Bahasa Pemrograman

S1 & D3 PT Elektronika

Priyanto
E-mail: priyanto@uny.ac.id

Kompetensi

Mahasiswa mampu membuat program untuk menyelesaikan permasalahan komputasi yang ada, menggunakan aturan bahasa pemrograman prosedural dengan mengacu pada metrik perangkat lunak yaitu **kopling rendah** dan **kohesi tinggi**.

Produk Mata Kuliah ini bukan program, tapi (perilaku) membuat program yang baik dan benar

Indikator Pencapaian Kompetensi

- Mengimplementasikan logika program ke dalam struktur bahasa pemrograman non prosedural.
- Menggunakan diagram alir untuk menuangkan algoritma program dan menterjemahkan ke dalam statemen program
- Membuat PROSEDUR dan FUNGSI program dengan berbagai variasi *passing parameter*
- Merubah Fungsi ke Prosedur dan sebaliknya
- Mengukur kualitas MODUL PROGRAM (Fungsi & Prosedur) menggunakan metrik perangkat lunak
- Mengkonversi program yang sudah ada menjadi *Stuctuce Chart* dan menuangkan *Stuctuce Chart* menjadi program

Sistem Penilaian

1. Uji kompetensi pada kegiatan praktikum (8 kali): Nilai berdasarkan durasi penyelesaian
2. Kuis (Teori)
3. Ujian Tengah Semester (Teori)
4. Kehadiran
5. Keaktifan di dalam kelas
6. Ujian Akhir Semester (tidak ada)

Butir 1 memiliki bobot paling tinggi

Informasi Perkuliahan

- Bahasa Pemrograman yang digunakan: Turbo Pascal 7.1
- Bila dimungkinkan ada Asisten Praktikum
- Labsheet praktikum disediakan

Pustaka

- Pemrograman Pascal Versi 7.1 (bebas)
- Pressman (2005). *Software Engineering: A Practioner Approach*. New York: McGrah-Hill.

Struktur Program Pascal
Tipe Data

Priyanto
E-mail: priyanto@uny.ac.id

Struktur Program Pascal: Kepala

{==========
Dokumentasi Program meliputi:
Informasi kegunaan program
Siapa yang membuat
Tanggal pembuatan
Berapa kali direvisi dan tangggal berapa
Informasi lain
==========}

Struktur Program Pascal: Kepala

```
PROGRAM nama_program; {Judul Program}
USES Crt; {DOS, printer, graph}
CONST  {deklarasi konstanta}
    PI = 3.14;
TYPE   {definisi tipe}
   Pecahan = real;
VAR   {deklarasi Variabel}
    Arus, tegangan: integer;
    Resistor: pecahan;  {=Resistor: real}
```


Struktur Program Pascal: Deklarasi Modul Program

```
PROCEDURE garis (c: char, n: byte);
Var i: byte;
BEGIN
    For i:= 1 to n do
        Write (c);
    Writeln;
END; {Procedure garis}

PROCEDURE . . . . . .
BEGIN
  - - - - - - - -
END;

FUNCTION pangkat3 (. . . . . .)
BEGIN
  - - - - - - - -
END;
```

Modul Program:
- Procedure
- Function

Struktur Program Pascal: Program Utama

```
BEGIN {Program Utama}
    clrscr;
    writeln (……..);
    pangkat3(10);
    garis ('=',30);
    . . . . . .
END. {Program Utama}
```


Struktur Memori (RAM)

1	0	0	0	1	1	1	1	7
1	1	0	0	1	1	1	1	6
1	1	0	0	1	1	1	1	5
1	1	0	0	1	1	1	1	4
1	1	0	0	1	1	1	1	3
1	1	0	0	1	1	1	1	2
1	1	0	0	1	1	1	1	1
0	0	0	1	1	1	0	1	0

Data 8 Bit

Alamat

Pascal Data Types

Name	Type of Data	Examples
String	Holds Text	‘Yogyakarta'
Integer	Holds whole numbers	3, 6, 1024
Real	Holds Decimal Numbers	3.14, 503.2
Boolean	Holds True or False	TRUE, FALSE
Character	Holds a single character	'A', 'E'

Variable Ranges

Data Type	Minimum Value	Maximum Value
Integer	-32,768	32,767
LongInt	-2,147,483,648	2,147,487,647
ShortInt	-128	128
Real	2.9 x 10 E-39	1.7 x 10 E+38

Tipe Data Numerik Integer

Tipe	Ukuran Memori (Byte)	Jangkauan Nilai
Byte	1	0 .. 255
Shortint	1	-128 .. 127
Integer	2	-32768 .. 32767
Word	2	0 .. 65535
Longint	4	-2147483648 .. 2147463674

Tipe Data Numerik Real

Tipe	Ukuran Memori (Byte)	Jangkauan Nilai
Single	4	1.5E-45 .. 3.4E38
Double	8	5.0E-324 .. 1.7E308
Extended	10	1.9E-4951 .. 1.1E4932

Tipe Data Karakter & String

Char: 1 karakter

String[10]: 10 karakter

Boolean: True dan false

Operator Precedence

Priority	Operation
First	* / DIV MOD
Second	+ -

Operator Logika

= equal to
<> not equal to
> greater than
< less than
>= greater than or equal to
<= less than or equal to

www.elearning.uny.ac.id

Flowchart
(Diagram Alir)

Priyanto
E-mail: priyanto@uny.ac.id

Simbol Flowchart

Terminasi — AWAL dan AKHIR suatu aktivitas

Mulai

Selesai

Input Interaktif — Operator harus memasukkan Nilai secara interaktif (Keyboard)

Masukkan Nilai radius

Simbol Flowchart

Proses — Proses atau isi informasi

Hitung Luas

Kondisi Logika — Berisi pertanyaan logika, dengan jawaban True atau False

Nilai A = 0? T F

Simbol Flowchart

A — *On-page* Connector: Sambungan FC pada halaman yang sama

C — *Off-page* Connector: Sambungan FC pada halaman yang berbeda

Predefined Process — Fungsi atau prosedur yang telah didefinisikan


Aktivitas Pagi
A
Persiapan makan
Warung buka?
Y
Makan
Tunggu
T
Y
Waktu cukup?
T
Berangkat
Selesai
Flowchart oleh Priyanto
7

Procedure & Function

Pass by Value

Priyanto
E-mail: priyanto@uny.ac.id

Format Program

```
PROGRAM nama_program;
USES Crt; {DOS, printer, graph}
CONST  {deklarasi konstanta}
TYPE   {deklarasi tipe}
VAR    {deklarasi Variabel}

PROCEDURE garis (c: char, n: byte);
Var i: byte;
BEGIN
    For i:= 1 to n do
        Write (c);
    Writeln;
END; {Procedure garis}

FUNCTION . . . . . .
BEGIN
   - - - - - - - -;
END;

BEGIN {Program Utama}
    clrscr;
    writeln (.....)
    garis ('=',30);
    . . . . . .
END. {Program Utama}
```

Deklarasi Subprogram (modul program)
- Procedure
- Function

Program Utama

Procedure & Function

- Prosedur adalah merupakan subprogram
- Procedure diawali dengan kata cadangan PROCEDURE di dalam deklarasi procedure
- Procedure dipanggil dan digunakan di dalam blok program yang lainnya dengan menyebut judul Procedure-nya

Procedure & Function

- Merupakan penerapan konsep program MODULAR, yaitu memecah program yang rumit menjadi subprogram- subprogram yang lebih sederhana.
- Untuk hal-hal yang sering dilakukan berulang-ulang, cukup ditulis sekali, dan dapat digunakan berkali-kali.

Parameter dalam Procedure & Function

- Parameter yang dikirim dari modul utama ke procedure disebut **parameter aktual**.
- Parameter yang ada dan dituliskan pada procedure disebut **parameter formal**.
- Proses pengiriman data lewat parameter aktual ke parameter formal disebut **parameter passing**.
- **Parameter aktual** dan **parameter formal** harus memiliki tipe sama
- **Parameter passing: *by value* dan *by reference***

Procedure: Pass by Value

(Pengiriman parameter secara nilai)

- Bila parameter dikirim secara nilai, parameter formal di procedure akan berisi nilai yang dikirimkan yang kemudian bersifat lokal di procedure.
- Bila nilai parameter formal di procedure tersebut berubah, tidak akan mempengaruhi nilai parameter aktual.
- Pengiriman secara nilai merupakan pengiriman searah, yaitu dari parameter aktual ke parameter formal.

Procedure: Contoh

```
PROGRAM Jumlah;

PROCEDURE Hitung (X, Y, Z: Integer);
Var
   V: Integer;
Begin
   V := X + Y + Z;
   Writeln (' Nilai V = ', V);
End; {Procedure Hitung}

Var
   A, B, C: Integer;

BEGIN {Program Utama}
   clrscr;
   A := 2;  B := 5;  C := 5;
   Hitung (A, B, C);
END. {Program Utama}
```

Nilai V = 12

Procedure

Pass by Value: Tracing

```
BEGIN
   clrscr;
   A := 2;  B := 5;  C := 5;
   Hitung (A, B, C);
END.
```

2 5 5

```
PROCEDURE Hitung (X, Y, Z: Integer);
Var
   V: Integer
Begin
   V := X + Y + Z;
   Writeln (' Nilai V = ', V);
End; {Procedure Hitung}
```

V = 12

Procedure

Pass by Value: Tracing

```
BEGIN
   Writeln ('Pascal');
   Garis ('*',6)
END.
```

* 6

```
PROCEDURE Garis (C: char, N: byte);
Var i: byte;
Begin
   For i:= 1 to N do
      Write (C);
   Writeln;
End;
```

Pascal
******

Function

- Function hampir sama dengan procedure, tetapi harus dideklarasikan dengan tipenya.
- Tipe deklarasi menunjukkan tipe hasil function

Function Nama_fungsi (*daftar parameter*): type;

Function: Contoh

```
FUNCTION Hitung (X, Y, Z: Integer): integer
Var
   V: Integer
Begin
   V := X + Y + Z;
   Hitung := V;
End; {Function Hitung}

Var
   A, B, C, D: Integer;

BEGIN {Program Utama}
   clrscr;
   A := 2;  B := 5;  C := 5;
   D := Hitung (A, B, C);
   Writeln ('Nilai D = ', D);
END. {Program Utama}
```

Nilai D = 12

Function

Pass by Value: Tracing

```
BEGIN
   clrscr;
   A := 2;  B := 5;  C := 5;
   D := Hitung (A, B, C);
   Writeln ('Nilai D = ', D);
END.
```

12 2 5 5

```
FUNCTION Hitung (X, Y, Z: Integer): Integer
Var
   V: Integer
Begin
   V := X + Y + Z;
   Hitung := V;
End;
```

V = 12

Procedure & Function
Pass by Reference

Priyanto
E-mail: priyanto@uny.ac.id

Format Program

```
PROGRAM nama_program;
USES Crt; {DOS, printer, graph}
CONST  {deklarasi konstanta}
TYPE   {deklarasi tipe}
VAR    {deklarasi Variabel}

PROCEDURE garis (c: char, n: byte);
Var i: byte;
BEGIN
   For i:= 1 to n do
      Write (c);
   Writeln;
END; {Procedure garis}

BEGIN {Program Utama}
   clrscr;
   writeln (……..)
   garis ('=',30);
   . . . . . .
END. {Program Utama}
```

Deklarasi Subprogram (modul program)
- Procedure
- Function

Program Utama

Parameter Yang Dikirim Dari Modul Utama Ke Procedure Disebut **Parameter Aktual**

```
BEGIN {Program Utama}
   A:= 10; B:= 20;
   tukar (A, B);
   writeln ('A = ', A, 'B = ', B);
END. {Program Utama}
```


Parameter Yang Ada Dan Dituliskan Pada Procedure Disebut **Parameter Formal**

```
PROCEDURE tukar (var X, Y: Integer);
Var Z: Integer;
BEGIN
   Z := X;
   X := Y;
   Y := Z;
END; {Procedure tukar}
```


Parameter Aktual & Parameter Formal HARUS memiliki TIPE SAMA

```
Var A, B: Integer;
BEGIN {Program Utama}
   A:= 10; B:= 20;
   tukar (A, B);
   writeln ('A = ', A, 'B = ', B);
END. {Program Utama}

PROCEDURE tukar (var X, Y: Integer);
Var Z: Integer;
BEGIN
   Z := X;
   X := Y;
   Y := Z;
END; {Procedure tukar}
```


Parameter passing: by value dan by reference

```
PROCEDURE garis (c:char, n:byte);
Var i: byte;
BEGIN
   For i:= 1 to n do
      Write (c);
   Writeln;
END; {Procedure garis}

BEGIN {Program Utama}
   clrscr;
   writeln (……..)
   garis ('=',30);
END. {Program Utama}
```

```
PROCEDURE tukar (var X, Y:Integer);
Var Z: Integer;
BEGIN
   Z := X;
   X := Y;
   Y := Z;
END; {Procedure tukar}

BEGIN {Program Utama}
   A:= 10; B:= 20;
   tukar (A, B);
   writeln ('A = ', A, 'B = ', B);
END. {Program Utama}
```

Procedure Tukar → Pass by Reference

```
BEGIN {Program Utama}
    A:= 10; B:= 20;
    tukar (A, B);
    writeln ('A = ', A, 'B = ', B);
END. {Program Utama}

PROCEDURE tukar (var X, Y: Integer);
Var Z: Integer;
BEGIN
    Z := X;
    X := Y;
    Y := Z;
END; {Procedure tukar}
```


Procedure Tukar → Pass by Value

```
BEGIN {Program Utama}
    A:= 10; B:= 20;
    tukar (A, B);
    writeln ('A = ', A, 'B = ', B);
END. {Program Utama}

PROCEDURE tukar (X, Y: Integer);
Var Z: Integer;
BEGIN
    Z := X;
    X := Y;
    Y := Z;
END; {Procedure tukar}
```


Pass by Value vs Pass by Reference

Pass by value	Pass by reference
• Parameter formal akan berisi nilai yang dikirimkan yang kemudian bersifat lokal.	• Parameter formal akan berisi nilai yang dikirimkan yang kemudian bersifat tidak lokal.
• Perubahan nilai parameter formal, tidak mempengaruhi nilai parameter aktual.	• Perubahan nilai parameter formal, mempengaruhi nilai parameter aktual.
• Merupakan pengiriman satu arah, yaitu dari parameter aktual ke parameter formal	• Merupakan pengiriman dua arah, dari parameter aktual ke parameter formal dan sebaliknya

F&P bisa memiliki PbV, PbR, atau Campuran

```
FUNCTION Hitung
   (X, Y, Z: Integer): Integer
Begin
   Hitung := X + Y + Z;
End; {Function Hitung}

BEGIN {Program Utama}
    clrscr;
    A := 2;  B := 5;  C := 5;
    D := Hitung (A, B, C);
END. {Program Utama}
```

```
PROCEDURE Hitung
   (X, Y, Z: Integer, var H: Integer);
Begin
    H := X + Y + Z;
End; {Procedure Hitung}

BEGIN {Program Utama}
    clrscr;
    A := 2;  B := 5;  C := 5;
    Hitung (A, B, C, D);
END. {Program Utama}
```


Procedure dengan Passing Parameter Campuran

Tracing

```
Var D: Integer;
BEGIN {Program Utama}
    clrscr;
    A := 2;  B := 5;  C := 5;
    Hitung (A, B, C, D);
END.

PROCEDURE Hitung (X, Y, Z: Integer, var H: Integer);
Begin
    H := X + Y + Z;
End; {Procedure Hitung}
```

2 5 5 12

2 5 5 12

http://elearning.uny.ac.id

Priyanto
E-mail: priyanto@uny.ac.id

Array

Priyanto
E-mail: priyanto@uny.ac.id

Loop FOR

```
FOR variabel := nilai_awal TO nilai_akhir DO
    Statemen;
```

```
FOR variabel := nilai_awal TO nilai_akhir DO
Begin
   Statemen1;
   Statemen2;
   Statemen3;
End;
```

Blok Statemen

Contoh Implementasi

```
FOR i := 1 TO 5 DO
    write (i);
writeln;
writeln ('Selesai');
```

Hasil

```
12345
Selesai
```

```
FOR i := 1 TO 5 DO
    x[i] := i + 2;
writeln (x[i]);
writeln ('Selesai');
```

Hasil

```
7
Selesai
```


Array vs Non Array

Array

```
X[1] := 10;
X[2] := 20;
X[3] := 30;
X[4] := 40;
X[5] := 50;
```

Satu Variabel

Non Array

```
X1 := 10;
X2 := 20;
X3 := 30;
X4 := 40;
X5 := 50;
```

Lima Variabel

Index Variabel

Variabel Non Array

```
Var x1, x2, x3, x4, x5: integer;
Begin
   Write ('Nilai x1 = '); Readln (x1);
   Write ('Nilai x2 = '); Readln (x2);
   Write ('Nilai x3 = '); Readln (x3);
   Writeln ('Nilai x1 = ', x1);
   Writeln ('Nilai x2 = ', x2);
   Writeln ('Nilai x3 = ', x3);
End.
```


Deklarasi Array

```
Var
   X: array[1..100] of integer;
```

Tipe data Array

Index Array

Nama Array

Variabel Array

```
Var x: array[1..5] of integer;
    i : integer;
Begin
  For i:= 1 to 5 do
  Begin
    Write ('Nilai x ', I, '='); Readln (x[i]);
  End;

  For i:= 1 to 5 do
    Writeln ('Nilai x ',i, '= ',  x[i]);
End.
```

Hasil:

```
Nilai x1 = 2
Nilai x2 = 3
Nilai x3 = 4
Nilai x4 = 5
Nilai x5 = 6

Nilai x1 = 2
Nilai x2 = 3
Nilai x3 = 4
Nilai x4 = 5
Nilai x5 = 6
```


Variabel Array: Contoh Implementasi

```
Var x: array[1..5] of integer;
    i: integer;
Begin
  For i:= 1 to 5 do
  Begin
    Write ('Nilai x ', i); Readln (x[i]);
  End;

  For i:= 1 to 5 do
    x[i] := i + 3;

  For i:= 1 to 5 do
    Writeln ('Nilai x ',i, '= ',  x[i]);
End.
```

Hasil:

```
Nilai x1 = 2
Nilai x2 = 3
Nilai x3 = 4
Nilai x4 = 5
Nilai x5 = 6

Nilai x1 = 5
Nilai x2 = 6
Nilai x3 = 7
Nilai x4 = 8
Nilai x5 = 9
```


Menghitung Rerata

```
Const n = 5;
Var x: array[1..n] of integer;
    i, sum: integer; rerata: real;
Begin
  For i:= 1 to n do
    x[i] := i + 3;
  sum := 0;
  For i:= 1 to n do
    sum := sum + x[i];
  rerata := sum/n;
  Writeln ('Rerata = ', rerata);
End.
```

Hasil → **Rerata = 6**

Array
Sebagai Parameter Function/Procedure

Array dapat digunakan sebagai parameter by value atau by reference

```
Program Penjumahan_Vektor;
Uses Ctr;
Type larik = array [1..5] of integer;
Var x, y, z: larik
Procedure jumlah_vektor (x,y: larik; var z: larik);
Begin
  ……
End;
```


Penjumlahan 2 Vektor

```
Program vektor;
uses crt;
type larik = array[1..5] of integer;
var a,b,c: larik;
   i: byte;

Procedure Jumlah_vektor (x,y: larik; var z: larik);
var i: byte;
Begin
  for i:= 1 to 5 do
    z[i] := x[i] + y[i];
end;
```

Penjumlahan 2 Vektor

```
Begin {program utama}
   clrscr;
   for i:= 1 to 5 do
      a[i] := i;
   for i:= 1 to 5 do
      b[i] := i+1;
   jumlah_vektor (a,b,c);
   for i:= 1 to 5 do
      writeln (c[i]);
   Readln;
end.
```


Tugas

baca_input (3, ‘Nilai Resistor’, resistor);

Procedure baca_input (n: byte; kalimat: teks; var r: larik);

Implementasi Array Bubble Sort

Priyanto
E-mail: priyanto@uny.ac.id

Bubble Sort

Cara:

- Meletakkan elemen dengan nilai paling besar pada posisi terakhir (posisi ke N), kemudian nilai paling besar kedua pada posisi N-1, dst.
- Meletakkan nilai terkecil pada 1, nilai terkecil kedua diletakkan pada posisi 2, dst

Algoritma Bubble Sort

Langkah ke 0:	Baca vektor yang akan diurutkan
Langkah ke 1:	Kerjakan langkah 2 untuk I = 1 sampai N-1
Langkah ke 2:	Kerjakan langkah 3 untuk J = 1 sampai N-I
Langkah ke 3:	Test → apakah A[J] > A[J+1]? Jika YA tukar kedua elemen ini
Langkah ke 4:	Selesai

Procedure Bubble Sort

```
Procedure BubbleSort (Var A: larik; N: integer);
var I, J: integer;
begin
    for I := 1 to N-1 do
        for J : = 1 to N-I do
            if A[J] > A[J+1] then
                Tukarkan (A[J], A[J+1])
end;
```

Pengurutan Vektor:	24	23	56	45	12

Proses Iterasi 1

Pengurutan Vektor:	24	23	56	45	12

Iterasi ke	A[1]	A[2]	A[3]	A[4]	A[5]
I = 1	23	24	56	45	12
	23	24	56	45	12
	23	24	45	56	12
	23	24	45	12	56

Proses Iterasi 2

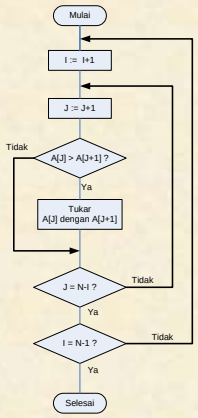
Iterasi ke	A[1]	A[2]	A[3]	A[4]	A[5]
	23	24	45	12	56
I = 2	23	24	45	12	56
	23	24	45	12	56
	23	24	12	45	56

Proses Iterasi 3-4

	23	24	12	45	56
I = 3	23	24	12	45	56
	23	12	24	45	56
I = 4	12	23	24	45	56
Akhir	12	23	24	45	56

24 July, 2014 Implementasi Array: Bubble Sort 7

Flow Chart



24 July, 2014 Implementasi Array: Bubble Sort 8

Materi Kuliah ini tersedia di
http//elearning.uny.ac.id

Terima Kasih

Priyanto
E-mail: priyanto@uny.ac.id

24 July 2014 Implementasi Array: Bubble Sort 9

Record

Priyanto
E-mail: priyanto@uny.ac.id

Apa itu Record?

- Adalah tipe data (baru) yang user defined, berisi elemen tipe data standar.
- Tidak seperti arrays, elemen record dapat berisi tipe data yang berbeda.

```
Type
 Data_mahasiswa= record
          Nama: string[10];
          NIM   : String[10];
          Nilai   : integer;
         end;
Var
  data: Data_mahasiswa;
```


Field, Record, dan Tabel

Nama	NIM	Nilai
Bambang	02333	90
Edi	02334	80

- Baris pada setiap tabel disebut **record**
- Elemen setiap Record disebut **Field**
- Kumpulan beberapa record menjdi **tabel**.

Contoh Record Mahasiswa

```
Type
 Data_mahasiswa= record
          Nama: string[10];
          NIM   : String[10];
          Nilai   : integer;
         end;
Var
  data: data_mahasiswa;
```


Input Data Record Mahasiswa

```
Begin
  for i:= 1 to n do
  begin
    write ('Nama  : '); Readln (data.Nama);
    write ('NIM    : '); Readln (data.NIM);
    write ('Nilai   : '); Readln (data.Nilai);
    writeln;
  end;
```


Contoh Lain

```
Type
 Data_listrik = record
          tegangan: real;
          arus: real;
          resistor: real;
         end;
Var
  listrik: data_listrik;

Begin
   Clrscr; write ('Menghitung Arus');
   Write ('Nilai Tegangan = '); readln (listrik.tegangan)
   Write ('Nilai Reasistor  = '); readln (listrik.resistor)
   Listrik.arus := listrik.tegangan/listrik.resistor;
   Writeln ('Nilai Arus = ', listrik.arus:8.2);
   Readln;
End.
```

Array of Record

Data: tabel_mahasiswa

	Nama	NIM	Nilai
[1]	Bambang	02333	90
[2]	Edi	02334	80

- Data[1].nama → Bambang
- Data[1].nilai → 90
- Data[2].nim → 02334

Array of Record

```
Type
 Data_mahasiswa= record
          Nama: string[10];
          NIM   : String[10];
          Nilai   : integer;
         end;
 tabel_mahasiswa = array [1..5] of Data_mahasiswa
Var
  data: tabel_mahasiswa;
```


Input Data Record Mahasiswa

```
Begin
  for i:= 1 to n do
  begin
    write ('Nama  : '); Readln (data[i].Nama);
    write ('NIM    : '); Readln (data[i].NIM);
    write ('Nilai   : '); Readln (data[i].Nilai);
    writeln;
  end;
```


Record & Keyword *With*

```
Begin
  for i:= 1 to n do
     with data[i] do
     begin
        write ('Nama  : '); Readln (Nama);
        write ('NIM    : '); Readln (NIM);
        write ('Nilai   : '); Readln (Nilai);
       writeln;
     end;
end;
```


Record & Array of Record Sebagai Argumen Function/Procedure

Parameter Prosedur

```
type data_mhs = record
          nim: string[10];
          nama: string [10];
          nilai: integer;
        end;
   larik_mhs = array[1..3] of
   data_mhs;
var data: larik_mhs;
```

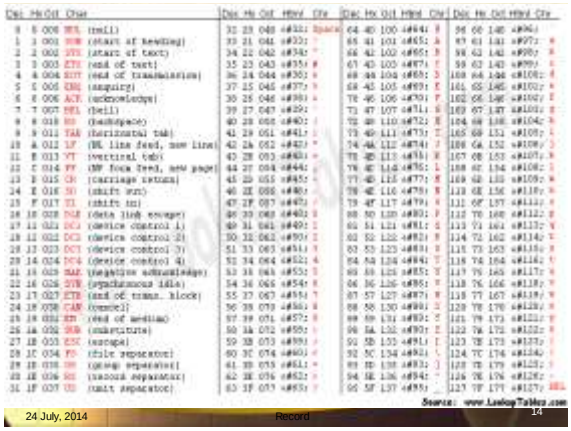
```
Procedure tukar (Var A, B:data_mhs);
Var T: data_mhs;
Begin
  T:= A;
  A:= B;
  B:= T;
End;
```

Parameter Prosedur

```
type data_mhs = record
            nim: string[10];
            nama: string [10];
            nilai: integer; nb: char;
          end;
  larik_mhs = array[1..3] of data_mhs;
var data: larik_mhs;
```

baca_data(data,x);

```
Procedure baca_data (var dat: larik_Mhs; n:byte);
var i: byte;
Begin
  for i:= 1 to n do
    begin
        write ('Nama : '); Readln (dat[i].Nama);
        write ('NIM    : '); Readln (dat[i].NIM);
        write ('Nilai   : '); Readln (dat[i].Nilai);
    end;
end;
```

Silahkan anda latihan Record untuk beberapa contoh kasus

Structured Development

Priyanto
E-mail: priyanto@uny.ac.id

Testing & Siklus pengembangan Software

Analisis → **Desain** → **Implementasi** → **Testing**

Spesifikasi Software
- DFD
- Spesiifikasi lain

Arsitektur Software
- Spesifikasi lain
- DFD diperhalus
- Structure Chart

Pemrograman
- Menterjemahkan Structure Chart menjadi Function dan Procedure

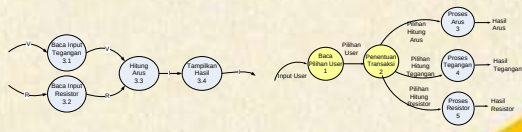
Pengujian

Pengembangan software

- Tahap Analisis menghasilkan DFD (*Data Flow Diagram*)
  - DFD terdiri dari DFD level 0, level 1, sampai level n
  - Pemecahan DFD sampai level berapa (n), sampai mudah diimplementasikan dalam program.
- Tahap Desain akan menghasilkan DFD yang lebih halus dan Structure Chart
  - Apabila DFD sudah cukup, langsung dibuat structure chart-nya
  - Pada tahap ini dibuat **spesifikasi proses** yang mendeskripsikan setiap **bulatan proses** pada DFD
  - Structure Chart menggambarkan struktur pemanggilan Fungsi dan Prosedur dalam suatu program.
- Tahap Implementasi, adalah pembuatan program
- Tahap Testing: menguji per modul dan pengujian seluruh program

2 Macam Data Flow Diagram

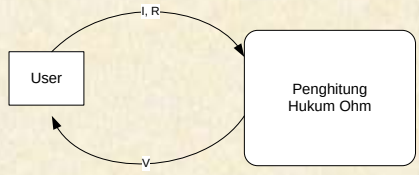
- Transform Flow
- Transaction Flow



Keduanya ada dalam satu program

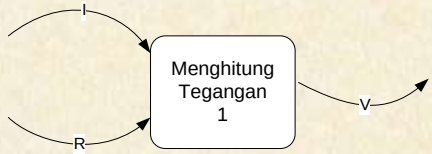
DFD Transform Flow: Dari DFD sampai Program

DFD Level 0 (Contex Diagram)



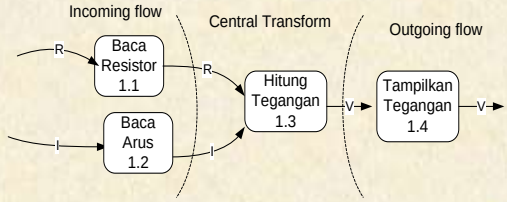
Menggambarkan Sistem sebagai bulatan tunggal dengan entitas yang terlibat

DFD Level 1

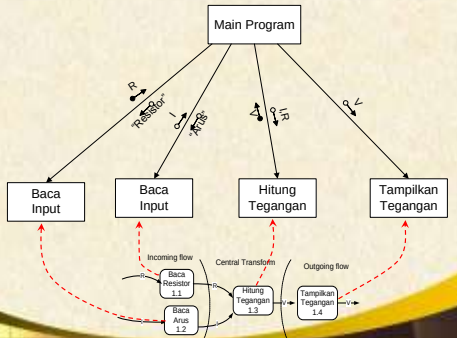


Perhatikan aliran data masuk dan keluar

DFD Level 2



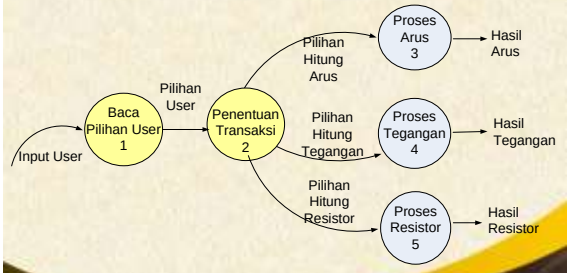
Structure Chart



DFD Transaction & Transform Flow: Partisi DFD sampai Structure Chart

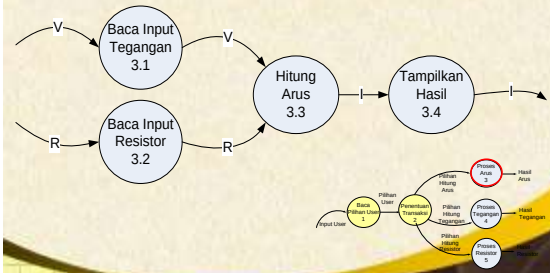
DFD: Transaction Flow

DFD Hukum Ohm LEVEL 1



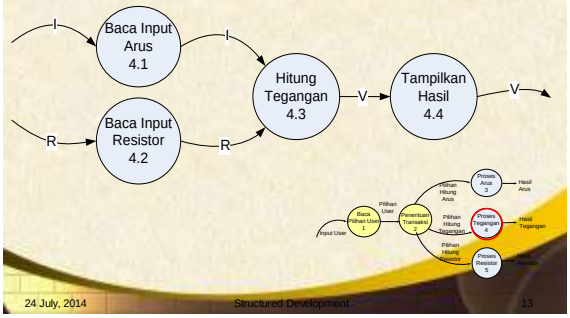
Partisi Proses Arus (Lingkaran 3)

DFD Proses Arus LEVEL 2



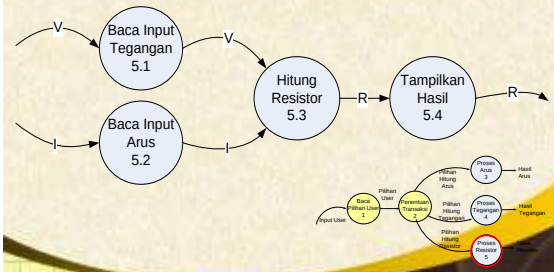
Partisi Proses Tegangan (Lingkaran 4)

DFD Proses Tegangan LEVEL 2

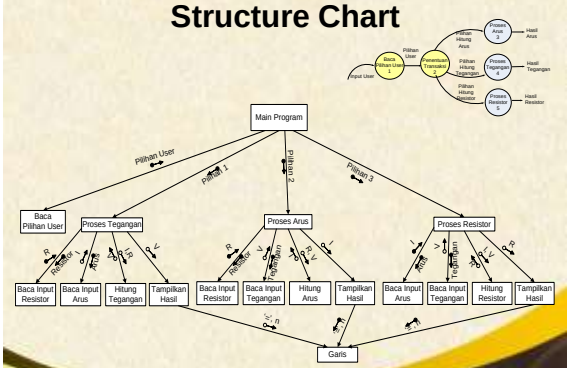


Partisi Proses Resistor (Lingkaran 5)

DFD Proses Resistor LEVEL 2



Structure Chart



Statemen CASE-OF (Selector)

CASE *Ungkapan* OF
  Ungkapan_1 : Statemen 1;
  Ungkapan_2 : Statemen 2;
  Ungkapan_3 : Statemen 3;
  ….
  Ungkapan_n : Statemen n;
END;

Selamat Berlatih

Modul Program yang Efektif

Priyanto
E-mail: priyanto@uny.ac.id

Partisi

- Masalah Besar → membagi/memecah permasalahan tersebut menjadi beberapa bagian yang lebih kecil.
- Menyelesaikan secara bertahap bagian demi bagian, baik secara sendiri maupun berkelompok, sehingga diperoleh solusi dari permasalahan yang besar tersebut.

Partisi

Sebagai gambaran, kita diminta untuk "menyelesaikan" (baca: memakan) buah semangka sampai habis.

- Langkah awal yang dilakukan adalah "mempartisi" (memotong) buah semangka menjadi beberapa bagian
- Memakannya satu persatu, sendiri atau bersama-sama.

Partisi

Program yang besar harus dipartisi menjadi beberapa modul yang mudah diselesaikan

Modul program dapat berupa

- **Procedure** dan/atau
- **Function**.

Mengukur Modul Program yang Baik

- ***Reusable*** adalah kunci pokok dalam pengembangan perangkat lunak, tema inilah yang mengilhami perancangan modul program dan perkembangan paradigma pengembangan perangkat lunak secara umum.
- ***Reusable*** bisa diperoleh bila menerapkan *information hiding*.

Information Hiding

- Setiap modul tersembunyi dengan yang lain
- Modul harus dirancang agar informasi (prosedur dan data) yang berada di dalam modul tidak dapat diakses oleh oleh modul lain yang tidak memerlukan informasi tersebut

Keuntungan Modul yang Efektif

- Mengurangi kompleksitas
- Mempermudah perubahan
- Lebih mudah diimplementasikan dan dapat dikerjakan secara paralel (Tim)
- mudah membagi dalam tim,
- mudah diubah,
- perambatan kesalahan berkurang, dan
- *reusable* bertambah.

Cohesion & Coupling

Independensi diukur dengan dengan dua kriteria kualitattif, yaitu: *Cohesion* dan *Coupling*.

- ***High cohesion*** (*functional cohesion*): modul hanya melakukan satu tugas dan memerlukan sedikit interaksi dengan modul lain dalam satu program.
- ***Low coupling***: modul memiliki kopling antar modul yang lemah atau sebebas mungkin dengan modul yang lain (independen). Kopling tergantung pada kompleksitas antarmuka modul.

Contoh Modul Program

Tidak Baik	Baik

Procedure garis (x:byte,y:char);
Begin
for i : = 1 to x do
write (y);
writeln;
End;
	Procedure garis (x:byte, y:char);
	Var **i: byte**;
	Begin
	for i : = 1 to x do
	write (y);
	writeln;
	End; *{procedure garis }*

Contoh Modul Program

Tidak Baik	Baik

Procedure pangkat_tiga (x: integer);
Var z: integer;
Begin
z := x * x * x;
writeln ('Hasil 1 = ,', z);
buat_garis (10,'=');
End;
	Function pangkat_tiga (x: integer): integer;
	Var z: integer;
	Begin
	z := x * x * x;
	Pangkat_tiga := z;
	End; *{function pangkat_tiga }*

Software Testing

Priyanto
E-mail: priyanto@uny.ac.id

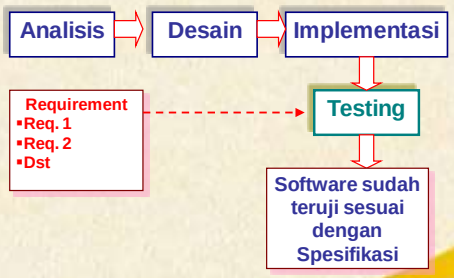
Software Testing Technique

- Testing adalah proses eksekusi program yang bertujuan untuk menemukan kesalahan
- *Testcase* yang baik dapat menemukan error yang belum ditemui sebelumnya.

Prinsip-prinsip Testing

- Seluruh test harus sesuai dengan *customer requirement*
- Dipersiapkan jauh sebelum mulai
- Testing harus dimulai "in the small" dan maju menuju testing "in the large".
- Dilakukan oleh pihak ketiga

Testing & Siklus pengembangan Software



Testcase Design

- **White-box Testing (Glass-box Testing)**
- **Black-box Testing**

White-box Testing

Metode pengujian yang menggunakan struktur kontrol desain prosedural untuk menghasilkan *testcase*.

White-box Testing

- Menjamin seluruh *independent path* di dalam modul diuji minimal satu kali
- Menguji seluruh keputusan logika (True & False)
- Mengeksekusi seluruh *loop* sesuai dengan batasan operasinya
- Menguji struktur data internal untuk menjamin validitas

Basis Path Testing

- Adalah WB testing yang pertama kali diusulkan oleh Tom McCabe (1976)
- Metode *basis path* memungkinkan *testcase designer* memperoleh ukuran kompleksitas lojik dari desain prosedural dan menggunakan ukuran ini sebagai pemandu untuk menentukan *basis set execution path*

Black-box Testing

- Memfokuskan pada kebutuhan **fungsional** program
- **Bukan alternatif** dari white-box testing, tetapi **complementer**

Black-box Testing

Menemukan kesalahan dengan kategori:

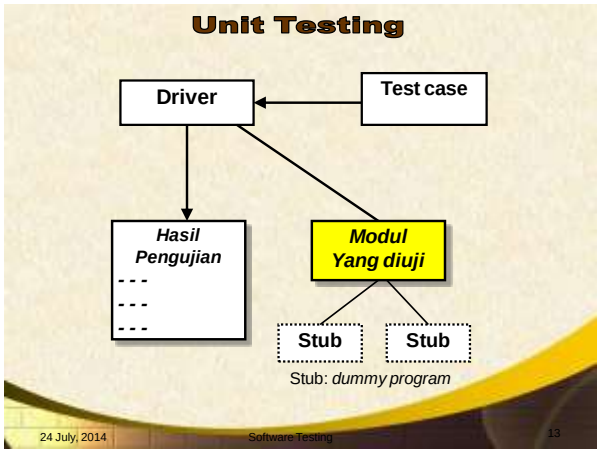
- Fungsi-fungsi yang tidak benar
- Error pada struktur data atau akses database eksternal
- Error Antarmuka, performa
- Error inisialisasi dan terminasi

Black-box Testing

- Menguji beberapa aspek sistem dengan sedikit memperhatikan struktur lojik internal program
- Digunakan untuk menunjukkan fungsi program beroperasi:
  Input diterima → Output benar

Software Testing Strategy

- **Unit Testing**
- **Integration Testing**
- **Validation Testing**
  - **Alpha Testing (**seperti test drive di pabrik mobil)
    - Developer's Site by Customer
    - Pada lingkungan yang terkendali
  - **Beta testing**
    - Customer's sites by end user
    - "live" application pada lingkungan yang tidak bisa dikendalikan developer
    - User melaporkan hasil ke developer
- **System Testing**


Unit Testing
Driver
Test case
Hasil Pengujian
- - -
- - -
- - -
Modul Yang diuji
Stub
Stub
Stub: dummy program
24 July, 2014
Software Testing
13

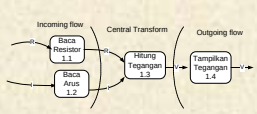

Materi Kuliah ini tersedia di
http//elearning.uny.ac.id
Terima Kasih
Priyanto
E-mail: priyanto@uny.ac.id
24 July, 2014
Software Testing
14

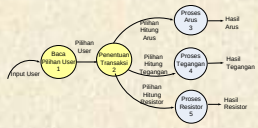
Transaction Flow

Priyanto
E-mail: priyanto@uny.ac.id

2 Macam Data Flow Diagram

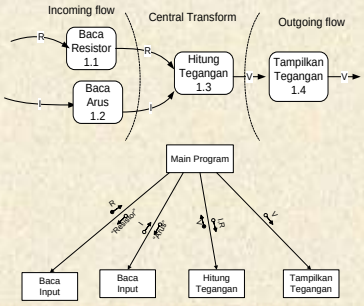
- Transform Flow
- Transaction Flow





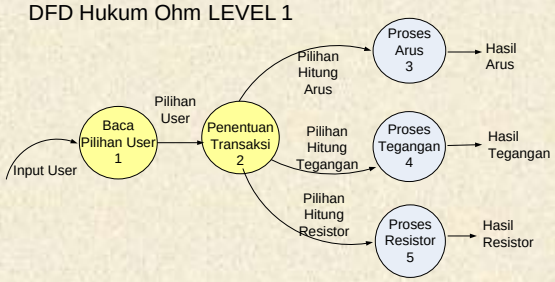
Keduanya ada dalam satu program

DFD: Transform Flow



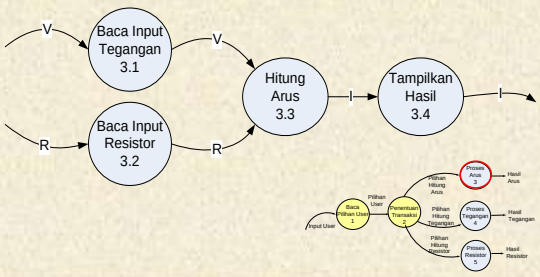
DFD: Transaction Flow

DFD Hukum Ohm LEVEL 1



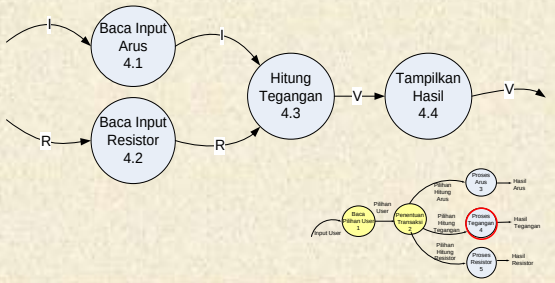
Partisi Proses Arus (Lingkaran 3)

DFD Proses Arus LEVEL 2



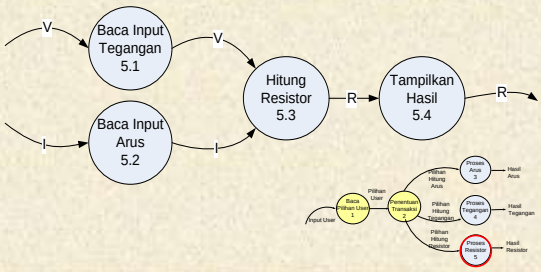
Partisi Proses Tegangan (Lingkaran 4)

DFD Proses Tegangan LEVEL 2

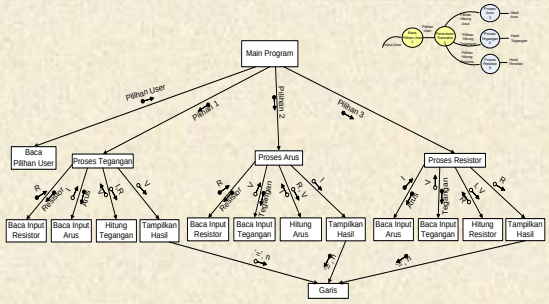


Partisi Proses Resistor (Lingkaran 5)

DFD Proses Resistor LEVEL 2



Structure Chart



Statemen CASE-OF (Selector)

```
CASE Ungkapan OF
    Ungkapan_1 : Statemen 1;
    Ungkapan_2 : Statemen 2;
    Ungkapan_3 : Statemen 3;
    ....
    Ungkapan_n : Statemen n;
END;
```