

TEORI ALGORITMA

Analogi :

- ▶ Jika seseorang ingin mengirim surat kepada kenalnya di tempat lain, langkah yang harus dilakukan adalah:

Langkah :

- ▶ Menulis surat
- ▶ Surat dimasukkan ke dalam amplop tertutup
- ▶ Amplop dikasih alamat penerima dan pengirim
- ▶ Amplop ditemplei perangko secukupnya.
- ▶ Pergi ke Kantor Pos terdekat untuk mengirimkannya

Apa Itu Algoritma ?

- Definisi :

- Urutan langkah-langkah untuk memecahkan masalah yang disusun secara sistematis dan logis.

- Kamus Besar Bahasa Indonesia:
Algoritma adalah urutan logis pengambilan putusan untuk pemecahan masalah.

- Algoritma dibutuhkan untuk memerintah komputer mengambil langkah-langkah tertentu dalam menyelesaikan masalah.

Apa Itu Program/Pemrograman?

Definisi

- Kumpulan instruksi-instruksi tersendiri yang biasanya disebut *source code* yang dibuat oleh programmer (pembuat program).

- Program : Realisasi dari Algoritma.

Program = Algoritma + Bahasa

Mengapa Algoritma ?

- Pembuatan atau penulisan algoritma tidak tergantung pada bahasa pemrograman manapun.
- Notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
- Apapun bahasa pemrogramannya, output yang akan dikeluarkan sama karena algoritmanya sama.

Syarat Algoritma Baik?

- Tingkat kepercayaannya tinggi (*realibility*)
Hasil yang diperoleh dari proses harus berakurasi tinggi dan benar.
- Pemrosesan yang efisien (*cost rendah*)
Proses harus diselesaikan secepat mungkin dan frekuensi kalkulasi yang sependek mungkin.
- Sifatnya general
Bukan sesuatu yang hanya untuk menyelesaikan satu kasus saja, tapi juga untuk kasus lain yang lebih general.

- Bisa dikembangkan (*expandable*)

Haruslah sesuatu yang dapat kita kembangkan lebih jauh berdasarkan perubahan requirement yang ada.

- Mudah dimengerti

Siapa pun yang melihat, dia akan bisa memahami algoritma Anda. Susah dimengertinya suatu program akan membuat susah di-*maintenance* (kelola).

- Portabilitas yang tinggi (*portability*)

Bisa dengan mudah diimplementasikan di berbagai *platform komputer*.

- *Precise* (tepat, betul, teliti)

- Efektif

Tidak boleh ada instruksi yang tidak mungkin dikerjakan oleh pemroses yang akan menjalankannya.

- Harus *terminate*

Jalannya algoritma harus ada kriteria berhenti.

- *Output* yang dihasilkan tepat.

Langkah Pembuatan Program

Mendefinisikan masalah

- a. Kondisi awal, yaitu *input* yang tersedia.
- b. Kondisi akhir, yaitu *output* yang diinginkan.
- c. Data lain yang tersedia.
- d. Operator yang tersedia.
- e. Syarat atau kendala yang harus dipenuhi.

Langkah Pembuatan Program

Buat Algoritma dan Struktur Cara Penyelesaian

- Jika masalahnya kompleks, maka dibagi ke dalam modul-modul

Langkah Pembuatan Program

Menulis program

- Pilihlah bahasa yang mudah dipelajari, mudah digunakan, dan lebih baik lagi jika sudah dikuasai, memiliki tingkat kompatibilitas tinggi dengan perangkat keras dan platform lainnya.

Langkah Pembuatan Program

Mencari Kesalahan

- a. Kesalahan sintaks (penulisan program).
- b. Kesalahan pelaksanaan: semantik, logika, dan ketelitian..

Langkah Pembuatan Program

- Uji dan Verifikasi Program
- Dokumentasi Program
- Pemeliharaan Program

STRUKTUR PENULISAN ALGORITMA

Setiap Algoritma akan selalu terdiri dari tiga bagian yaitu :

- Judul (Header)
- Kamus
- Algoritma

Header (Judul)

Judul adalah bagian teks algoritma yang digunakan sebagai tempat mendefinisikan nama dengan menentukan apakah teks tersebut adalah program, prosedur, fungsi.

```
Program Luas_Kubus      ← {Judul Algoritma}  
{ Menghitung luas kubus untuk ukuran sisi yang dibaca dari piranti masukan lalu  
  mencetak hasilnya kepiranti keluaran; } ← {Spesifikasi Algoritma}
```

Kamus (Deklarasi)

Kamus adalah bagian teks algoritma sebagai tempat untuk mendefinisikan :

- Nama type
- Nama konstanta
- Nama variabel
- Nama fungsi
- Nama prosedur.

Kamus (Deklarasi)

Kamus

{Nama type, hanya untuk type yang bukan type dasar}

type jam : <hh,mm,ss : **integer**> {Type jam terdiri dari 3 masukan yaitu "hh" sebagai jam, "mm" sebagai menit dan "ss" sebagai detik}

{Nama konstanta, harus menyebutkan type dan nilai }

constant phi : **real** = 3,14159

constant nama : **string** = 'Alex'

constant benar : **boolean** = **true**

{Nama Informasi, menyebutkan type}

x,y : **integer** {suatu nilai yang bertipe bilangan bulat}

NMax : **real** {nilai maksimum yang bertipe bilangan real}

Nama : **string** {suatu nilai yang merupakan kumpulan character}

P : **point** {suatu nilai pada bidang kartesian}

Cari : **Boolean** {suatu nilai logika}

Algoritma (Deskripsi)

Algoritma adalah bagian inti dari suatu algoritma yang berisi instruksi atau pemanggilan aksi yang telah didefinisikan.

Algoritma

input (c,d) {menerima masukan 2 bilangan c dan d}

if c < d **then** {operasi kondisional}

e ← a + b {e di *assignment* oleh nilai a dan b}

else

e ← a - b

output (c) {hasil keluaran berupa bilangan c}

PENYAJIAN ALGORITMA

Bentuk penyajian untuk algoritma dibagi menjadi 3 (tiga) bentuk penyajian, yaitu :

- Algoritma dengan struktur Bahasa Indonesia
- Algoritma dengan *Pseudocode*
- Algoritma dengan *Flowchart*

Struktur Bahasa Indonesia

Sifat: Umum

- Tidak menggunakan simbol atau sintaks dari suatu bahasa pemrograman.
- Tidak tergantung pada suatu bahasa pemrograman.
- Notasi-notasinya dapat digunakan untuk seluruh bahasa manapun.

Struktur Bahasa Indonesia

Contoh : Menghitung rata-rata tiga buah data

Algoritma dengan struktur bahasa Indonesia :

- 1) Baca bilangan a, b, dan c
- 2) Jumlahkan ketiga bilangan tersebut
- 3) Bagi jumlah tersebut dengan 3
- 4) Tulis hasilnya

Pseudo-Code

Penyajian algoritma dengan *pseudocode* berarti menggunakan kode yang mirip dengan kode pemrograman yang sebenarnya. Pseudocode lebih rinci dari English/Indonesia *Structure*.

Pseudo-Code

Contoh (1) : Menghitung rata-rata tiga buah data

Algoritma dengan struktur pseudocode :

- 1) *input (a, b, c)*
- 2) *Jml = a+b+c*
- 3) *Rerata = Jml/3*
- 4) *Output (Rerata)*

Flowchart

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urutan prosedur dari suatu program. Flowchart menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian.

Flowchart

Kegunaan:

- Untuk mendesain program
- Untuk merepresentasikan program

Maka, flowchart harus dapat
Merepresentasikan komponen-komponen dalam
bahasa pemrograman

Mengapa Flowchart

a. Relationship

Flowchart dapat memberikan gambaran yang efektif, jelas, dan ringkas tentang prosedur *logic*. *Teknik penyajian yang* bersifat grafis jelas akan lebih baik daripada uraian-uraian yang bersifat teks khususnya dalam menyajikan logika yang bersifat kompleks.

Mengapa Flowchart

b. Analysis




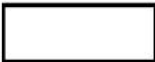
Dengan adanya pengungkapan yang jelas dalam model atau chart, maka para pembaca dapat dengan mudah melihat permasalahan atau memfokuskan perhatian pada area-area tertentu sistem informasi.

Mengapa Flowchart

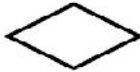
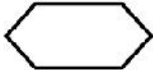

c. Communication

Karena simbol-simbol yang digunakan mengikuti suatu standar tertentu yang sudah diakui secara umum, maka flowchart dapat merupakan alat bantu yang sangat efektif dalam mengkomunikasikan logika suatu masalah atau dalam mendokumentasikan logika tersebut.





Lambang

Keterangan	Lambang
Mulai/selesai (<i>terminator</i>)	
Aliran data	
Input/Output	
Proses	

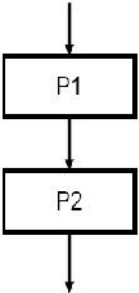
Lambang (2)

Keterangan	Lambang
Percabangan (<i>Decision</i>)	
Pemberian nilai awal suatu variabel (<i>Preparation</i>)	
Memanggil prosedur/fungsi (<i>Call</i>)	

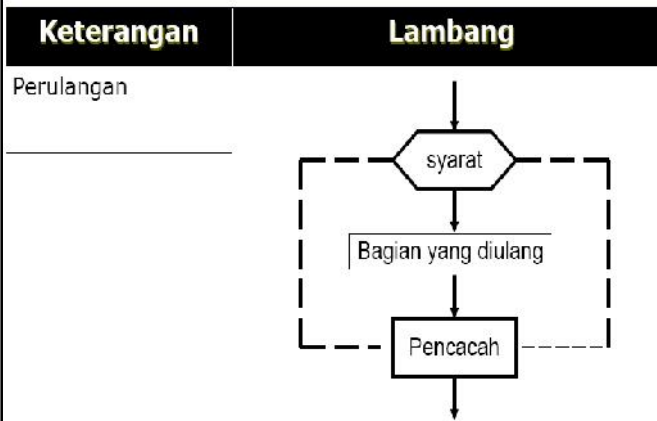
Lambang (3)

Keterangan	Lambang
Connector (di halaman yg sama)	
Off page Connector (halaman lain)	
Dokumen / Multi dokumen	
Harddisk	

Lambang (4)

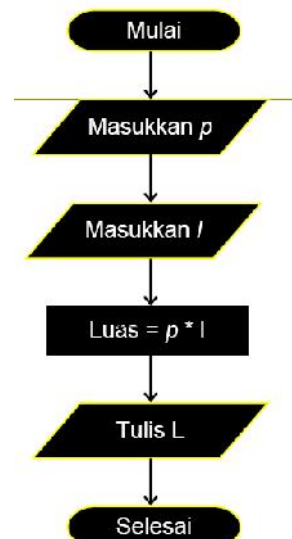
Keterangan	Lambang
Sequence Process	 <pre> graph TD In(()) --> P1[P1] P1 --> P2[P2] P2 --> Out(()) </pre>

Lambang (5)



Contoh Flowchart

- Problem:
Menghitung
Luas persegipanjang
- Algoritma:
1. Masukkan panjang (p)
 2. Masukkan lebar (l)
 3. Hitung luas (L),
yaitu panjang kali lebar
 4. Cetak luas (L)



latihan

Contoh (1) : Menghitung rata-rata tiga buah data

Algoritma dengan struktur pseudocode :

- 1) *input (a, b, c)*
- 2) *Jml = a+b+c*
- 3) *Rerata = Jml/3*
- 4) *Output (Rerata)*

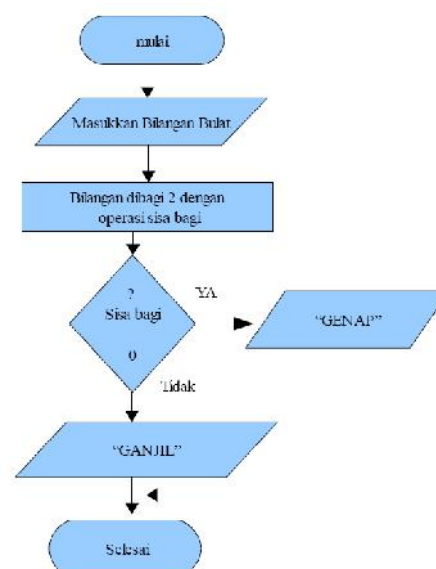
LATIHAN SOAL

1. Buat Algoritma untuk mengecek suatu bilangan positif atau negatif !



Contoh Flowchart

- Problem:
Menentukan
Bilangan ganjil atau
Genap



STRUKTUR DASAR ALGORITMA

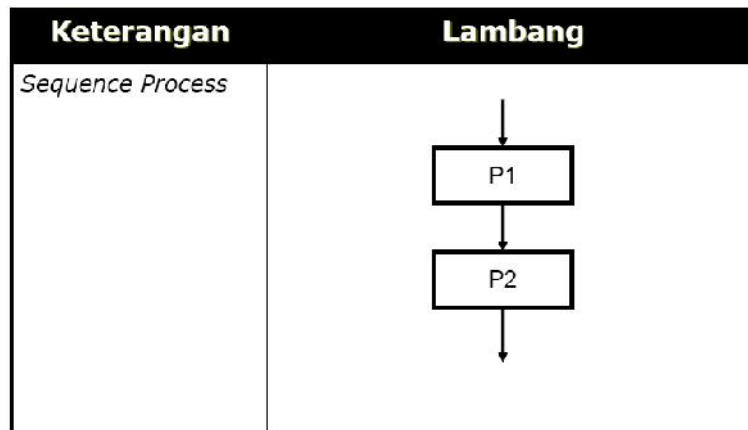
Struktur dasar algoritma :

1. Struktur Runtunan (Sequence Proses)
2. Struktur Pemilihan (Selection Proses)
3. Struktur Pengulangan (Iteration Proses)

Struktur Runtunan

Sebuah runtunan terdiri dari satu atau lebih 'instruksi'. Tiap-tiap instruksi dilaksanakan secara berurutan sesuai dengan urutan penulisannya; sebuah instruksi baru bisa dilaksanakan setelah instruksi sebelumnya selesai dilaksanakan.

Struktur Runtunan



Struktur Pemilihan

Pada struktur ini, jika kondisi terpenuhi maka salah satu aksi akan dilaksanakan dan aksi yang ke dua diabaikan.

Kondisi adalah persyaratan yang dapat dinilai benar atau salah sehingga akan memunculkan 'aksi' yang berbeda dengan 'kondisi' yang berbeda.

Jenis Pemilihan

- ▶ Banyak Kasus
- ▶ Satu/Dua Kasus

Banyak Kasus

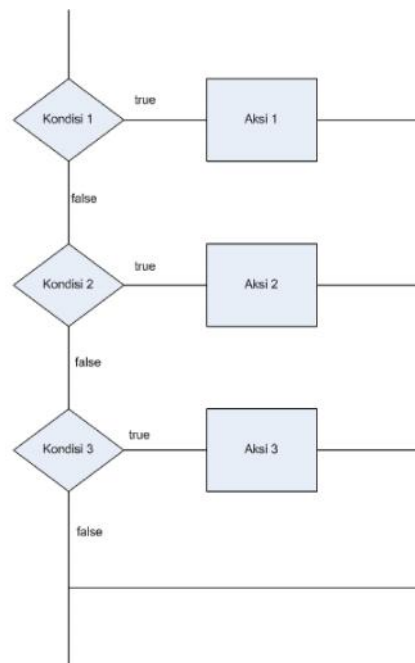
depend on (sesuatu, sesuatu)

<kondisi-1> ; <aksi-1>

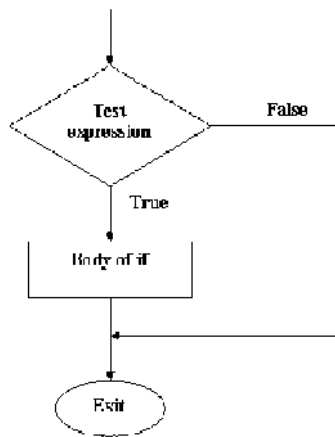
<kondisi-1> ; <aksi-1>

<kondisi-3> ; <aksi-3>

<kondisi-n> ; <aksi-n>

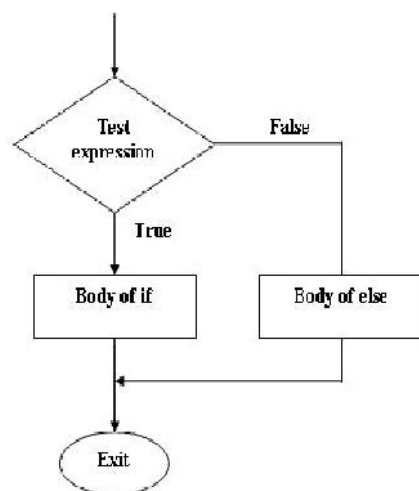


Satu atau Dua Kasus



Notasi algoritmik :
if Syarat then
 Aksi {True}
endif {False}

Struktur Pemilihan



Notasi Algoritma,
IF syarat THEN
 aksi-1 {true}
ELSE
 aksi-2 {false}
ENDIF

Struktur Pemilihan

CONTOH :

Menentukan bilangan terbesar diantara 3 bilangan:

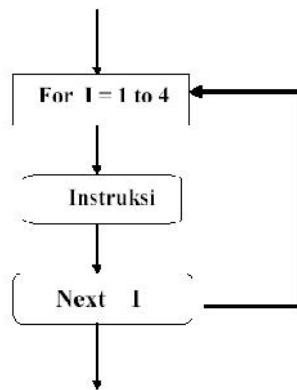
```
'if' x > y 'then'  
'if' x > z 'then'  
  tulis x sebagai bilangan terbesar  
'else'  
  tulis z sebagai bilangan terbesar  
'else'  
  'if' y > z 'then'  
    tulis y sebagai bilangan terbesar  
  'else'  
    tulis z sebagai bilangan terbesar
```

Struktur Pengulangan

Digunakan untuk program yang pernyataannya akan dieksekusi berulang-ulang. Instruksi dikerjakan selama memenuhi suatu kondisi tertentu. Jika syarat (kondisi) masih terpenuhi maka pernyataan (aksi) akan terus dilakukan secara berulang.

Struktur Pemilihan

For-Next



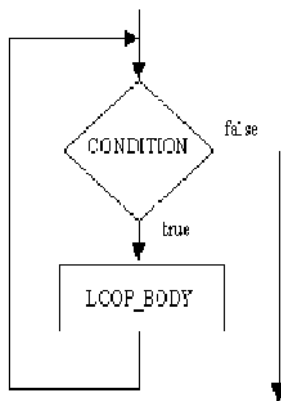
For var=awal **to** akhir

 instruksi-instruksi

Next var

Struktur Pengulangan

While - do



Bentuk umum :

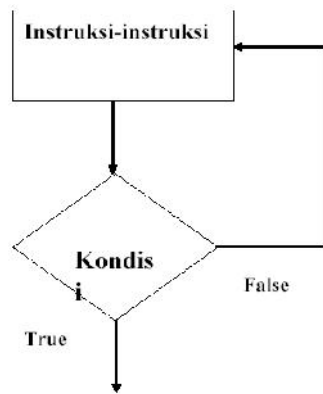
While {kondisi} **do**

 instruksi-instruksi

Endwhile

Struktur Pengulangan

Repeat - Until



Bentuk Umum ;

Repeat

.....
Instruksi

.....
Until (kondisi)

Struktur Pengulangan

Contoh :

Algoritma Cetak_Angka

{mencetak 1, 2, ..., 8 ke piranti keluaran}

Deklarasi :

K: integer

Deskripsi :

$K \leftarrow 1$ {inisialisasi}

while $k \leq 8$ do

write (k)

$k \leftarrow k + 1$

endwhile

Contoh :

Algoritma Cetak_Angka

{mencetak 1, 2, ..., 8 ke piranti keluaran}

Deklarasi :

K: integer

Deskripsi :

$K \leftarrow 1$ {inisialisasi}

repeat

write (k)

$k \leftarrow k + 1$

until $k > 8$

LATIHAN SOAL

1. Buat Algoritma untuk mengecek suatu bilangan positif atau negatif !

1. Buat algoritma untuk menampilkan deret angka berikut :

0
2
5
10
26
37

LATIHAN SOAL

3. Diketahui sebuah algoritma berikut ini :

Deklarasi :

i, m : integer

Deskripsi :

i = 0

m = 0

while i < 9 then

*m = i * i*

cetak m

i = i + 1

endwhile.

Tulis output yang dihasilkan algoritma di atas !

Daftar Pustaka

- ▶ Jajat Sudrajat, PENGANTAR ALGORITMA DAN IMPLEMENTASI BAHASA PASCAL, http://www.google.co.id/url?sa=t&rct=j&q=syarat%20algoritma%20yang%20baik&source=web&cd=1&ved=0CBcQFjAA&url=http%3A%2F%2Fyusufhdc.edublogs.org%2Ffiles%2F2010%2F01%2FPENGANTAR-ALGORITMA_adzet.ppt&ei=K2-FTryKEOqtiQfYuqCSDw&usg=AFQjCNFR-bfs2ySV6zumBAvRI1tOwM-pvw&cad=rja