



# Pemrograman Prosedural

# PENDAHULUAN

Tim Pengajar KU1071

Sem. 1 2008-2009



# Tujuan Kuliah

## Pemrograman Prosedural

- Mahasiswa mampu:
  - Memecahkan masalah dengan paradigma prosedural dan menuliskan spesifikasi dan algoritmanya tanpa tergantung bahasa pemrograman
  - Menulis algoritma dengan metodologi dan skema standard yang diajarkan
  - Menuliskan program yang “baik”



# Program Prosedural (1)

- Program dalam bahasa Pascal termasuk dalam program prosedural : Algoritma + Struktur Data
- Pemrograman prosedural (imperatif) :
  - Dihasilkan berdasarkan dekomposisi “aksional”, menjadi Aksi yang akan dijalankan secara berurutan (sekuensial).

# Program Prosedural (2)

- Pemrograman prosedural (imperatif) :
  - Aksi :
    - Jelas Initial state, Final state dan harus dalam waktu terbatas
    - Dapat didekomposisi menjadi Sub Aksi
  - Aksi diterjemahkan menjadi sederetan instruksi (aksi primitif) yang dapat dijalankan oleh mesin
- Ilustrasi: **MENGUPAS KENTANG** (diktat hal. 17-25)

# Aksi Mengupas Kentang (1)

Initial State (I.S.):  $T_0$ , kentang di kantong, ada di rak dapur

Final State (F.S.):  $T_1$ , kentang terkupas di panci, siap dimasak, kantong kembali ke rak dapur

Sub-aksi:

1. Ambil kantong kentang dari rak
2. Ambil panci dari lemari
3. Kupas kentang
4. Kembalikan kantong kentang ke rak

Aksi 1 dan 2 bebas urutan, bisa digabungkan



# Aksi Mengupas Kentang (2)

1. Ambil kantong kentang dari rak
  2. Ambil panci dari lemari
  3. Kupas kentang
  4. Kembalikan kantong kentang ke rak
- Harus dikerjakan dengan urutan yang diinginkan → aksi sekuensial
  - Saat tertentu (jika baju warna muda) perlu celemek → aksi kondisional/ analisis kasus
  - Mengupas kentang dilakukan hingga pada kriteria tertentu (jumlah) → pengulangan
  - Mengupas kentang bagian dari aksi menyiapkan makan malam → subprogram

# Notasi Algoritmik

- Teks Algoritma (diktat hal. 28-31)
    - Judul (Header)
    - Kamus
    - Algoritma
- Harus didefinisikan  
nama sebagai  
identifikasi
- Nama: sesuatu yang dipakai untuk identifikasi type, tempat penyimpanan (variable), konstanta, prosedur, fungsi, atau modul program, algoritma → harus unik

# Notasi Algoritmik

- Template Program

**JUDUL**

{Spesifikasi teks algoritmik secara umum}

**KAMUS**

{Definisi konstanta, type, deklarasi variabel, spesifikasi prosedur, fungsi}

**ALGORITMA**

{Teks algoritma - tidak berada di antara tanda kurung kurawal}



# Kamus

- Kamus dipakai untuk deklarasi
- To 'declare' = to make known formally, officially, or explicitly
- Deklarasi nama yang didefinisikan pemrogram : "type", variabel, konstanta
- Deklarasi nama-nama lain (biasanya tidak harus): nama fungsi, prosedur dan spesifikasinya
- Deklarasi BUKAN instruksi
- Contoh deklarasi:

i	: <u>integer</u>
ff	: <u>real</u>
S	: string
P	: Point

# Algoritma

- Adalah bagian program dalam bentuk teks algoritmik yang berisi instruksi atau pemanggilan aksi
- Teks algoritmik tsb. dapat berupa:
  - Instruksi dasar: I/O, assignment
  - *Sequential statement*
  - Analisis kasus
  - pengulangan



# Type

- Definisi Type (pola struktur informasi):
  - nama type
  - domain nilai, batasan nilai yang dapat disimpan dalam type tsb. (kelak, ketika menjadi variabel)
  - perjanjian cara menuliskan konstanta (literal) type tsb
  - operator
- Type:
  - primitif
  - bukan primitif, buatan pemrogram, harus dideklarasikan

# Type

- Type dasar, primitif disediakan oleh bahasa
- Type enumerasi
- Type koleksi, misal: array (akan dijelaskan kemudian)
- Type bentukan: nama type dibuat dan didefinisikan oleh pemrogram.



# Type Dasar

- Disediakan oleh bahasa
- Lihat diktat hal 32-35
  - Boolean
  - Integer
  - Real
  - Character
  - String

# Type Enumerasi

- Type yang definisi domainnya tidak dilakukan menurut suatu aturan (by definition), melainkan dengan menyebutkan satu per satu nilai anggotanya (meng-"*enumerate*")

type T : (e1, e2, e3, ..)

tt : T {tt adalah var bertipe T}

- Seringkali tersedia dalam bahasa tingkat tinggi untuk mendefinisikan dengan jelas suatu himpunan yang sudah pasti
- Cara akses suatu nilai

First(tt), Last(tt), Succ(e1), Prec(e2)

# Contoh Type Enumerasi

- Type Hari

```
type Hari : (senin, selasa, rabu,  
             kamis, jumat, sabtu, minggu)
```

- Deklarasi nama variable H ber-type Hari

```
H: Hari
```

- Cara akses:

```
First(H)      {menghasilkan nilai: senin}
```

```
Last(H)       {menghasilkan nilai: minggu}
```

```
Succ(jumat)   {menghasilkan nilai: sabtu}
```

```
Prec(jumat)   {menghasilkan nilai: kamis}
```

- Konstanta

```
senin, sabtu, minggu, dll
```

# Type Bentukan

- Type yang dibentuk (dan diberi nama) dari beberapa komponen ber-type tertentu
- Disebut juga type komposisi, agregat
- Mengapa perlu type bentukan?
  - Perancang memutuskan bahwa keseluruhan (hasil komposisi) komponen type mempunyai makna **semantik**
- Operasi terhadap type bentukan:
  - Terhadap komponen bertipe dasar: seperti tipe dasar
  - Terhadap keseluruhan: bisa didefinisikan, bisa tidak



# Type Bentukan

```
type namatype: < elemen1 : type1,  
                  elemen2 : type2,  
                  ... >
```

- Membentuk type berarti menentukan struktur data
- Pembentuk keseluruhan: Konstruktor
- Pengacu elemen: Selektor

# Contoh: Type Point

- Type Point

```
type Point: < X : real,  
              Y : real  
              >
```

- Deklarasi nama variable P

```
P: Point
```

- Cara acu/akses: P.X, P.Y

- Domain: <real, real>


- Konstanta: <1.0, 1.0>, <10.0, 5.0>

- Operator:

- Operasi terhadap Point harus dibuat
- operasi real terhadap P.X, P.Y



Lihat  
contoh lain  
pada  
Diktat hal.  
37-39



Lihat contoh  
pada Diktat  
hal. 51-53,  
dan kerjakan  
latihan soal  
(no3) hal 53



# Kapan harus membuat type?

- Untuk mempermudah berpikir dalam abstraksi lebih tinggi (tidak pada tingkatan primitif).
- Kalau data yang diolah dapat diabstraksikan menjadi suatu data yang merupakan kesatuan, misalnya: Point, Jam, Date, Orang, Pegawai, SIM, Kartu, Window, Figure, Square, .....  
Dengan memberi nama Type, pengolahan jadi lebih jelas.
- Biasanya, nama type mewakili nama 'benda' dengan ciri tertentu.

# Variabel

- Variabel menyimpan nilai ber-"type" sesuai dengan deklarasi, yang dapat dioperasikan sesuai operator
- Variabel :
  - deklarasi (supaya nama dikenal),
  - alokasi (supaya punya memori),
  - inisialisasi nilai (siap dimanipulasi)
- Contoh
  - Deklarasi (dan alokasi):  $i : \underline{\text{integer}}$
  - Inisialisasi:  $i \leftarrow 0$

Disebut *assignment* → akan dijelaskan kemudian

# Variabel: Scope & Life time



- Scope : lingkup
  - lingkup dimana nama variabel berlaku/dikenal.
- Lifetime : masa hidup
  - kapan terdefinisi memori & nilainya
  - kapan dihancurkan/tidak ada lagi
- Dalam mendeklarasi variabel, harus memahami betul mengenai scope dan life time, serta pemusnahannya.



# Membuat Variabel yang “baik”

- Nama sesuai dengan artinya, tidak membingungkan.
- Memakai nama yang tidak berarti sesuai ‘konvensi’, misalnya:
  - $i, j, k$  untuk indeks array, matriks
- Dirancang sesuai scope & life time yang diperlukan.

# Konstanta

- Nilai, literal yang diberi nama
- Berbeda dengan Variable, tidak boleh diubah nilainya.
- Menambah robustness, *readability* (bandingkan dengan program yang mengandung literal di mana-mana)
- Contoh:

```
constant PI : real = 3.14159
```

# Memakai konstanta

- Memang untuk nilai yang *constant*
- Nama sesuai dengan nilainya, tidak membingungkan. Misalnya

constant Satu: integer = 1

bukan

constant Satu: integer = 7

- Sebagai 'parameter' program.



# Perintah paling dasar

- Pemberian nilai (assignment) sesuai dengan type.
- Perbandingan (kesamaan, ketidak-samaan)
- Operasi relasional lain (lebih besar, lebih kecil,.....)
- Operasi aritmetika (khusus untuk nilai numerik)

# Nilai, Input+Output

- Nilai atau harga: suatu besaran bertipe yang telah dikenal
- Pengisian nilai:
  - Assignment
  - Dibaca dari piranti masukan
- Lihat contoh notasi untuk assignment, input, output (diktat hal 41-42)

# Ekspresi

- Ekspresi:
  - “rumus perhitungan”
  - Terdiri atas operator dan operan
- Notasi yang dipakai: infix
  - Contoh:  $27 * 5 - 2$
- Jenis Ekspresi: logika/boolean, numerik, karakter, dan string (diktat hal 43-45)



# Ekspresi

- Ekspresi :
  - ketat terhadap type
  - “loose” terhadap type
- Usahakan menulis ekspresi ketat type

# Assignment (←)

- Ruas kiri ← Ruas Kanan
- Ruas kiri harus variable
- Ruas kanan harus <ekspresi>
- Ekspresi :
  - operan (nama variabel, konstanta, aplikasi fungsi) dan operator harus kompatibel, ketat type.
  - ekspresi bukan hanya ekspresi aritmetika, ada ekspresi boolean, ekspresi relasional

# Aksi Sekuensial

- Adalah struktur kontrol algoritmik paling sederhana
- Dilaksanakan oleh komputer berdasarkan urutan penulisannya
- Initial State vs Final State. Final State sebuah aksi ke- $i$  menjadi Initial State bagi aksi selanjutnya, aksi ke  $i+1$
- Perhatikan: Program SEQ1, SEQ2, SEQ3, Contoh 3: Jarak (diktat hal. 46-50)



# Translasi Algoritmik ke Pascal





# Translasi Notasi Algoritmik → Pascal

Item Perbandingan	Algoritmik	Pascal
Judul	<b>program</b> <judul> {Spesifikasi}	program <judul>; (*Spesifikasi*)
Kamus	<b>Kamus</b> ...	(* Kamus *) ...
Algoritma	<b>Algoritma</b> ...	(* Algoritma *) begin ... end.

# Translasi Notasi: Algoritmik → Pascal



Item Perbandingan	Algoritmik	Pascal
Variable	<code>i : <u>integer</u></code> <code>Found : <u>boolean</u></code>	<code>var</code> <code>    i : integer;</code> <code>    Found : boolean;</code>
Type	<code><u>type</u> Point:</code> <code>&lt;X : <u>real</u>; Y : <u>real</u>&gt;</code>	<code>type</code> <code>    Point = record</code> <code>        X : real; Y : real;</code> <code>    end;</code>
Konstanta	<code><u>constant</u> PI : <u>real</u> = 3.14</code> <code><u>constant</u> Satu : <u>integer</u> = 1</code>	<code>const</code> <code>    PI = 3.14;</code> <code>    Satu = 1;</code>
Input	<code><u>input</u> (x)</code>	<code>read(x); readln(x);</code>
Output	<code><u>output</u> (x)</code>	<code>write(x); writeln(x);</code>
Assignment	<code>i ← 0</code>	<code>i := 0;</code>
Komentar	<code>{Ini adalah komentar}</code>	<code>(*Ini adalah komentar*)</code> <b><u>atau</u></b> <code>{Ini adalah komentar}</code>



# Pengantar Program Pascal

- Pola

```
program <nama_program>;
(*header, deskripsi program*)
(*Kamus*)
const (*optional*)
type (*optional*)
var (*optional*)
(*Algoritma*)
begin
    (* badan program *)
end.
```

# Contoh Translasi Notasi



```
{ Program dengan notasi
algoritmik }
```

## **Program** HitungVolBola

```
{ Program menghitung volume
bola }
```

## **KAMUS**

```
constant PI: real = 3.14
v : real { vol. bola }
r : real { jari2 bola }
```

## **ALGORITMA**

```
input (r)
v ← (4.0/3.0)*PI*r*r*r
output(v)
```

```
(* Program dengan Bahasa Pascal
*)
```

```
Program HitungVolBola;
(* Program menghitung volume
bola *)
```

```
(* KAMUS *)
```

```
const
    PI = 3.14;
var
    v: real; (*vol. bola*)
    r: real; (*jari-jari bola*)
```

```
(* ALGORITMA *)
```

```
begin
    readln(r);
    v := (4.0/3.0)*PI*r*r*r;
    writeln(v);
end.
```

# Latihan Soal

- Buatlah program kecil dalam notasi algoritmik untuk persoalan-persoalan berikut:
  - Menghitung volume gas ideal ( $V$ ) dalam liter dengan masukan tekanan ( $P$ ) dalam *kiloPascal*, banyaknya mol gas ( $n$ ) dalam *mol*, dan temperatur ( $T$ ) dalam *derajat Kelvin*, serta diketahui konstanta gas ideal ( $R$ ) yaitu  $8.314 \text{ JK}^{-1}\text{mol}^{-1}$  dengan rumus:  $P V = n R T$
  - Menghitung tegangan listrik ( $V$ ) dalam volt dari masukan besarnya arus listrik ( $I$ ) dalam ampere dan hambatan listrik ( $R$ ) dalam ohm, dengan hukum Ohm:  $V = I R$
  - Menghitung luas sebuah trapesium ( $L$ ) berdasarkan masukan  $a$  dan  $b$  yang merupakan panjang dua sisi sejajar trapesium dan  $h$  yang merupakan tinggi trapesium dengan rumus:  $L = \frac{1}{2} * h * (a+b)$



# Program kecil pascal

- Lihat diktat “Contoh Program Kecil dalam Bahasa Pascal” (hal 1-10)
  - Latihan: Buat notasi algoritmik dari program tersebut