Sub Program : Prosedur

Tim Pengajar KU1071 – PTI A

Semester I 2009/2010

Tujuan Perkuliahan

- Mahasiswa memahami makna dan kegunaan prosedur sebagai salah satu sub program
- Mahasiswa dapat menggunakan notasi prosedur dengan benar
- Mahasiswa dapat membuat program dengan menggunakan prosedur
- Mahasiswa dapat membedakan antara fungsi dan prosedur

Prosedur vs Fungsi

Proses menghitung Tegangan (V) dengan rumus R * A

```
function HITUNG_V(R1,A1:integer)→integer
 {menghasilkan tegangan sebagai hasil kali R dan A}
Algoritma
  → R1 * A1
```

```
procedure HITUNG_V(input R1,A1:integer;output V1:integer)
 {Prosedur untuk memproses tahanan & arus menjadi tegangan
  I.S: R1 dan A1 telah terdefinisi
  F.S: V1 terdefinisi dengan rumus V1=R1*A1}
Algoritma
  V1 ← R1 * A1
```

Definisi Prosedur

- Prosedur adalah sederetan instruksi algoritmik yang diberi nama, dan akan menghasilkan efek neto yang terdefinisi
- Mendefinisikan prosedur berarti
  - menentukan nama prosedur serta parameternya (jika ada)
  - Mendefinisikan **keadaan awal (*initial state*)** dan **keadaan akhir (*final state*)**
  - Prosedur didefinisikan dalam **kamus**
- Cara penulisan spesifikasi
  - prosedur diberi **nama** dan
  - **parameter formal** (jika ada), yang diberi nama dan dijelaskan typenya

Pendefinisian dan Pemanggilan Prosedur

- Sebuah prosedur yang terdefinisi “disimpan” di tempat lain, dan ketika “dipanggil” dengan menyebutkan namanya “seakan-akan” teks yang tersimpan di tempat lain itu menggantikan teks pemanggilan
- Pada saat itu terjadi asosiasi parameter (jika ada)
- Dengan konsep ini, maka IS dan FS dari prosedurlah yang menjamin bahwa eksekusi program akan menghasilkan efek netto yang diharapkan
- Setiap prosedur harus:
  - Didefinisikan (dbuat spesifikasinya) dan dituliskan kode programnya
  - Dipanggil, pada saat eksekusi

Parameter Prosedur

- **Prosedur tanpa parameter** memanfaatkan nilai dari nama-nama yang terdefinisi pada kamus global.
  - Pemakaiannya biasanya harus “hati-hati”, apalagi jika teks program sudah sangat besar dan implementasinya menjadi banyak file
- **Prosedur berparameter** dirancang, agar sepotong kode yang sama ketika eksekusi dilakukan, dapat dipakai untuk nama parameter yang berbeda-beda
  - Nama parameter yang dituliskan pada definisi/spesifikasi prosedur disebut sebagai **parameter formal**
  - Sedangkan parameter yang dituliskan pada pemanggilan prosedur disebut sebagai **parameter aktual**

Parameter Formal

Parameter Formal: nama-nama variabel (list nama) yang dipakai dalam mendefinisikan prosedur, dan membuat prosedur tersebut dapat dilaksanakan dengan nama-nama yang berbeda ketika dipanggil.

- Parameter formal adalah list nama yang akan dipakai pada prosedur, yang nantinya akan diasosiasikan terhadap nama variabel lain pada saat pemanggilan
- Sesuai dengan ketentuan nilainya, ada 3 type:
  - **Parameter Input:** parameter yang diperlukan prosedur sebagai masukan untuk melakukan aksi yang efektif
  - **Parameter Output:** parameter yang nilainya **akan** dihasilkan oleh prosedur. Hasil nilai akan disimpan pada parameter output ini.
  - **Parameter Input/Output**: parameter yang nilainya diperlukan prosedur sebagai masukan untuk melakukan aksi, dan pada akhir prosedur akan dihasilkan nilai yang baru

Parameter Aktual

Parameter Aktual: nama-nama informasi yang dipakai ketika prosedur itu dipakai ("dipanggil").

- Parameter aktual dapat berupa nama atau harga, tetapi harus berupa nama jika parameter tersebut adalah parameter output (karena hasilnya akan disimpan dalam nama tersebut)
- Sesuai dengan jenis parameter formal, parameter aktual pada saat pemanggilan:
  - **Parameter Input** harus terdefinisi nilainya (karena dibutuhkan oleh prosedur untuk menghasilkan nilai)
  - **Parameter Output** tidak perlu terdefinisi nilainya, tetapi justru setelah pemanggilan prosedur akan dimanfaatkan oleh deretan instruksi berikutnya, karena nilainya akan dihasilkan oleh prosedur
  - **Parameter Input/Output** harus terdefinisi nilainya dan nilai baru yang diperoleh karena eksekusi prosedur akan dimanfaatkan oleh deretan instruksi berikutnya

Pemanggilan Prosedur

- Memakai atau “memanggil” prosedur adalah menuliskan nama prosedur yang pernah didefinisikan, dan memberikan harga-harga yang dibutuhkan oleh prosedur itu untuk dapat melaksanakan suatu aksi tertentu
- Sebuah prosedur juga boleh “memakai” atau memanggil prosedur
- Pada saat eksekusi, terjadi asosiasi nama parameter formal dengan nama parameter aktual
- Pada notasi algoritmik, asosiasi dilakukan dengan cara **“by position”**, urutan nama parameter aktual akan diasosiasikan sesuai dengan urutan parameter formal. Karena itu, type harus kompatibel.

Kamus Lokal

- Prosedur dapat mempunyai **kamus lokal**, yaitu pendefinisian nama yang dipakai dan hanya berlaku dalam ruang lingkup prosedur tersebut
- Jika nama yang dipakai di dalam prosedur tidak terdefinisi dalam list parameter formal atau dalam kamus lokal, maka nama tersebut harus sudah terdefinisi pada prosedur yang memakainya
- Penulisan kamus lokal = kamus global, bedanya adalah lingkup berlakunya nama yang didefinisikan
  - Pada kamus “global”, nama berlaku untuk program dan semua prosedur/fungsi yang didefinisikan
  - Pada kamus lokal, nama berlaku untuk prosedur/fungsi yang bersangkutan dan prosedur/fungsi yang didefinisikan di dalamnya

Program yang Moduler

- Adalah program yang dibagi-bagi menjadi modul-modul yang terdefinisi dengan baik dalam bentuk prosedur-prosedur
- Setiap prosedur harus jelas definisi dan ruang lingkupnya, supaya dapat dipanggil secara independen
- Pembagian program besar dalam prosedur-prosedur akan mempermudah pembagian kerja di antara beberapa pemrogram
- Penulisan prosedur juga akan memudahkan program untuk dibaca oleh “manusia” karena kita tidak perlu terpaku pada detil kode prosedur untuk mengerti efek neto yang dihasilkannya
- Dalam beberapa hal, pemrogram tidak perlu tahu sama sekali “isi” atau kode dari prosedur dengan mengetahui spesifikasinya
- Beberapa bahasa pemrograman menyediakan prosedur terdefinisi yang sering dipakai dalam memrogram sehingga pemrogram tidak perlu lagi menuliskan kodenya

Notasi Algoritmik untuk Pendefinisian Prosedur (1)

procedure NAMAPROSEDUR (Input/Output : [*list nama parameter formal*: *type*])

{Spesifikasi, Initial State, Final State}

Kamus lokal:

```
{semua NAMA yang dipakai dalam BADAN
PROSEDUR}
```

Algoritma:

```
{BADAN PROSEDUR}
{deretan instruksi pemberian harga, input,
output, analisa kasus, pengulangan atau
prosedur}
```

Notasi Algoritmik untuk Pendefinisian Prosedur (2)

Dengan syarat:

- nama prosedur dan parameternya harus disebutkan dalam kamus pemanggil
- list parameter formal boleh tidak ada (kosong), dalam hal ini di dalam prosedur akan dipakai nama lokal dan nama-nama yang telah terdefinisi dalam kamus “pemakai”-nya
- jika list parameter ada (tidak kosong, minimal satu nama), maka harus berupa satu atau beberapa nama INFORMASI beserta typenya

Notasi Algoritmik untuk Pemanggilan Prosedur (1)

Program POKOKPERSOALAN

{Spesifikasi, Input, Proses, Output}

Kamus:

```
{semua NAMA yang dipakai dalam algoritma}
   procedure NAMAPROSEDUR (Input/Output : [list nama parameter formal: type])
   {Spesifikasi, Initial State, Final State}
```

Algoritma:

```
{deretan instruksi pemberian harga, input, output, analisa kasus, pengulangan}
   NAMAPROSEDUR (list parameter aktual)
```

Notasi Algoritmik untuk Pemanggilan Prosedur (2)

Dengan syarat:

- Pada waktu pemanggilan terjadilah korespondensi antara parameter formal dengan parameter aktual sesuai dengan urutan penulisan dalam list-nama parameter formal
- List parameter aktual harus sama jumlah, urutan, dan type-nya dengan list parameter formal
- List parameter aktual yang berupa **input** dapat berupa nama INFORMASI atau KONSTANTA yang telah terdefinisi dalam kamus atau konstanta, dapat juga berupa harga konstanta, atau harga yang dihasilkan oleh suatu ekspresi atau fungsi
- List parameter aktual yang berupa **output** harus berupa nama INFORMASI
  - Jika didefinisikan sebagai input, walaupun pernah diubah dalam badan prosedur, isi dari nama yang dipakai pada parameter aktual tidak pernah berubah
  - Jika didefinisikan sebagai output dan parameter formal korespondensinya pernah diubah harganya dalam badan prosedur, isinya akan berubah

Contoh 1 – Prosedur: Voltage

- Solusi 1: Prosedur tanpa parameter (hal 81)
- Solusi 2: Prosedur dengan parameter (hal 81)

Catatan:

1. Prosedur dengan parameter lebih menjamin modularitas program. Sedapat mungkin semua prosedur diparameterisasi dengan baik
2. Prosedur tanpa parameter bekerja dengan nama global. Hanya boleh dipakai untuk kasus yang sangat khusus, yaitu jika nama global merupakan "*universe*" (dunia lingkungan) dari program
3. Prosedur tanpa parameter tidak boleh dipakai jika alasannya hanya karena pemrogram malas menuliskan parameter!

Program VOLTAGE1
{Program yg membaca tahanan dan arus, menghitung Voltage dan mencetak hasil perhitungan, versi tanpa parameter}

Kamus

```
 R: integer {tahanan dlm ohm}
 A: integer {arus dlm ampere}
 V: integer {tegangan dlm volt}
 procedure HITUNG_V1
 {Prosedur untuk memproses tahanan & arus menjadi
tegangan}
```

Algoritma

```
  input(R,A)
  HITUNG_V1
  output(V)
```

```
procedure HITUNG_V1
{Prosedur untuk memproses tahanan & arus menjadi tegangan
  I.S: R dan A telah terdefinisi
  F.S: R dan A diproses dengan rumus V=R*A}
```

Kamus Lokal

Algoritma

```
 V ← R*A          {variabel global}
```

Program VOLTAGE2
{Program yg membaca tahanan dan arus, menghitung Voltage dan mencetak hasil perhitungan, versi dengan parameter}

Kamus

```
 R: integer {tahanan dlm ohm}
 A: integer {arus dlm ampere}
 V: integer {tegangan dlm volt}
 procedure HITUNG_V2(input R1,A1:integer;output
V1:integer)
 {Prosedur untuk memproses tahanan & arus menjadi
tegangan}
```

Algoritma

```
  input(R,A)
  HITUNG_V2(R,A,V)
  output(V)
```

```
procedure HITUNG_V2(input R1,A1:integer;output V1:integer)
{Prosedur untuk memproses tahanan & arus menjadi tegangan
  I.S: R dan A telah terdefinisi
  F.S: R dan A diproses dengan rumus V=R*A}
```

Kamus Lokal

Algoritma

```
 V1 ← R1*A1          {variabel global}
```

Contoh 2 – Prosedur: TUKAR

Prosedur untuk menukar dua harga yang disimpan dalam dua nama a dan b

IS: diberikan a= A dan b = B

FS: a=B dan b=A

(halaman 82)

Program TUKAR
{Program yg membaca dua bilangan bulat dan menukarkannya}

Kamus

```
I1,I2: integer {nilai integer yang akan ditukarkan}
procedure tukar2var(input/output tkr1,tkr2:integer)
{Prosedur untuk menukar dua bilangan}
```

algoritma

```
input(I1,I2)
tukar2var(I1,I2)
output(I1,I2)
```

procedure tukar2var(input/output tkr1,tkr2:integer)
{Program yg membaca dua bilangan bulat dan menukarkannya
I.S: tkr1 dan tkr2 telah terdefinisi
F.S: nilai tkr1 dan tkr2 telah ditukar}

Kamus Lokal
```
temp: integer
```

algoritma
```
temp ← tkr1
  tkr1 ← tkr2
  tkr2 ← temp
```

Program TUKAR3BIL
{Program yg membaca tiga bilangan bulat dan menukarkannya}

Kamus

```
I1,I2,I3: integer {nilai integer yang akan ditukarkan}
procedure tukar2var(input/output tkr1,tkr2:integer)
{Prosedur untuk menukar dua bilangan}
```

algoritma

```
input(I1,I2,I3)
tukar2var(I1,I3) {I1=I3,I2=I2,I3=I1}
tukar2var(I2,I3) {I1=I3,I2=I1,I3=I2}
output(I1,I2,I3)
```

Translasi ke Pascal

Pendefinisian/Spesifikasi Prosedur

```
(* procedure NAMAPROSEDUR (input [list nama parameter formal: type]),
                          input/output [list nama parameter formal: type]),
                          output [list nama parameter formal: type])
*)
procedure NAMAPROSEDUR ( [list nama parameter formal: type],
                         VAR [list nama parameter formal: type] );

(* Spesifikasi , Initial State, Final State *)
```

```
(* Kamus lokal: boleh mengandung VAR, TYPE, CONST *)
(* semua NAMA yang dipakai dalam BADAN PROSEDUR *)
```

```
(* ALGORITMA *)
begin
   (* BADAN PROSEDUR *)
   (* deretan instruksi pemberian harga, input, output, analisis kasus,
      pengulangan atau prosedur *)

end;
```

Pemanggilan Prosedur

```
(* Program POKOKPERSOALAN *)
Program POKOKPERSOALAN;
(* Spesifikasi , Input, Proses, Output *)
```

```
KAMUS
(* semua NAMA yang dipakai dalam program }
(* semua NAMA yang dipakai dalam program *)
(* TYPE  *)
(* CONST  *)
(* VAR    *)


(*        spesifikasi dan Body prosedur langsung dituliskan di bagian ini *)

(* procedure NAMAPROSEDUR  (input/output : <list-nama parameter formal>) *)
 procedure NAMAPROSEDUR  (<list-nama parameter formal Input:type ;
           VAR <list-nama parameter formal Input/Output atau output:type>) ;
{ Spesifikasi : Initial State, Final State}
{ Boleh mengandung kamus lokal VAR, TYPE, CONST }
begin
end;
```

```
(* ALGORITMA program utama *)
begin
{Deretan instruksi assignment/pemberian harga, input, output, analisis kasus,
pengulangan )

          NAMAPROSEDUR  (<list parameter aktual>)
end.
```

Unit (dalam Pascal)

- Saat membuat program ‘skala besar’ dan kompleks, lebih memudahkan untuk memecah program menjadi modul yang lebih kecil (dalam file terpisah)
- Unit hanyalah bagian program untuk:
  - Deklarasi tipe bentukan
  - Deklarasi dan implementasi prosedur/ fungsi/ primitif untuk tipe bentukan tersebut
- Karena hanya bagian program, harus dibuat suatu **Program Utama** (*driver*) yang memanfaatkan **unit**
- Kerangka :

```
unit
interface
implementation
end.
```

Contoh Unit

```
(* File: unit.pas*)
unit upoint;
(* unit yang berisi definisi tipe point dan beberapa primitif untuk manipulasinya *)
```

```
Interface
Const
          MaxPoint: 8;
(*KAMUS*)
(* TYPE  *)
Type Point = record
          x : integer;
          y : integer;
          end;
(* CONST  *)
(* VAR    *)

(*spesifikasi prosedur dituliskan di bagian ini *)
procedure MakePoint (x,y: integer; var P: Point);
{*Membentuk point dari x dan y*}
```

```
(* implementasi *)
implementation
(*spesifikasi prosedur dituliskan di bagian ini *)
procedure MakePoint (x,y: integer; var P: Point);
begin
          P.x := x; P.y := y;
end;
end.
```

Contoh *Driver*

```
(* File: mainpoint.pas*)
Program mainpoint;
uses upoint;
(* unit yang berisi definisi tipe point dan beberapa primitif untuk manipulasinya *)
```

```
(*KAMUS*)
 (* CONST  *)
(* VAR    *)
          P : Point;
```

```
(* Algoritma Utama *)
begin
          MakePoint(5,5,P);
end.
```

Library

- Library adalah kumpulan unit yang sudah disediakan oleh bahasa pemrograman (Pascal)
- Program yang memanfaatkan unit dari Pascal harus menuliskan secara eksplisit unit apa saja yang digunakan
- Beberapa unit yang ada di Pascal:
  - crt: screen and keyboard handling unit
  - graph: unit to handle screen graphics
  - math: Additional mathematical routines
  - printer: Provide access to the printer
  - etc

Latihan Soal

- Pecahan (hal 75)
  - Definisikan sebuah type pecahan yang terdiri dari pembilang dan penyebut bilangan integer, dan sekumpulan prosedur yang merupakan realisasi dari operator pecahan:
    - JumlahP: menerima dua pecahan, menghasilkan jumlah berupa pecahan
    - KurangP: menerima dua pecahan, menghasilkan selisih berupa pecahan
    - KaliP: menerima dua buah pecahan, menghasilkan hasil kali berupa pecahan
    - BagiP: menerima dua buah pecahan, menghasilkan hasil bagi berupa pecahan