMUS
    MA, MB, MHasil : array [1..3] of array [1..3] of integer
    pilihan : integer
    i, j : integer
    count : integer
```

```
ALGORITMA
    { Mengisi matriks }
    output("Masukan Matriks A = ")
    i traversal [1..3]
        j traversal [1..3]
            output("Elemen ke-[", i, ",", j, "] = ")
            input(MA[i][j])
    output("Masukan Matriks B = ")
    i traversal [1..3]
        j traversal [1..3]
            output("Elemen ke-[", i, ",", j, "] = ")
            input(MB[i][j])

    { 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]
```

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):

```
{ input Matriks A }
{ input Matriks B }

input (pilihan)

depend on pilihan
   pilihan = 1 : { jumlahkan matriks A dan B; tampung dalam MHasil }
                 { cetak matriks MHasil }
   pilihan = 2 : { kurangkan matriks A dengan B; tampung dalam MHasil }
                 { cetak matriks MHasil }
   pilihan = 3 : { cetak apakah matriks A adalah matriks satuan }
                 { cetak apakah matriks B adalah matriks satuan }

   else : output (“Bukan pilihan yang benar”)
```

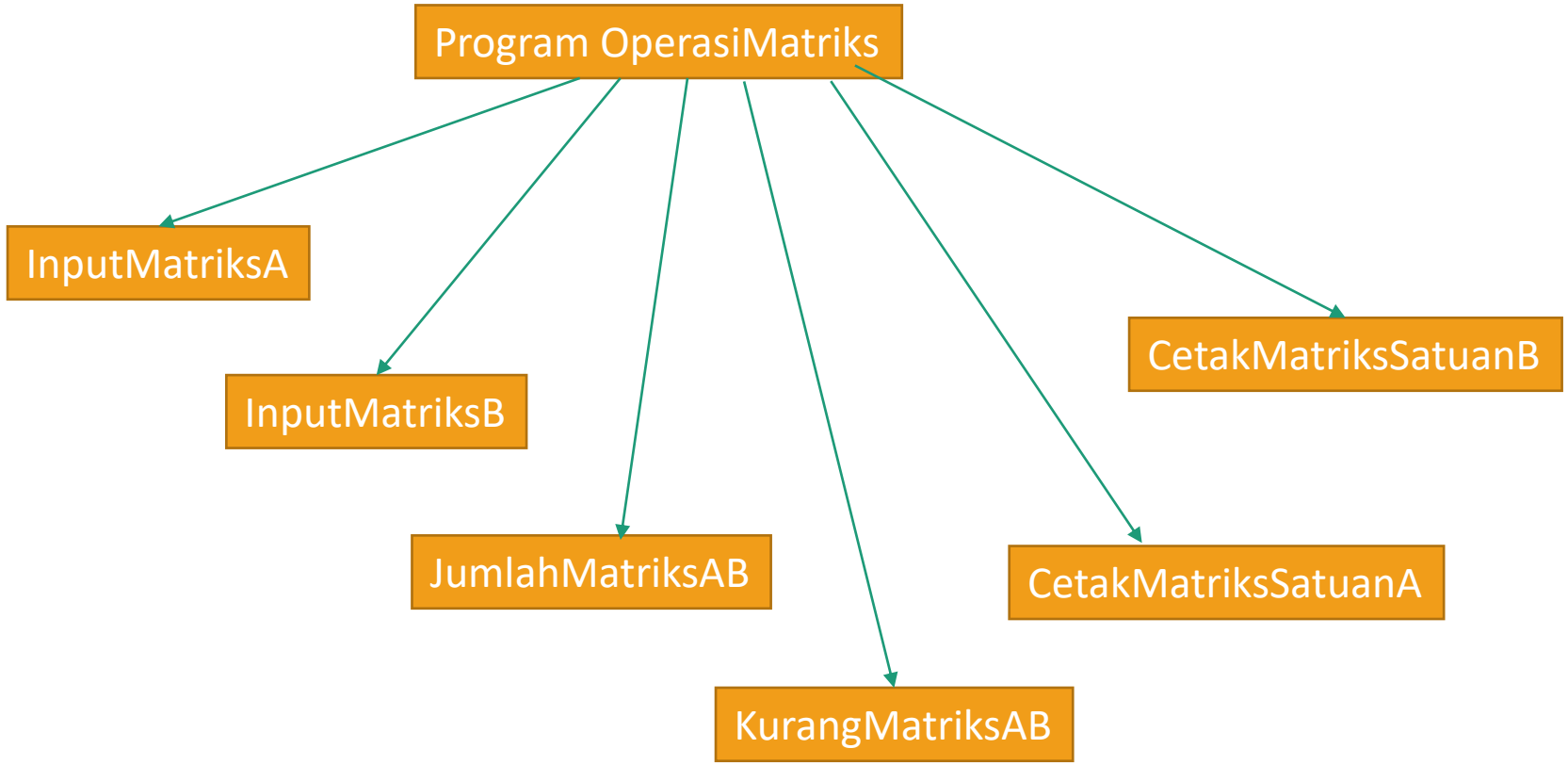
Solusi 2

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 ← 0
       i traversal [1..3]
              i traversal [1..3]
                     if (MA[i][j] ≠ 0) and (MA[i][j] ≠ 1) then
                       count ← 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 ← 0
       i traversal [1..3]
              i traversal [1..3]
                     if (MB[i][j] ≠ 0) and (MB[i][j] ≠ 1) then
                       count ← 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

```
{ input Matriks A }
{ input Matriks B }

input(pilihan)

depend on pilihan
  pilihan = 1 : { jumlahkan matriks A dan B; tampung dalam MHasil }
                { cetak matriks MHasil }
  pilihan = 2 : { kurangkan matriks A dengan B; tampung dalam MHasil }
                { cetak matriks MHasil }
  pilihan = 3 : { cetak apakah matriks A adalah matriks satuan }
                { cetak apakah matriks B adalah matriks satuan }

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

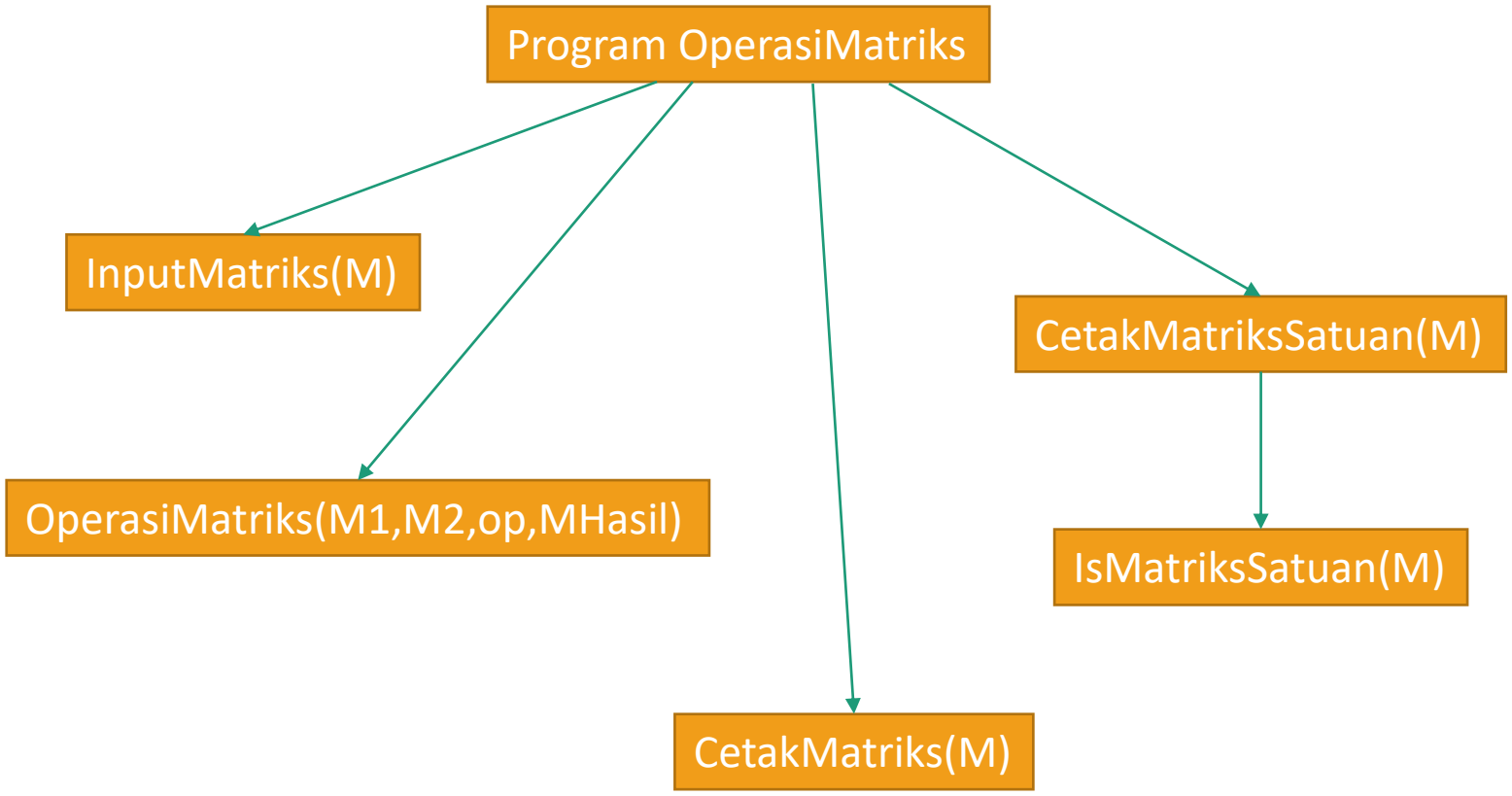
Solusi 3

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)
input (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 bertype array [1..3] of array [1..3] of integer

Abstraksi Data Matriks (1)

- Untuk semua solusi di atas digunakan variabel/parameter bertype 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)
```

- Di notasi algoritmik dan beberapa bahasa pemrograman dimungkinkan membuat type bentukan: yaitu type data yang dibuat oleh programmer

Type Bentukan

- Tipe data bentukan/komposit/ record
  - Tidak tersedia secara otomatis, harus dibuat oleh programmer
  - Dibentuk dari gabungan tipe dasar

```
type JAM :
< JJ : integer[0..23],
  MM : integer[0..59],
  DD : integer[0..59] >
```

```
type POINT:
 < X : integer {absis},
   Y : integer {ordinat} >
```

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)
```

Dekomposisi algoritmik dan fungsional sesuai dengan solusi 3, namun penyesuaian dilakukan menggunakan type data Matriks → **solusi 4**

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 NBrsEff x NKolEff ??

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

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

Selamat Belajar