10 PROSEDUR

Seringkali dalam membuat program besar, pemrogram perlu memecah program menjadi beberapa subprogram yang lebih kecil. Tiap subprogram(modul) dapat dirancang oleh pemrogram selain orang yang mengembangkan program utama. Modul yang sudah dirancang dapat dipasang ke dalam program lain yang membutuhkan → Teknik pemrograman modular(prosedur, *routine*, fungsi)

Keuntungan modularisasi :

1. Untuk aktivitas yang harus dilakukan lebih dari sekali, cukup ditulis sekali sehingga dapat mengurangi panjang program. Contoh :

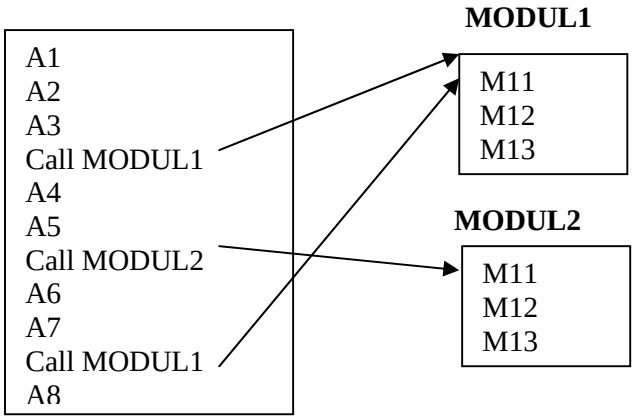
```
Algoritma ABCD
DEKLARASI
        A, B, C, D, temp : integer

DESKRIPSI
…..
{Pertukarkan nilai A dan B}
temp ← A
A ← B
B ← temp
.....
if C > D then
      {pertukarkan nilai C dan D}
      temp ← C
      C ← D
      D ← temp
endif
```

```
Procedure TUKAR(input/output P, Q : integer)
{mempertukarkan nilai P dan Q}
DEKLARASI
      Temp : integer

DESKRIPSI
      Temp ← P
      P ← Q
      Q ← Temp
```

```
Algoritma ABCD
DEKLARASI
      A, B, C, D, temp : integer
      Procedure TUKAR(input/output P, Q : integer)

DESKRIPSI
…..
{Pertukarkan nilai A dan B}
TUKAR(A,B)                    {panggil prosedur TUKAR}
.....
if C > D then
      {pertukarkan nilai C dan D}
      TUKAR(C,D)              {panggil prosedur TUKAR}
endif
```

Ketika sebuah program dipanggil, pelaksanaan program berpindah ke dalam modul. Lalu seluruh instruksi dalam modul dilaksanakan secara beruntun sampai akhir modul. Setelah instruksi dalam modul dilaksanakan, pelaksanaan program kembali ke program utama.

Program Utama



Ilustrasi :

a. Prosedur URUS PASPOR (di kantor imigrasi)
   - Isi formulir permintaan paspor dengan lampiran foto copy KTP, Kartu keluarga, pas foto
   - Serahkan formulir yang sudah diisi beserta biaya pembuatan paspor
   - Wawancara dengan petugas imigrasi
   - Terima paspor
b. Prosedur URUS VISA (di kantor kedutaan besar)
   - Isi formulir permintaan visa dengan lampiran foto copy KTP, paspor, pas foto, tiket pesawat
   - Serahkan formulir yang sudah diisi beserta biaya pembuatan visa
   - Terima visa
c. Prosedur BERANGKAT DARI BANDARA
   - Datang ke bandara satu jam sebelum keberangkatan
   - Jika sudah disuruh naik ke pesawat, tunjukkan tiket, paspor, dan visa ke petugas
   - Naik ke pesawat
   - Selamat jalan...

Algoritma PERGI_KE_LUAR_NEGERI

DESKRIPSI :

a. Urus Paspor
b. URUS VISA
c. BERANGKAT DARI BANDARA

2. Kemudahan menulis dan mencari kesalahan(*debug*) program

   Kemudahan menulis berguna jika sebuah program dilaksanakan oleh satu tim pemrogram. Masalah dipecah menjadi beberapa submasalah. Setiap submasalah ditulis ke dalam modul individual yang dikerjakan oleh orang yang berbeda. Setelah selesai, semua modul diintegrasikan kembali menjadi program lengkap.

   Program modular mudah dipahami dan mudah dicari kesalahannya karena setiap modul melakukan aktivitas spesifik

STRUKTUR PROSEDUR

1. Bagian *Header* → nama prosedur dan komentar yang menjelaskan spesifikasi prosedur
2. Bagian Kamus
3. Badan Prosedur (Deskripsi)

Nama prosedur sebaiknya diawali kata kerja, misalnya Hitung_Luas, Tukar, CariMaks

```
Procedure Nama_Prosedur
{Spesifikasi prosedur, penjelasan yang berisi uraian singkat mengenai apa yang dilakukan prosedur}

Kamus
{semua nama yang dipakai dalam prosedur dan hanya berlaku lokal didefinisikan disini}

Deskripsi
{Badan prosedur, berisi kumpulan instruksi}
```

Contoh :

```
Procedure CETAK_HALLO
{mencetak string ”Hallo, Dunia”}
DEKLARASI

DESKRIPSI
write(‘Hallo, Dunia’)
```

```
Procedure HIT_LUAS_SEGITIGA
{menghitung luas segitiga dengan rumus L = ½ (alas x tinggi)}
DEKLARASI
a, t , L: real

DESKRIPSI
read(a , t)
L ← a * t / 2
write(L)
```

```
Procedure HIT_LUAS_ PERSEGI_PANJANG
{menghitung luas empat persegi panjang dengan rumus L = panjang x lebar}
DEKLARASI
p, l , Luas: real

DESKRIPSI
read(p , l)
Luas ← p * l
write(‘Luas = ’, Luas)
```

```
Procedure HIT_LUAS_LINGKARAN
{menghitung luas segitiga dengan rumus L = ½ (a x t)}
DEKLARASI
const phi = 3.14
r , L: real

DESKRIPSI
read(r)
L ← phi * r * r
write(L)
```

PEMANGGILAN PROSEDUR

Prosedur tidak bisa dieksekusi langsung. Instruksi pada prosedur bisa dilaksanakan jika prosedur diakses. Prosedur diakses dengan memanggil namanya dari program pemanggil (program utama atau modul program lain)

NAMA_PROSEDUR

Ketika nama prosedur dipanggil, kendali program berpindah ke prosedur tersebut. Setelah semua instruksi prosedur selesai dilaksanakan, kendali program berpindah kembali ke program pemanggil. Dalam program pemanggil, harus mendeklarasikan *prototype* prosedur (*header*) dalam bagian deklarasi supaya dikenali oleh program pemanggil dan mengetahui cara mengaksesnya.. **Contoh :**

```
Algoritma HALLO
{program utama untuk mencetak string ”Hallo, Dunia”}
DEKLARASI
    Procedure CETAK_HALLO

DESKRIPSI
CETAK_HALLO        {panggil prosedur CETAK_HALLO}
```

```
Algoritma HITUNG_LUAS_SEGITIGA
{program utama untuk menghitung luas segitiga}
DEKLARASI
    Procedure HIT_LUAS_SEGITIGA

DESKRIPSI
write(‘Menghitung Luas Sebuah Segitiga’)
HIT_LUAS_SEGITIGA                    {panggil prosedur HIT_LUAS_SEGITIGA}
write(’Selesai’)
```

```
Algoritma HITUNG_LUAS
{program utama untuk menampilkan menu perhitungan luas segitiga, luas persegi panjang, dan
lingkaran, memilih menu, dan melakukan proses perhitungan sesuai pilihan menu}
DEKLARASI
    nomor_menu : integer
    Procedure HIT_LUAS_SEGITIGA
    Procedure HIT_LUAS_PERSEGI_PANJANG
    Procedure HIT_LUAS_LINGKARAN

DESKRIPSI

Repeat
        {cetak menu ke layar}
        write(‘        # Menu Pilihan Menghitung Luas #        ’)
        write(’          1. Menghitung Luas Segitiga          ’)
        write(’          2. Menghitung Luas Persegi Panjang   ’)
        write(’          3. Menghitung Luas Lingkaran         ’)
        write(’          4. Keluar Program                    ’)
        write(’        Masukkan Pilihan Menu (1 / 2 / 3 / 4) : ’)
        read(nomor_menul)
        depend on ( nomor_menu)
            1 : HIT_LUAS_SEGITIGA
            2 : HIT_LUAS_PERSEGI_PANJANG
            3 : HIT_LUAS_LINGKARAN
            4 : write(’Keluar Program...Sampai Jumpa’)
until nomor_menu = 4
```

VARIABE GLOBAL DAN VARIABEL LOKAL

Variabel Lokal → Ditulis pada bagian Kamus prosedur dan hanya dapat digunakan oleh prosedur
Variabel Global → Ditulis pada bagian Kamus program utama, dapat digunakan di dalam program utama maupun prosedur. **Contoh :**

```
Procedure Hitung_Rata_Rata
{program utama untuk menghitung rata-rata N buah bilangan bulat yang dibaca dari keyboard}
DEKLARASI
    x : integer        {data bilangan bulat yang dibaca dari keyboard}
    k : integer        {pencacah banyak bilangan}
    jumlah : real      {jumlah seluruh bilangan}
DESKRIPSI
    k ← 1
    jumlah ← 0
    while k ≤ N do
        read(x)
        jumlah ← jumlah + x
        k ← k + 1
    endwhile
    rata ← jumlah / N
```

```
Algoritma Rata_Rata_Bilangan_Bulat
{program utama untuk menghitung rata-rata N buah bilangan bulat yang dibaca dari keyboard }
DEKLARASI
    N : integer        {banyaknya bilangan bulat }
    rata : real        {nilai rata-rata bilangan bulat}
    Procedure Hitung_Rata_Rata

DESKRIPSI
read(N)
write('Menghitung rata-rata bilangan bulat')
Hitung_Rata_Rata
write('Nilai rata-rata : ',rata)
```

Usahakan menggunakan nama global sesedikit mungkin karena dengan nama lokal, program terlihat lebih elegan dan mempermudah mencari kesalahan program yang disebabkan oleh nama tersebut

Parameter

Kebanyakan program memerlukan pertukaran informasi antara prosedur / fungsi dengan titik dimana ia dipanggil → penggunaan parameter

Parameter adalah :

Nama- nama peubah yang dideklarasikan pada bagian *header* prosedur.

Parameter actual (argument) adalah :

Parameter yang disertakan pada waktu pemanggilan prosedur.

Parameter formal adalah :

Parameter yang dideklarasikan di dalam bagian *header* prosedur itu sendiri.

Tiap item data ditransfer antara parameter aktual(yang disertakan pada waktu pemanggilan) dengan parameter formal(yang dideklarasikan di prosedur). Ketika pemanggilan, parameter aktual menggantikan parameter formal. Tiap parameter berpasangan dengan parameter formal yang bersesuaian. Pendeklarasian parameter di dalam prosedur bukanlah keharusan. Dengan kata lain boleh ada atau tidak ada.

```
Procedure Nama_Prosedur(daftar parameter formal)
{Spesifikasi prosedur, penjelasan yang berisi uraian singkat mengenai apa yang dilakukan
prosedur}

Kamus
{semua nama yang dipakai dalam prosedur dan hanya berlaku lokal didefinisikan disini}

Algoritma
{Badan prosedur, berisi kumpulan instruksi}
```

Memanggil prosedur dengan parameter :

NAMA_PROSEDUR(daftar parameter actual)

Prosedur yang baik adalah

Prosedur yang *independent* dari program pemanggilannya. Prosedur yang tidak menggunakan peubah-peubah global didalam badan prosedurnya.

Jika program utama perlu mengkomunikasikan nilai peubah global ke dalam prosedur, maka ada satu cara untuk melakukannya yaitu dengan menggunakan parameter.

Aturan penting korespondensi satu-satu antara parameter formal dengan parameter aktual :

1. Jumlah parameter aktual harus sama dengan jumlah parameter formal
2. Tipe parameter aktual harus sama dengan tipe parameter formal
3. Tiap parameter aktual harus diekspresikan dengan cara yang benar dengan parameter formal bersesuaian, tergantung jenis parameter formal

Jenis parameter formal :

1. Parameter Masukan(*input parameter*) = parameter nilai(*value parameter*) dalam bahasa pemrograman → nilainya berlaku sebagai masukan untuk prosedur
   Nilai parameter aktual diisikan(*assign*) ke parameter formal bersesuaian. Nilai tersebut digunakan dalam badan prosedur tetapi tidak dapa dikirimkan ke titik pemanggilan .
   Perubahan nilai parameter dalam badan prosedur tidak mengubah nilai parameter aktual. Nama parameter aktual boleh berbeda dengan nama parameter formal. **Contoh :**

```
Procedure SATU (input x : integer , input y : real)
{Contoh prosedur dengan parameter formal jenis parameter masukan}

Deklarasi

Deskripsi
x ← x + 1
y ← y + 1
write(x)
write(y)
```

```
Algoritma PQR
{Contoh program utama yang memanggil prosedur SATU}

Deklarasi
    a , b : integer
    c , d : real
    Procedure SATU (input x : integer , input y : real)

Deskripsi
    SATU(4,10.5)
    read(a,b,c,d)
    SATU(a,c)
    SATU(b,d)
    SATU(a+5,c/d)
    SATU(a,b)
```

Yang tidak boleh :

SATU(c,d) SATU(a) SATU(a,c,b)

```
Procedure HIT_LUAS_SEGITIGA(input a , t : real)
{menghitung luas segitiga dengan rumus L = ½ (alas x tinggi)}
DEKLARASI
L: real

DESKRIPSI
L ← a * t / 2
write('Luas Segitiga = ',L)
```

```
Algoritma HITUNG_LUAS_SEGITIGA
{program utama untuk menghitung luas segitiga}
DEKLARASI
    alas , tinggi : real
    Procedure HIT_LUAS_SEGITIGA(input a , t : real)

DESKRIPSI
write('Menghitung Luas Sebuah Segitiga')
read(alas,tinggi)
HIT_LUAS_SEGITIGA(alas,tinggi)
write('Selesai')
```

Parameter formal ← Nilai parameter aktual

2. Parameter Keluaran(*output parameter*) → menampung keluaran yang dihasilkan oleh prosedur

   Bila prosedur menghasilkan satu atau lebih nilai yang akan digunakan oleh program pemanggil, maka nilai keluaran ditampung di dalam parameter keluaran. Bila prosedur dengan parameter keluaran dipanggil, nama parameter aktual dalam program pemanggil akan menggantikan nama parameter formal yang bersesuaian dalam prosedur. Nama parameter aktual akan digunakan selama pelaksanaan prosedur.

   Parameter formal ← Parameter aktual

   Karena nama parameter merupakan suatu lokasi di memori, maka jika parameter aktual diisi suatu nilai di dalam prosedur, nilai tersebut akan tetap berada dalam parameter aktual walaupun prosedur selesai dilaksanakan. **Contoh :**

```
Procedure DUA(input x : integer , output y : real)
{Contoh prosedur dengan parameter formal jenis parameter masukan dan jenis parameter keluaran}

Deklarasi

Deskripsi
x ← x + 1
y ← x * 10
```

```
Algoritma PQR
{Contoh program utama yang memanggil prosedur DUA}

Deklarasi
    a , b : integer
    Procedure DUA (input x : integer , output y : real)

Deskripsi
    DUA (4,b)
    write(b)
    read(a)
    DUA (a,b)
    write(b)
    DUA (a+5,b)
    write(b)
```

Yang tidak boleh : output berupa nilai atau ekspresi

DUA(4,8.5) DUA(a,a+5)

```
Procedure HIT_LUAS_SEGITIGA(input a , t : real, output L : real)
{menghitung luas segitiga dengan rumus L = ½ (alas x tinggi)}
DEKLARASI

DESKRIPSI
L ← a * t / 2
```

```
Algoritma HITUNG_LUAS_SEGITIGA
{program utama untuk menghitung luas segitiga}
DEKLARASI
    alas , tinggi, Luas : real
    Procedure HIT_LUAS_SEGITIGA(input a , t : real, output L : real)

DESKRIPSI
write('Menghitung Luas Sebuah Segitiga')
read(alas,tinggi)
HIT_LUAS_SEGITIGA(alas,tinggi, Luas)
write('Luas segitiga = ', Luas)
```

```
Procedure HITUNG_TITIKTENGAH(input T1, T2 : Titik , output Tt : Titik)
{menghitung titik tengah dari dua buah titik T1 dan T2}
DEKLARASI

DESKRIPSI
Tt.x ← (T1.x + T2.x) / 2
Tt.y ← (T1.y + T2.y) / 2
```

```
Algoritma TITIKTENGAH
{Program untuk menghitung titik tengah dari dua buah titik T1 dan T2}
DEKLARASI
    type Titik : record < x : real,
                          y : real
                        >
    P1, P2, Pt : Titik
    Procedure HITUNG_TITIKTENGAH(input T1, T2 : Titik , output Tt : Titik)

DESKRIPSI
read(P1.x, P1.y)
read(P2.x, P2.y)
HITUNG_TITIKTENGAH(P1, P2, Pt)
write(Pt.x, Pt.y)
```

3. Parameter masukan/keluaran(*input/output parameter*)→ sebagai masukan dan keluaran bagi prosedur
   Parameter masukan digunakan pada situasi dimana informasi dikirim dari titik pemanggilan→ prosedur
   Parameter keluaran digunakan pada situasi dimana informasi dikirim dari titik pemanggilan → prosedur
   Pada kebanyakan aplikasi, informasi harus dikirim dalam dua arah → Parameter masukan / keluaran

Dengan parameter masukan/keluaran, nama dan nilai parameter aktual dari program pemanggil akan digunakan di seluruh bagian prosedur. Bila parameter aktual diubah nilainya di dalam prosedur, maka sesudah pemanggilan prosedur, niai parameter aktual di titik pemanggilan juga berubah. **Contoh :**

```
Procedure TIGA (input x,y : integer)
{Menambahkan nilai x dengan  dan mengurangi nilai y dengan 2 }

Deklarasi

Deskripsi
x ← x + 2
y ← y – 2
write('Nilai x dan y di akhir Prosedur TIGA : ')
write('x = ',x)
write('y = ',y)
```

```
Algoritma FGH
{Contoh program utama yang memanggil prosedur TIGA}

Deklarasi
    a , b : integer
    Procedure TIGA (input x, y : integer)

Deskripsi
    a ← 15
    b ← 10
    write('Nilai a dan b sebelum pemanggilan : ')
    write('a = ',a)
    write('b = ',b)

    TIGA(a,b)
    write('Nilai a dan b sesudah pemanggilan : ')
    write('a = ',a)
    write('b = ',b)
```

Bila algoritma di atas ditranslasikan ke dalam salah satu bahasa pemrograman, lalu dijalankan, hasilnya :

Nilai a dan b sebelum pemanggilan :

a = 15

b = 10

Nilai x dan y di akhir Prosedur TIGA :

a = 17

b = 8

Nilai a dan b sesudah pemanggilan :

a = 15

b = 10

Procedure TIGA (input/output x,y : integer)
{Menambahkan nilai x dengan dan mengurangi nilai y dengan 2 }

Deklarasi

Deskripsi

```
x ← x + 2
y ← y – 2
write('Nilai x dan y di akhir Prosedur TIGA : ')
write('x = ',x)
write('y = ',y)
```

Algoritma FGH
{Contoh program utama yang memanggil prosedur TIGA}

Deklarasi

```
a , b : integer
Procedure TIGA (input/output x, y : integer)
```

Deskripsi

```
a ← 15
b ← 10
write('Nilai a dan b sebelum pemanggilan : ')
write('a = ',a)
write('b = ',b)

TIGA(a,b)
write('Nilai a dan b sesudah pemanggilan : ')
write('a = ',a)
write('b = ',b)
```

Bila algoritma di atas ditranslasikan ke dalam salah satu bahasa pemrograman, lalu dijalankan, hasilnya :

Nilai a dan b sebelum pemanggilan :

a = 15

b = 10

Nilai x dan y di akhir Prosedur TIGA :

a = 17

b = 8

Nilai a dan b sesudah pemanggilan :

a = 17

b = 8

```
Procedure TUKAR(input/output A, B : integer)
{Mempertukarkan nilai antara A dan B }

Deklarasi
    temp : integer
Deskripsi
    temp ← A
    A ← B
    B ← temp
```

```
Procedure TUKAR_A_B
{Program untuk mempertukarkan nilai antara A dan B }

Deklarasi
    A , B :  integer
    Procedure TUKAR(input/output A, B : integer)

Deskripsi
    read(A , B)
    write(A , B)
    TUKAR(A , B)
    write(A , B)
```

Program dengan Prosedur atau tanpa Prosedur ?

Program yang modular menunjukkan teknik pemrograman yang baik dan terstruktur

Prosedur dengan parameter atau tanpa parameter?

Parameter digunakan sebagai media komunikasi antara prosedur dengan program pemanggil dan dapat mengurangi kebutuhan penggunaan variabel global

Parameter Masukan atau Parameter Keluaran?

Bila prosedur menghasilkan keluaran yang dibutuhkan program pemanggil, gunakan parameter keluaran untuk menampung keluaran tersebut.

Bila prosedur tidak menghasilkan keluaran atau keluarannya hanya digunakan di dalam prosedur, gunakan parameter masukan. Jika prosedur menerima masukan sekaligus keluaran pada parameter yang sama, gunakan parameter masukan/keluaran