Pemrograman Modular

Pengantar

- *Untuk mencapai* suatu tujuan *besar, maka* tujuan tersebut *harus dibagi-bagi menjadi* tujuan *kecil sampai* tujuan *kecil itu merupakan tujuan yang dapat dicapai berdasarkan kondisi dan* potensi *yang dimiiiki saat ini*
- *(Ai-Khuwarizmi)*
- Ucapan Ai-Khuwarizmi di atas sangat mengena daiam kegiatan memprogram.

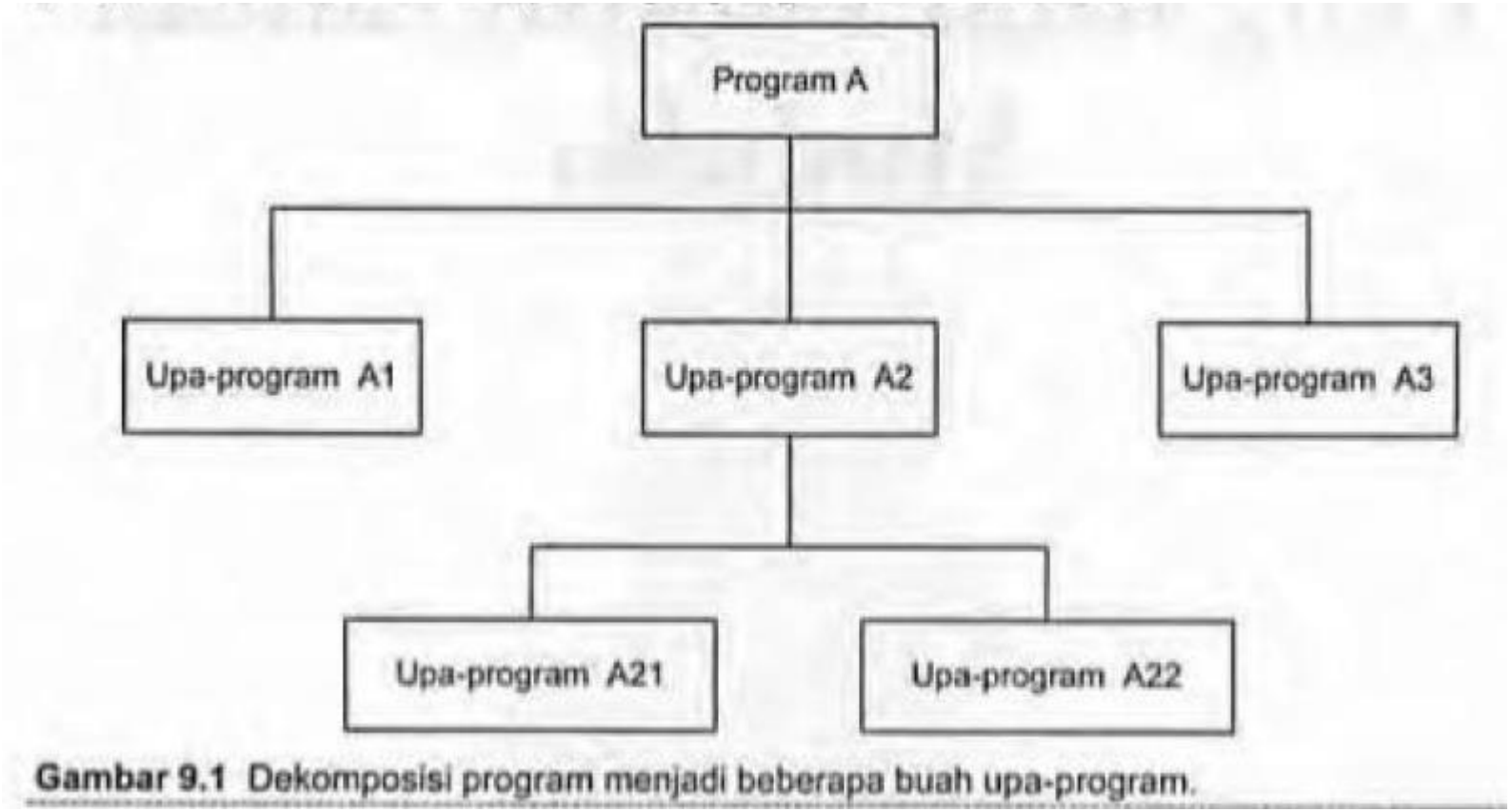
Program – sub program

- Program yang besar lebih sulit dimengerti (dibaca) dan lebih sulit iagi dalam meiakukan pelacakan kesalahan jika ada).
- Oleh karena itu, program sebaiknya dipecah menjadi beberapa upa-program *(subprogram)* yang lebih kecil.
- Setiap upa-program melakukan komputasi yang spesifik.
- Upa-program yang baik adalah upa-program yang independen dari program utama sehingga programnya dapat dirancang tanpa mempertimbangican konteks di mana ia digunakan.
- Dengan kata iain, pemrogram tidak perlu mempermasalahkan *bogaimana* upa-program tersebut dilakukan, tetapi cukup memikirikan *apa* yang ia lakukan

Istilah

- Teknik pemecahan program menjadl sejumlah upa-program dinamakan teknik pemrograman modular *(modular programming).*
- Beberapa bahasa pemrograman menamakan upa-program dengan sebutan sub-rutin *(subroutine),* modul, prosedur, atau fungsi

Contoh Pemrograman Modular



Gambar 9.1 Dekomposisi program menjadi beberapa buah upa-program.

Upa -Program

- masing-masing upa-program mempunyai struktur yang sama dengan program utama: ada judul upa-program, ada bagian deklarasi, dan bagian algoritma.
- Setiap upa-program dapat disimpan di dalam berkas terpisah atau Bersatu dengan berkas program utama

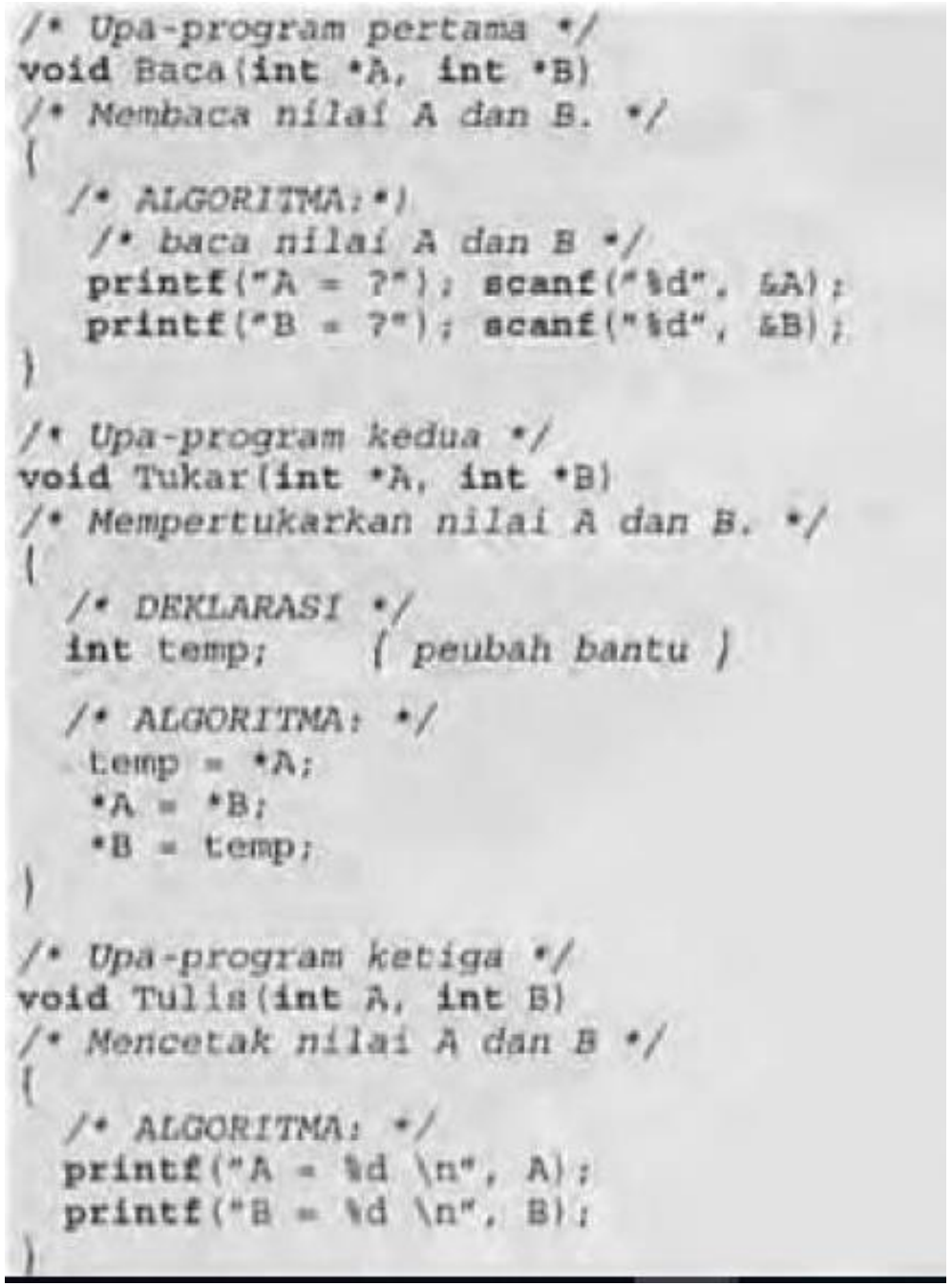
Contoh : Program Pertukaran

- Sebagai ilustrasi permecahan program menjudi sejumlah upa-program, tinjau kembali program pertukaran nilai *A* dan *B* dalam Bahasa C

```
/* PROGRAM Pertukaran */
/* Mempertukarkan nilai A dan B. Nilai A dan B dibaca terlebih dahulu

#include <stdio.h>

main()
{
   /* DEKLARASI */
   int A, B, temp;

   /* ALGORITMA: */

   /* baca nilai A dan B */
   printf("A = ?"); scanf("%d", &A);
   printf("B = ?"); scanf("%d", &B);

   /* proses pertukaran */
   temp = A;
   A = B;
   B = temp;

   /* Tulis nilai A dan B setelah pertukaran */
   printf("A = %d \n", A);
   printf("B = %d \n", B);
}
```

Pemecahan Program

- Program pertukaran di atas dapat kita pecah menjadi tiga buah upa-program yang spesifik, yaltu :
- Upa-program untuk membaca data (A dan B),
- Upa-program untuk melakukan pertukaran nilai A dan B, dan
- Upa-program untuk mencetak nilai A dan B setelah pertukaran.



```
/* Upa-program pertama */
void Baca(int *A, int *B)
/* Membaca nilai A dan B. */
{
  /* ALGORITMA:*)
   /* baca nilai A dan B */
   printf("A = ?"); scanf("%d", &A);
   printf("B = ?"); scanf("%d", &B);
}

/* Upa-program kedua */
void Tukar(int *A, int *B)
/* Mempertukarkan nilai A dan B. */
{
  /* DEKLARASI */
  int temp;     { peubah bantu }

  /* ALGORITMA: */
   temp = *A;
   *A = *B;
   *B = temp;
}

/* Upa-program ketiga */
void Tulis(int A, int B)
/* Mencetak nilai A dan B */
{
  /* ALGORITMA: */
  printf("A = %d \n", A);
  printf("B = %d \n", B);
}
```

Keuntungan Pemrograman Modular

Modularisasi program memberikan dua keuntungan. Pertama, untuk aktivitas yang harus dilakukan lebih dari satu kali, modularisasi menghindari penulisan teks program yang sama secara berulangkali. Di sini, upa-program cukup ditulis sekali saja, lalu upa-program dapat dipanggil dari bagian lain di dalam program. Di sini, penggunaan upa-program dapat mengurangi panjang program. Sebagai ilustrasi, tinjau program dalam bahasa *C* di dalam Algoritma 9.3. Hanya bagian algoritma yang sama dan ditulis berulang-ulang yang ditampilkan di sini (bagian yang diarsir). Bagian algoritma lain cukup dinyatakan dengan "...".

Keuntungan kedua dari modularisasi program adalah kemudahan menulis dan menemukan kesalahan (*debug*) program. Kemudahan menulis akan sangat berguna pada masalah besar yang dikerjakan oleh satu tim pemrogram yang beranggotakan beberapa orang. Masalah yang diprogram dipecah menjadi beberapa masalah yang lebih kecil. Setiap masalah yang lebih kecil tersebut ditulis ke dalam modul yang spesifik dan dikerjakan oleh orang yang berbeda. Satu modul bisa berisi satu atau lebih upa-program. Seluruh modul diintegrasikan menjadi satu buah program yang lengkap. Program yang modular menjadi lebih mudah untuk dibaca dan dimengerti. Program yang tidak modular sulit dipahami, khususnya kalau program tersebut panjang atau terdiri dari puluhan, ratusan, atau ribuan baris instruksi.

Jenis Upa-Program

- Terdapat dua bentuk upa-program :
- pertama **prosedur** (procedure) dan
- Kedua **fungsi** (function).
- Struktur setiap upa-program tersebut pada hakikatnya sama dengan struktur program biasa, yaitu ada bagian judul (header) yang berisi nama modul, bagian deklarasi, dan badan (body) program yang berisi instruksi yang akan dilaksanakan.

Contoh

/*Program Pertukaran */

/* Mempertukarkan nilai A dan B, Nilai A dan B dibaca terlebih dahulu */

#include <stdio.h>

Void Baca (int *A, int *B)

Void Tukar (int *A, int *B)

Void Tulis (int A, int B)

```
main()  /* program utama */
{
   /* DEKLARASI */
   int A, B;
   /* ALGORITMA: */
   Baca(A,B);  /* baca nilai A dan B */
   Tukar(&A,&B); /* proses pertukaran */
   Tulis(A,B); /* Tulis nilai A dan B setelah pertukaran */
}

void Baca(int *A, int *B)
/* Membaca nilai A dan B. */
{
  /* ALGORITMA:*)
   /* baca nilai A dan B */
   printf("A = ?"); scanf("%d", &A);
   printf("B = ?"); scanf("%d", &B);
}

void Tukar(int *A, int *B)
/* Mempertukarkan nilai A dan B. */
{
  /* DEKLARASI */
  int temp;     { peubah bantu }

  /* ALGORITMA: */
   temp = *A;
   *A = *B;
   *B = temp;
}
```

Prosedur & Fungsi

Ilustrasi Prosedur

Prosedur Daftar Ulang

1. **Ambil *Form* Rencana Studi (FRS) di kantor Tata Usaha Akademik dengan memperlihatkan Kartu Tanda Mahasiswa (KTM) dan Kartu Studi Mahasiswa (KSM).**
2. **Lakukan pembayaran SPP di loker pembayaran dan minta kwitansinya.**
3. **Isi FRS dengan mata kuliah – mata kuliah yang akan diambil di semester ini.**
4. **Lakukan perwalian untuk mengesahkan FRS oleh Wali Akademik.**
5. **Jika SPP sudah lunas, maka serahkan FRS yang sudah disahkan oleh Wali Akademik ke petugas di Kantor Tata Usaha Akademik. Jika SPP belum lunas, kembali ke langkah 2.**
6. **Serahkan foto ukuran 2 × 3 untuk KTM yang baru, untuk kemudian di-cap oleh petugas.**
7. **Selesai**

- Ketika sebuah prosedur dieksekusi, maka instruksi-instruksi di dalamnya dikerjakan satu per satu

Prosedur

Di dalam dunia pemrograman, prosedur adalah modul program yang mengerjakan tugas/aktivitas yang spesifik dan menghasilkan suatu efek *netto* [LIE96]. Suatu efek *netto* diketahui dengan membandingkan keadaan awal dan keadaan akhir pada pelaksanaan sebuah prosedur. Oleh karena itu, pada setiap prosedur kita perlu mendefinisikan **keadaan awal** (K.Awal) sebelum rangkaian instruksi di dalam prosedur dilaksanakan dan **keadaan akhir** (K.Akhir) yang diharapkan setelah rangkaian instruksi di dalam prosedur dilaksanakan.

Fungsi

Seperti halnya prosedur, fungsi juga merupakan upa-program yang mempunyai tujuan spesifik. Di dalam Bab 11 ini akan dijelaskan pengertian fungsi, cara mendefinisikan fungsi, cara pemanggilan fungsi, dan translasi fungsi ke dalam bahasa *Pascal* dan bahasa *C*. Pada akhir bab juga dijelaskan konversi prosedur menjadi fungsi dan sebaliknya konversi fungsi menjadi prosedur. Baik fungsi maupun prosedur keduanya merupakan upa-program yang ekivalen, namun pada beberapa masalah adalanya kita lebih tepat menggunakan fungsi ketimbang prosedur, demikian juga sebaliknya.

Fungsi

Fungsi adalah upa-program yang memberikan/mengembalikan (*return*) sebuah nilai dari tipe tertentu (tipe dasar atau tipe bentukan). Definisi fungsi di dalam program bersesuaian dengan definisi fungsi di dalam matematika. Di dalam matematika, kita mengenal cara penulisan fungsi seperti pada contoh berikut:

1. $f(x) = 2x^2 + 5x - 8$
2. $H(x, y) = 3x - y + xy$

Pada kedua contoh di atas, f dan H adalah nama fungsi, sedangkan x dan y adalah *parameter* fungsi yang bersangkutan. Nilai yang diberikan oleh fungsi bergantung pada masukan parameter. Sebagai misal:

1. $x = 2$, maka $f(2) = 2 \cdot 2^2 + 5 \cdot 2 - 8 = 10$
2. $x = 1, y = 2$, maka $H(1, 2) = 3 \cdot 1 - 2 + 1 \cdot 2 = 3$

Prosedur

Pendefinisian Prosedur

- Pendefinisian prosedur artinya menuliskan nama prosedur, mendeklrasikan nama-nama konstanta, peubah dan tipe (jika ada), dan menjabarkan rangkaian aksi yang dilakukan.
- Pada dasamya, struktur prosedur sama dengan struktur algoritma yang sudah Anda kenal, yaitu:
  - Judul *(header)* yang terdiri atas nama prosedur dan deklarasi parameter (jika ada),
  - Deklarasi untuk mengumumkan nama-nama, dan bagian algoritma yang disebut badan prosedur.
  - Setiap prosedur mempunyai narna yang unik
- Nama prosedur sebaiknya diawali dengan kata kerja karena prosedur berisi suatu aktivitas, misalnya HitungLuas, Tukar, CariMaks, Inisialiaasi, AktifkanMenu, dan lain sebagainya.

Parameter

- Parameter adalah nama-nama peubah yang dideklaraskan pada bagian *header* prosedur.
- Sebagian besar program memerlukan pertukaran data/informasi antara prosedur (atau fungsi) dan titik di mana ia dipanggil.
- Penggunaan parameter menawarkan mekanisme pertukaran lnformasi tersebut.
- Tiap *item* data ditransfer antara parameter aktual dan parameter formal yang bersesuaian.
    a. Parameter aktual (kadang-kadang disebut juga argumen) adalah parameter yang dlsertakan pada waktu pemanggilan prosedur
    b. Parameter formal adalah parameter yang dldeklarasikan di dalam bagian *header* prosedur itu sendiri.

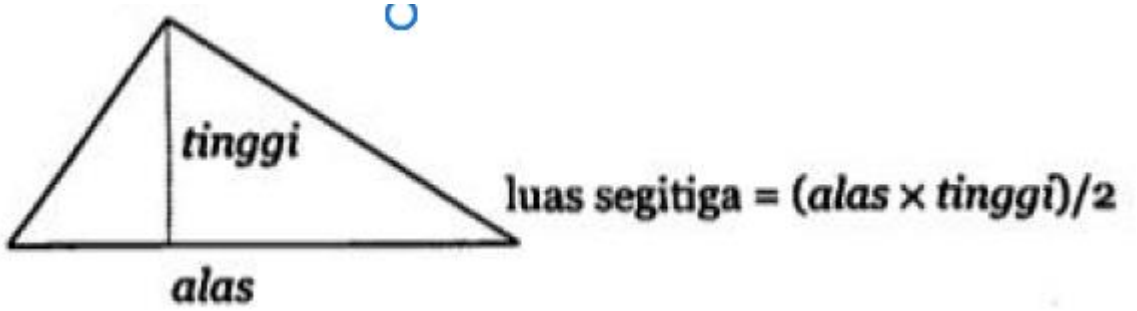
Notasi Algaritmik

- Ketika prosedur dipanggil, parameter aktual menggantikan parameter formal.
- Tiap-tiap parameter actual berpasangan dengan parameter formal yang bersesuaian.
- Notasi algoritmik untuk mendefinisikan prosedur (tanpa parameter) adalah:

```
procedure NamaProsedur (deklarasi parameter, jika ada)
{ Spesifikasi prosedur, berisi penjelasan tentang apa yang dilakukan
oleh prosedur ini.
K.Awal : keadaan sebelum prosedur dilaksanakan.
K.Akhir: keadaan setelah prosedur dilaksanakan. }
DEKLARASI
  { semua nama yang dipakai di dalam prosedur dan hanya berlaku lokal
di dalam prosedur didefinisikan di sini }

ALGORITMA:
    { badan prosedur, berisi urutan instruksi }
```

Algoritma 10.1 Struktur prosedur

- Buatlah prosedur yang membaca panjang alas dan tinggl segitiga, menghitung luas segitiga dengan rumus *luas (alas* x *tinggi/2,* dan mencetak luas segitiga tersebut.



Penyelesaian

```
procedure HitungLuasSegitiga
{ Menghitung luas segitiga dengan rumus L = (alas x tinggi)/2 }
{ K.Awal : sembarang }
{ K.Akhir: luas segitiga tercetak.}

DEKLARASI
  alas    : real   { panjang alas segitiga, dalam cm }
  tinggi  : real   { tinggi segitiga, dalam cm }
  luas    : real   { luas segitiga, dalam cm² }

ALGORITMA:
  read(alas,tinggi)
  luas ← (alas * tinggi)/2
  write(luas)
```

Algoritma 10.2 Prosedur untuk menghitung luas segitiga

- Keadaan awal (K.Awal) prosedur adalah sembarang kondisi
- Namun setelah prosedur selesai dilaksanakan, keadaan akhimya
- (K.Akhir) adalah kondisl di mana luas segitiga tercetak (misalnya ke layar).

Pemanggilan Prosedur

- Prosedur bukan program yang berdiri sendlri, jadi ia tidak dapat dieksekusi secara langsung.
- lni berarti, instruksi-lnstruksi di dalam prosedur hanya dapat dilaksanakan hanya bila prosedur tersebut diakses.
- Prosedur diakses dengan cara memanggil namanya dari program pemanggil
- misalnya dari program utama atau dari modul program lalnnya.
- Jika prosedur tanpa parameter, maka pemanggilannya cukup dengan namanya saja:

Purwarupa Prototype

- Agar nama prosedur dikenal oleh program pemangil, maka di dalam program pemanggil,
- kita harus mendeklaraslkan purwarupa (prototype) prosedur tersebut,
- Purwarupa prosedur hanya berisi bagian header prosedur.
- Pendeklarasian purwarupa juga untuk memberitahu program pemanggil bagaimana cara-cara mengakses prosedur jumlah parameter dan tipe parametemya jika ada.

Contoh Program Luas N-segitiga

- Misalkan kita mempunyai *N* buah segitiga dan kita ingin menghitung luas semua segitiga. Prosedur untuk menghitung luas segitiga sudah kita definisikan:

```
PROGRAM Segitiga
{ Menghitung luas N buah segitiga. }

DEKLARASI
   i, N : integer
   procedure HitungLuasSegitiga
  { Menghitung luas segitiga dengan rumus L = (alas x tinggi)/2 }

ALGORITMA:
   read(N)   { tentukan banyaknya segitiga }
   for i ← 1 to N do
     HitungLuasSegitiga
   endfor
```

Algoritma 10.3 Menghitung luas N buah segitiga.

Nama Global, Nama Lokal, dan Lingkup

- Nama-nama (konstanta, peubah, tipe, dan lain-lain), yang dldeklarasikan di dalam prosedur (termasuk parameter, jika ada) hanya "dikenal" di dalam badan, prosedur yang bersangkutan.
- Nama-nama yang dideklaraslkan di dalam prosedur tersebut dikatakan lingkupnya (scope) ‘lokal’
- Nama-nama lokal hanya berlaku di dalam prosedur yang melingkupinya saja.
- Setelah prosedur selesai dieksekusi nama-nama tersebut tidak dikenal lagi di luar prosedur, alas, tinggi dan luas hanya dapat digunakan di dalam prosedur yang bersangkutan.

Nama Global, Nama Lokal, dan Lingkup

- Sebaliknya, nama-nama (konstanta, peubah, tipe. dan lain-lain) yang dideklarasikan di dalam program utama dikatakan lingkupnya "global".
- Nama-nama global dapat digunakan di bagian manapun di dalam program, baik di dalam program utama maupun di dalam prosedur yang dipanggil
- Tinjau program Segitiga pada Contoh 10.2. Peubah i dan N dapat digunakan di dalam program utama maupun di dalam prosedur HitungLuasSegitiga (kalau diperlukan).

- Sekarang mari kita modifikasi prosedur HitungLuassegitiga dengan
- meniadakan peubah alas, dan tinggi di dalam bagian deklarasinya, dan menghilangkan pernyataan pembacaan data di dalam prosedur.

```
procedure HitungLuasSegitiga
{ Menghitung luas segitiga dengan rumus Luas = (alas x tinggi)/2 }
{ K.Awal : sembarang }
{ K.Akhir: luas segitiga tercetak.}

DEKLARASI
  luas : real
```

```
ALGORITMA:
  luas ← (alas * tinggi)/2
  write(luas)
```

- Selanjutnya, modiflkasi program utama segitiga dengan memasukkan
- Peubah alas dan tinggi ke dalam bagian deklarasi program utama, dan membaca alas dan tinggi

```
PROGRAM Segitiga
{ Menghitung luas N buah segitiga. }

DEKLARASI
   i, N : integer
   alas, tinggi : real
   procedure HitungLuasSegitiga
  { Menghitung luas segitiga dengan rumus L = (alas x tinggi)/2 }

ALGORITMA:
   read(N)   { tentukan banyaknya segitiga }
   for i ← 1 to N do
     read(alas, tinggi)
     HitungLuasSegitiga
   endfor
```

Parameter

- Prosedur dengan parameter diakses dengan cara memanggll namanya dari program pemanggll (program utama atau modul program lain) disertai parametemya.
- Parameter yang disertakan pada waktu pemanggilan disebut parameter aktual. Cara pemanggilan prosedur dengan parameter adalah:
- NamaProsedur(parameter aktual)
- Ketika prosedur dipanggil, parameter aktual berkoresponden dengan parameter formal (parameter yang dideldaarsikan pada bagian *heackr* prosedur). Tup-tiap parameter aktual berpasangan dengan parameter fonnal yang bersesuaian.

Parameter

- Berdasarkan maksud penggunaanya, terdapat tiga jenis parameter formal }l&llg disertakan di dalam prosedur:
- 1. parameter masukan *(input parameter)*
- 2. parameter keluaran *(output* parameter)
- 3. parameter masukanfkeluaran *(input/output* parameter)

Parameter Masukan

- Pada parameter masukan, nilai *(value)* parameter aktual diisikan (assign) ke dalam parameter formal yang bersesuaian. Nilai ini dlgunakan di dalam badan prosedur yang bersangkutan
- Perubahan nilai parameter di dalam badan prosedur tidak mengubah nilai parameter aktual.
- Karena yang dipentingkan adalah nilainya, maka nama parameter actual boleh berbeda dengnn nama parameter formal yang bersesuaian.

Contoh Prosedur

```
procedure HitungLuasSegitiga(input alas, tinggi : real)
{ Menghitung luas segitiga dengan rumus Luas = (alas * tinggi)/2 }
{ K.Awal : alas dan tinggi sudah terdefinisi nilainya }
{ K.Akhir: luas segitiga tercetak.}

DEKLARASI
  luas : integer

ALGORITMA:
  luas ← (alas * tinggi)/2
  write(luas)
```

Algoritma 10.6 Prosedur menghitung luas segitiga dengan parameter masukan

Contoh Main

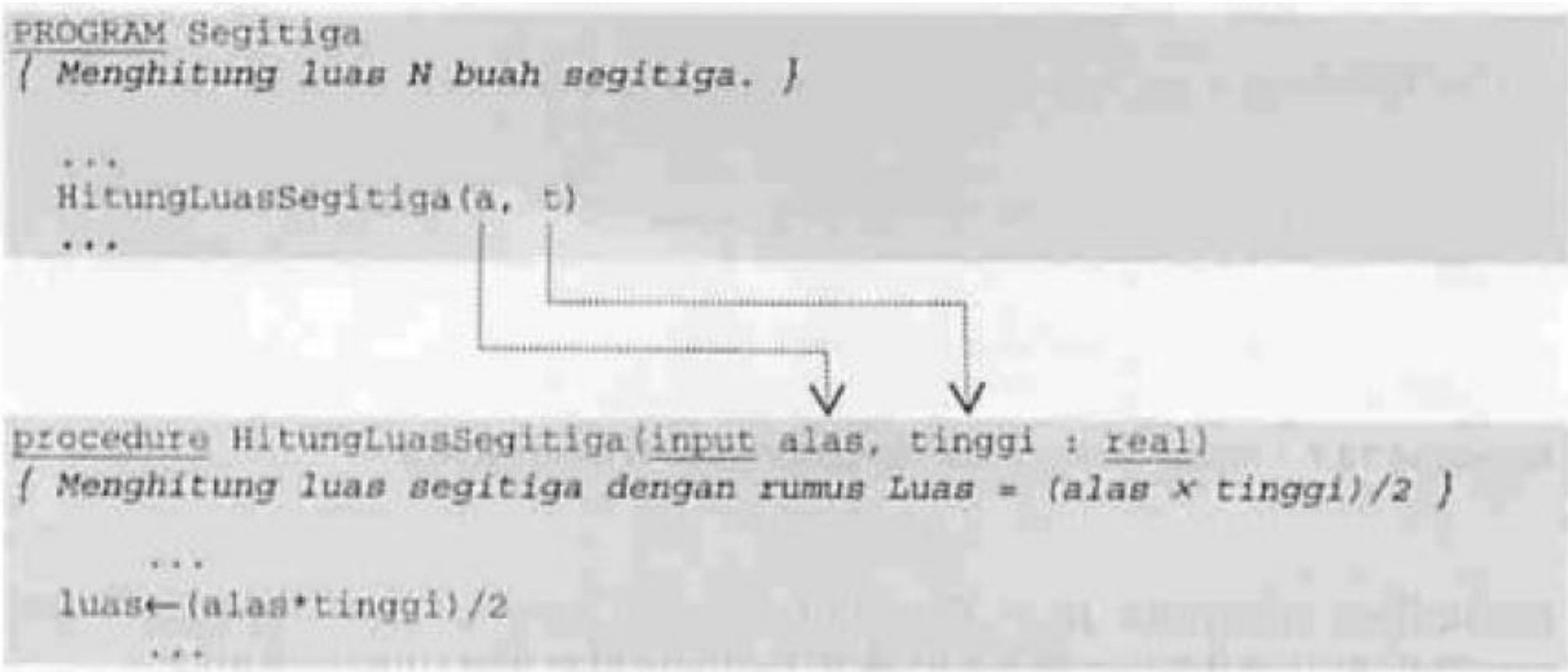
```
PROGRAM Segitiga
{ Menghitung luas N buah segitiga. }

DEKLARASI
   i, N : integer
   alas, tinggi : real
   procedure HitungLuasSegitiga(input alas, tinggi : real)
  { Menghitung luas segitiga dengan rumus L = (alas x tinggi)/2 }

ALGORITMA:
   read(N)   { tentukan banyaknya segitiga }
   for i ← 1 to N do
     read(alas, tinggi)
     HitungLuasSegitiga(alas,tinggi)
   endfor
```

Algoritma 10.7 Program utama untuk menghitung luas N buah segitiga.

- Karena yang dlpentingkan adalah nilainya, maka nama parameter aktual tidak harus sama dengnn nama parameter formal yang bersesuaian asalkan tipenya tetap sama (lihat Algoritma 10.7).

```
PROGRAM Segitiga
{ Menghitung luas N buah segitiga. }

DEKLARASI
   i, N : integer
   a, t : real
   procedure HitungLuasSegitiga(input alas, tinggi : real)
  { Menghitung luas segitiga dengan rumus L = (alas x tinggi)/2 }

ALGORITMA:
   read(N)   { tentukan banyaknya segitiga }
   for i←1 to N do
     read(alas, tinggi)
     HitungLuasSegitiga(a,t)
   Endfor
```

Algoritma 10.8 Program utama untuk menghitung luas N buah segitiga (nama parameter aktual tidak sama dengan nama paramater formal).



Gambar 10.1 Mekanisme korespondensi satu-satu antara parameter aktual dengan parameter formal yang berjenis masukan.

Karena yang dipentingkan adalah nilainya, maka parameter aktual boleh berupa ekspresi atau konstanta. Jadi, pemanggilan berikut:

```
HitungLuasSegitiga(a*0.2, t*0.1)
```

benar (misalnya panjang alas dan tinggi segitiga dikoreksi dengan mengalikannya dengan faktor 0.2 dan 0.1).

Begitu juga pemanggilan

```
HitungLuasSegitiga(12,6)
```

adalah benar.

Parameter Keluaran

- Prosedur mungkin menghasilkan satu atau lebih keluaran yang akan digunakan oleh program pemanggil. Jika ini kasusnya, maka nilai keluaran tersebut ditampung di dalam parameter keluaran.
- Ketika prosedur yang mengandung parameter keluaran dipanggil, maka nama parameter actual menggantikan *(substitute)* nama parameter formal yang bersesuaian di dalam prosedur.
- Selanjutnya, nama parameter aktual akan digunakan selama pelaksanaan prosedur (ini berlawanan dengan parameter masukan, yang dalam hal ini nilai dari parameter aktual yang di-assign ke dalam parameter formal).
- Karena nama parameter merupakan suatu lokasi di memori, maka bila di dalam prosedur parameter aktual diisi suatu nilai, nilai ini akan tetap berada di dalam parameter aktual meskipun prosedur selesal dllaksanakan.
- Jadi, setelah pemanggilan, parameter aktual berisi suatu nilai yang merupakan keluaran dari prosedur tersebut.

Parameter Keluaran

- Parameter keluaran dideklarasikan di dalam *header* prosedur, sebagaimana parameter masukan. Tetapi, parameter keluaran harus dideklaraslkan dengan kata kunci output.
- Tinjau prosedur HitungLuassegitiga di dalam Algoritma 10.5.
- Luas segitiga disimpan di dalam peubah luas dan nilainya dicetak di dalam prosedur tersebut.
- Andaikan kita menginginkan luas segitiga dicetak di dalam program pemanggil (jadi, bukan dlm dalam prosedur), maka kita harus menyatakan luas sebagai parameter keluaran, seperti yang ditullskan di dalam Algorltma 10.11 berikut ini:

Contoh Parameter Output

```
procedure HitungLuasSegitiga(input alas, tinggi : real,
                             output luas : real)
{ Menghitung luas segitiga dengan rumus Luas = (alas x tinggi)/2 }
{ K.Awal : alas dan tinggi sudah terdefinisi nilainya }
{ K.Akhir: luas berisi luas segitiga. }

DEKLARASI
   { tidak ada }
ALGORITMA:
   luas ← (alas * tinggi)/2
```

Algoritma 10.11 Prosedur menghitung luas segitiga dengan luas sebagai parameter keluaran

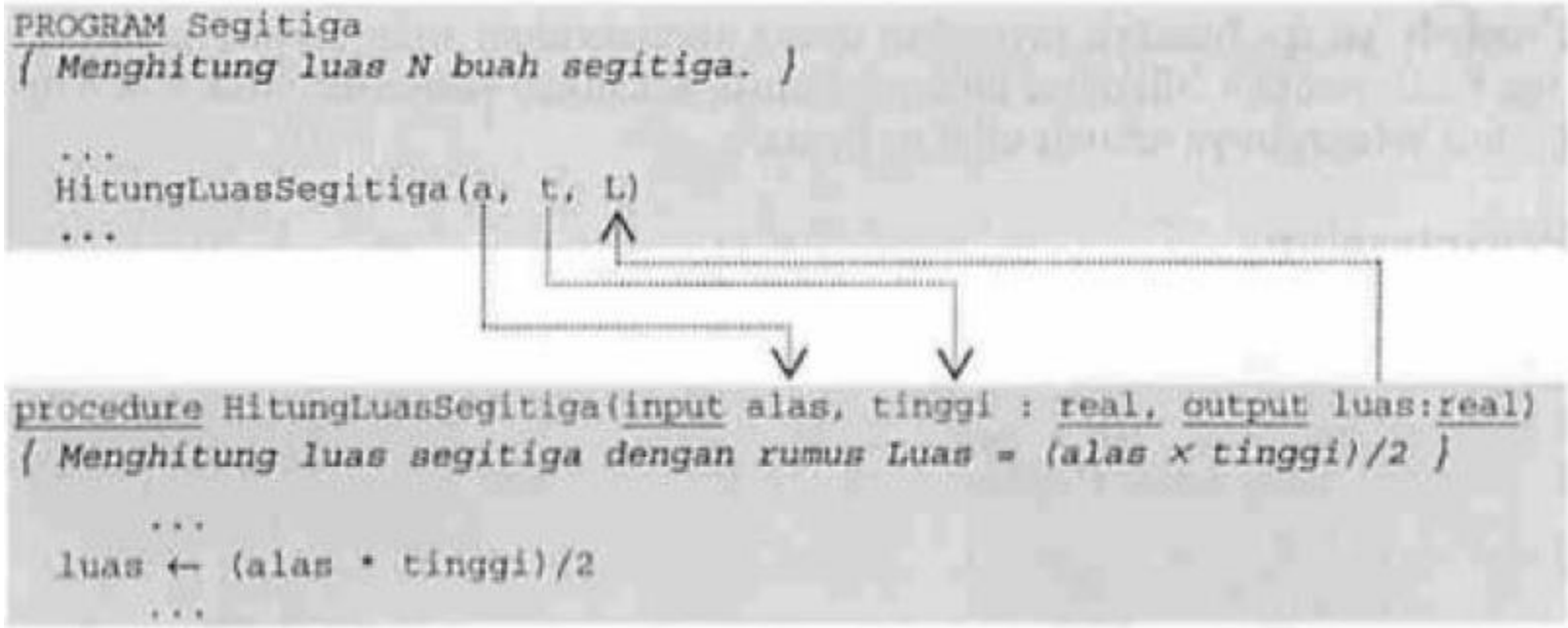
Contoh Parameter Output

- Kata kunci lnput sebelum alas dan tinggi pada bagian *header* menyatakan bahwa kedua parameter tersebut parameter masukan, dan kata kunci output sebelum luas menyatakan bahwa luas adalah parameter keluaran.
- Keadaan awal (K.Awal) prosedur adalah kondisi di mana alas dan tinggal sudah terdeflnisi nilainya.
- lni berarti bahwa alas dan tinggi sudah harus sudah berisi nilai sebelum pelaksanan prosedur.
- Keadaan akhir (K.Akhir) prosedur adalah kondisi dimana luas berisi luas segitiga. Program utama yang memanggil prosedur HitungLuasSegitiga
- harus mendeklarasikan prosedur ini dan memunggilnya dengan parameter aktual yang bersesuaian, Keluaran prosedur, yaitu Luas, dicetak dalam program utama:

```
PROGRAM Segitiga
{ Menghitung luas N buah segitiga. }

DEKLARASI
    i, N : integer
    a, t, L : real    { alas, tinggi, dan luas segitiga}
    procedure HitungLuasSegitiga(input alas, tinggi : real,
                                 output luas : real)
  { Menghitung luas segitiga dengan rumus L = (alas x tinggi)/2 }

ALGORITMA:
    read(N)   { tentukan banyaknya segitiga }
    for i ← 1 to N do
      read(a, t)
      HitungLuasSegitiga(a, t, L)
      write(L)
    endfor
```

Algoritma 10.12 Program utama untuk menghitung luas N buah segitiga.

- Ketika prosedur HitungLuasSegitiga dipanggil, maka nilai parameter
- aktual a dan t diisikan ke dalam parameter formal alas dan tinggi (lihat Gamhnr 10.2), sedangkan nama parameter aktual L menggantikan nama parameter formal luas (lihm Gambnr 10.3).



Hal lain yang harus diperhatikan pada jenis parameter keluaran ini adalah parameter aktual harus berupa peubah, tidak boleh berupa konstanta atau ekspresi. Jadi, pemanggilan seperti contoh di bawah ini adalah SALAH:

```
HitungLuasSegitigaIGA(a,t,10)      { karena 10 adalah konstanta }
HitungLuasSegitigaIGA(a,t,10*a)    { karena 10*a adalah ekspresi }
```

Contoh 10.3. Buatlah prosedur untuk menghitung nilai rata-rata *N* buah data bilangan bulat. Data bilangan bulat dibaca dari piranti masukan. Masukan prosedur adalah *N* dan keluarannya adalah nilai rata-rata.

Prosedur menghitung nilai rata-rata:

```
procedure HitungRataRata(input N : integer, output u : real)
{ Menghitung rata-rata N buah bilangan bulat yang dibaca dari piranti
  masukan. }
{  K.Awal : N sudah berisi banyaknya bilangan bulat, N > 0. }
{  K.Akhir: u berisi rata-rata seluruh bilangan. }

DEKLARASI
   x : integer    { data bilangan bulat yang dibaca dari papan kunci }
   k : integer    { pencacah banyak bilangan }
   jumlah : integer    { jumlah seluruh bilangan }
ALGORITMA:
   jumlah ← 0    { inisialisasi }
   for k ← 1 to N do
     read(x)
     jumlah ← jumlah + x
   endwhile

   u ← jumlah/N
```

Algoritma 10.13 Menghitung nilai rata-rata .

Contoh 10.4. Buatlah prosedur untuk menentukan nilai terbesar antara dua buah peubah bilangan bulat, *A* dan *B*. Masukan prosedur adalah *A* dan *B*, dan keluarannya adalah nilai terbesar.

Penyelesaian

```
procedure TentukanMaksimum(input A, B : integer, output maks :
integer)
{ Menentukan nilai terbesar dari dua buah peubah, A dan B. }
{ K.Awal : Nilai A dan B sudah terdefinisi. }
{ K.Akhir: maks berisi nilai terbesar antara A dan B. }

DEKLARASI
    { tidak ada }

ALGORITMA:
  if A > B then
    maks ← A
  else
    maks ← B
  endif
```

Algoritma 10.14 Menenetukan nilai terbesar dari dua buah data.

Parameter Masukan/Keluaran

- Ketika prosedur yang mengandung parameter keluaran dipanggil, nama parameter aktual di dalam program pemanggil menggantikan *(substitute)* nama parameter formal yang bersesuaian di dalam prosedur.
- Selain ltu, isi atau nilai parameter aktual juga ikut disalin ke dalam parameter formal. Jadi, nama dan nilai parameter aktual digunakan di seluruh bagian prosedur

Parameter Masukan/Keluaran

- Akibat penggunaan parameter masukan/keluaran, bila parameter aktual diubah nilainya di dalam badan prosedur, maka sesudah pemanggilan prosedur nilai parameter aktual di titik pemanggilan juga berubah.
- Ini berbeda dengan parameter masukan, yang dalam hal ini meskipun nilai parameter aktual di dalam badan prosedur diubah, nilai parameter actual tersebut tidak berubah di titik pemanggilan.
- Parameter masukan/keluaran dideklarasikan *di* dalam *header* prosedur dengan kata kunci input/ output.
- parameter aktual harus berupa peubah, tidak boleh berupa konstanta utau ekspresi.

Contoh 10.5. Di dalam bahasa *Pascal* terdapat procedur `Inc(x)` yang berkoresponden dengan `x:=x+1`. Tulislah prosedur `Inc` dalam notasi algoritmik.

Penyelesaian

Peubah `x` harus dinyatakan sebagai parameter masukan/keluaran, sebab prosedur `Inc` harus menerima nilai `x` dan menghasilkan nilai `x` yang baru setelah ditambah satu.

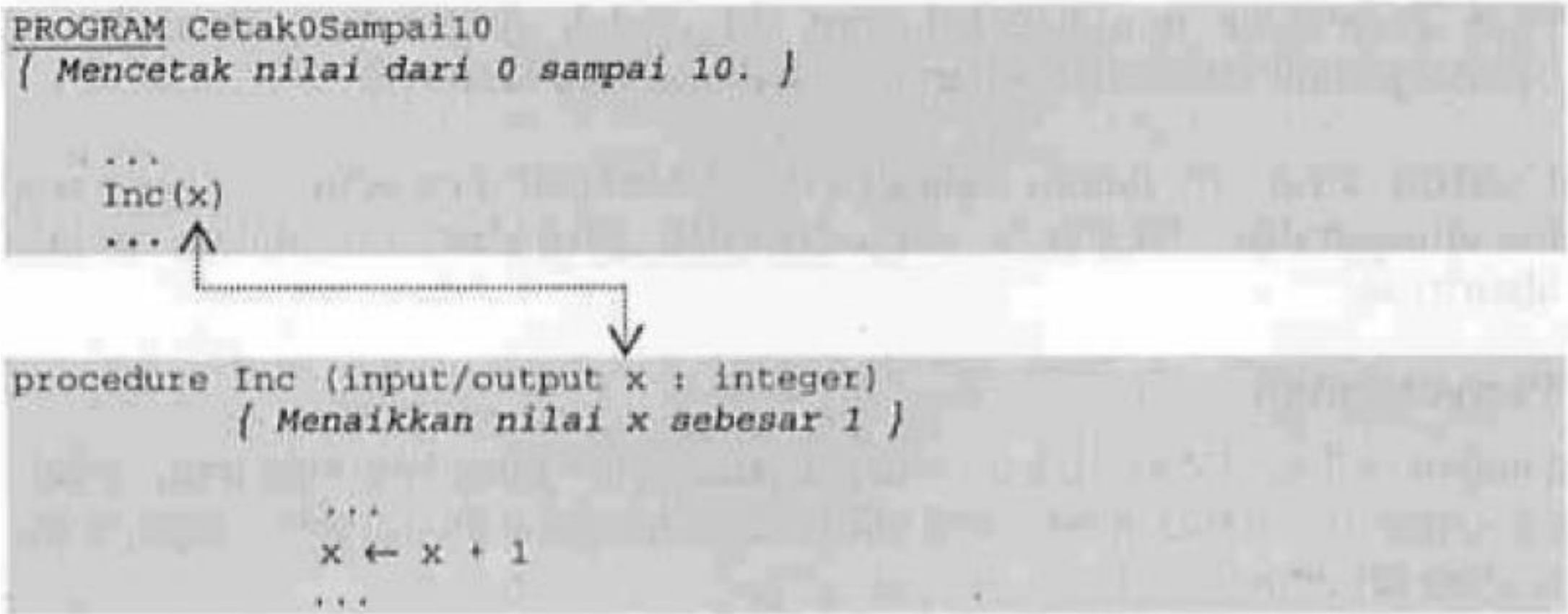
```
procedure Inc(input/output x : integer)
{ Menaikkan nilai x sebesar 1. }
{ K.Awal : x sudah terdefinisi nilainya.  }
{ K.Akhir: nilai x bertambah 1. }

DEKLARASI

ALGORITMA:
   x ← x + 1
```

Algoritma 10.15 Menaikkan nilai peubah sebesar 1.

Contoh program yang memanggil prosedur `Inc` ditunjukkan pada Algoritma 10.14. Prosedur mencetak nilai-nilai `x` dari 0 sampai 10.

```
PROGRAM CetakOSampai10
{ Mencetak nilai dari 0 sampai 10. }

DEKLARASI
   x : integer
   procedure Inc(input/output x : integer)
   { Menaikkan nilai x sebesar 1. }

ALGORITMA:
   x ← 0
   repeat
     write(x)
     Inc(x)
   until x > 10
```

Algoritma 10.16 Program utama untuk mencetak 0 sampai 10.



Gambar 10.3 Mekanisme korespondensi satu-satu antara parameter actual dengan parameter formal yang berjenis masukan/keluaran.

Contoh Parameter Input

```
#include <stdio.h>
#include <stdlib.h>

int main(){
        //deklarasi
        void TambahEmpat(int x, int y);
        int X,Y;
        X=9;
        Y=15;
        //algoritma
        TambahEmpat(X,Y);
        printf("%d\n",X);
        printf("%d\n",Y);
}
```

```
void TambahEmpat(int x, int y){
      //deklarasi
      x=x+4;
      y=y+4;
      printf("%d\n",x);
      printf("%d\n",y);
}
```

Perbandingan Parameter Input/Output/ I/O

Contoh Parameter Input/Output/ I/O

```
int main(){
        //deklarasi
        int X,Y;
        X=8;
        Y=10;
        void tambah4(int X, int Y);
        //algoritma
        tambah4(X,Y);
        printf("setelah pemanggilan\n");
        printf("%d\n",X);
        printf("%d\n",Y);

}

void tambah4(int X, int Y){
        X=X+4;
        Y=Y+4;
        printf("%d\n",X);
        printf("%d\n",Y);
}
```

Contoh Parameter Input/Output

```
#include <stdio.h>
#include <stdlib.h>

int main(){
		//deklarasi
		int X,Y;
		X=8;
		Y=10;
		void tambah4(int *X, int *Y);
		//algoritma
		tambah4(&X,&Y);
		printf("setelah pemanggilan\n");
		printf("%d\n",X);
		printf("%d\n",Y);

}
```

```
void tambah4(int *X, int *Y){
		*X=*X+4;
		*Y=*Y+4;
		printf("%d\n",*X);
		printf("%d\n",*Y);
}
```

Contoh Parameter I/O dan Output

```
int main(){
        //deklarasi
        int X,Y,tot;
        X=8;
        Y=10;
        void tambah4(int *X, int *Y, int *tot);
        //algoritma
        tambah4(&X,&Y,&tot);
        printf("setelah pemanggilan\n");
        printf("%d\n",X);
        printf("%d\n",Y);
        printf("%d\n",tot);

}
```

```
void tambah4(int *X, int *Y, int *tot){
        *X=*X+4;
        *Y=*Y+4;
        *tot=*X+*Y;
        printf("%d\n",*X);
        printf("%d\n",*Y);
        printf("%d\n",*tot);
}
```

Latihan

1. Menghitung nilai rata-rata dari N bilangan dimana bilangan dibaca dari input sebanyak N. Program utama akan menghitung nilai rata-rata dan mengidentifikasi jika rata-rata kurang dr 50 maka hasil jelek, jika lebih sama dengan 50 maka hasil baik
2. Soal :

Tulislah prosedur untuk menghitung jumlah N buah bilangan genap pertama (bilangan genap dimulai dari 0). Prosedur menerima (parameter) masukan N dan memberikan (parameter) keluaran jumlah N buah bilangan genap pertama.

1. Soal :

   Tulislah prosedur yang menghasilkan nilai rata-rata sekumpulan data bilangan bulat yang dibaca secara berulang-ulang dari papan ketik (akhir pembacaan adalah 9999). Prosedur memiliki parameter keluaran, yaitu nilai rata-rata yang dihasilkan.

2. Soal :

Contoh Parameter Output

```
#include "stdio.h"

void HitungLuas(int P, int L, int *Luas){
        *Luas = P*L;
}

void TampilLuas(int Luas){
        printf("\nLuas adalah %d",Luas);
}
```

```
int main(){
        //deklarasi
        void HitungLuas(int P, int L, int *Luas);
        void TampilLuas(int Luas);
        int pjg, lbr, LUAS;
        //Algoritma
        pjg=10;
        lbr=5;
        HitungLuas(pjg, lbr, &LUAS);
        TampilLuas(LUAS);//50
        //luas 2 kali lipat
        LUAS=LUAS*2;
        printf("\nLuas kali 2 = %d",LUAS);//100
}
```

Contoh parameter I/O

```
#include <stdio.h>

void Input(int *A, int *B){
        scanf("%d", A);
        scanf("%d", B);

void Tukar(int *A, int *B){
        int C;
        C=*A; //C=100
        *A=*B; //A=200
        *B=C;//B=10
void TampilTukar(int A, int B){
        printf("\nNilai A baru = %d",A);
        printf("\nNilai B baru = %d",B);
}
```

```
int main(){
        //deklarasi
        void Input(int *A, int *B);
        void Tukar(int *A, int *B);
        void TampilTukar(int A, int B)
        //parameter actual
        int a, b;
        //algoritm
        Input(&a,&b);
        Tukar(&a,&b);
        TampilTukar(a,b);//a=200 b=100
        a=a*10;
        b=b*10;
        printf("\nNilai a baru = %d",a);
        printf("\nNilai b baru = %d",b);
        printf("\nNilai b baru = %d",a*b);
}
```