1 ALGORITMA DAN BAHASANYA

Kita semua telah tahu, bahwa pada hakikatnya sebuah KOMPUTER merupakan alat untuk membantu kita memecahkan masalah. Dalam usaha kita memecahkan masalah tersebut, kita harus dapat mengubah atau menyajikan masalah ke dalam suatu model yang kiranya tepat bagi komputer. Model yang tepat tersebut, pada umumnya adalah suatu model matematika.

Memang tidak mudah bagi kita untuk mendapatkan model matematika yang sesuai bagi suatu masalah sehari-hari. Untuk hal ini diperlukan studi lanjut, baik dalam disiplin Matematika itu sendiri, maupun disiplin Ilmu Komputer ataupun yang lainnya.

Bagi kita yang baru dalam taraf mulai belajar memecahkan masalah dengan komputer, tentunya kita harus memulainya dengan keadaan yang sederhana. Buku kita ini tidak membahas bagaimana suatu masalah dibawa ke suatu model matematika yang rumit, namun membahas bagaimana cara kita memberi instruksi kepada komputer untuk menyelesaikan masalah. Masalahnya sendiri sudah dalam model yang berorientasi ke komputer (atau computer oriented).

Untuk itu, mula-mula kita susun lebih dahulu langkah-langkah instruksi pemecahan masalah. Langkah-langkah instruksi ini secara umum kita kenal sebagai

ALGORITMA. Kemudian, setelah langkah-langkah instruksi, atau Algoritma tersebut selesai kita susun, kita harus menyajikan langkah-langkah tersebut dalam salah satu bahasa yang dikenal oleh komputer. Tentunya penyajian himpunan instruksi yang kita buat tersebut haruslah teratur dan memenuhi berbagai persyaratan yang diperlukan, agar benar-benar nantinya mencapai sasaran seperti yang kita inginkan. Di sini dikatakan, bahwa kita membuat sebuah PROGRAM. Kegiatan kita menulis atau membuat langkah instruksi tersebut, sekaligus menuliskannya dalam suatu bahasa yang dikenal komputer dinamakan PEMROGRAMAN. Dapat dicatat bahwa tingkat yang termasuk agak rawan dalam mencari pemecahan masalah dengan menggunakan komputer ialah pembuatan dan perancangan Algoritma serta struktur data yang tepat. Jadi, sebuah Algoritma pada hakikatnya merupakan suatu Prosedur yang tepat untuk mendapatkan pemecahan masalah menggunakan bantuan komputer serta dengan menggunakan suatu bahasa Pemrograman tertentu.

Algoritma umumnya mudah dibaca oleh manusia sedangkan Program diperuntukkan bagi komputer. Sewaktu merancang sebuah Algoritma, perlu sekali diingat bahwa sebuah komputer hanya mengikuti instruksi dan tidak dapat melakukan sesuatu sebelum diperintahkan. Oleh karena itu, perancang harus menuliskan setiap segi dari permasalahan di dalam Algoritma yang bersangkutan. Bab ini menguraikan tentang syarat-syarat yang harus dimiliki oleh suatu Prosedur pemecahan masalah jika ia akan kita proses menggunakan Algoritma. Salah satu cara penyajian Algoritma adalah dalam bentuk diagram atau bagan alur (flowchart).

1.1 DEFINISI ALGORITMA

Secara pembahasan yang lebih teoritis, kita mengenal pengertian PROSEDUR, SEMI-ALGORITMA, dan ALGORITMA. Namun, secara praktis kita boleh menganggap ketiga pengertian di atas sebagai satu pengertian yang sama, yakni Prosedur atau Algoritma.

Sebuah **Prosedur (yang efektif)** didefinisikan sebagai himpunan hingga instruksi, yang bersifat diskrit dan jelas, serta dapat dijalankan secara mekanik. Untuk pemecahan dengan komputer, pengertian dapat dijalankan secara mekanik dapat diartikan sebagai *dapat dibuatkan Program komputernya*. **Semi-Algoritma** adalah Prosedur yang mampu menghasilkan pemecahan atau solusi masalah, bila solusi memang ada, dan kemudian berhenti. Sedangkan **Algoritma** adalah Semi-Algoritma yang mampu mendeteksi tidak adanya solusi, bila solusi memang tidak ada, dan mampu pula berhenti.

Jadi, singkatnya *Algoritma merupakan suatu himpunan hingga instruksi yang secara jelas memperinci langkah-langkah proses pelaksanaan, dalam pemecahan suatu masalah tertentu , atau suatu kelas masalah tertentu, dengan dituntut pula bahwa himpunan instruksi tersebut dapat dilaksanakan secara mekanik.*

Algoritma boleh dibilang merupakan suatu "resep" untuk memecahkan masalah. Sebuah Algoritma dapat disiapkan pada berbagai tingkat permasalahan. Perangkat

mesin komputer hanya dapat menjalankan suatu instruksi, jika instruksi telah dibuat dalam suatu bahasa yang dimengerti oleh komputer tersebut.

Contoh 1-1

Sebagai contoh yang mudah, berikut ini Prosedur yang dapat kita gunakan ketika kita ingin mengirimkan surat kepada teman, yaitu :

1. Tulis surat pada secarik kertas surat
2. Ambil sampul surat
3. Masukkan surat ke dalam sampul
4. Tutup sampul surat menggunakan perekat
5. Jika kita ingat alamat teman tersebut, maka tulis alamat pada sampul surat. Jika tidak ingat, lihat lebih dahulu pada buku alamat, baru kemudian kita tulis alamat pada sampul surat.
6. Tempel perangko pada surat
7. Bawa surat ke kantor pos untuk diposkan.

Ketujuh langkah di atas merupakan Prosedur yang efektif. Banyak instruksi adalah hingga, masing-masing bersifat diskrit dan jelas, serta dapat kita kerjakan (secara mekanik).

Contoh 1-2

Berikut ini contoh Prosedur untuk menyelesaikan masalah komputasi akar bulat positif dari suatu bilangan bulat (integer) posistif a, yaitu :

1. Baca a
2. Masukkan x sama dengan 1
3. Hitung y sebesar x * x
4. Jika y tidak sama dengan a maka cetak x sebagai akar dari a. Selesai.
5. Tambah nilai x dengan 1
6. Pergi ke langkah 3.

Kita dapat pula membuat sebuah program bagi Prosedur ini. Contohnya adalah program dalam bahasa BASIC berikut ini :

Contoh 1-3

```
10  INPUT A
20  X = 1
30  Y = X * X
40  IF Y = A THEN PRINT X : END
50  X = X + 1
60  GO TO 30
```

Kalau kita masukkan nilai 4 misalnya, jelas akan tercetak nilai 2 sebagai akar dari 4. Namun, bila kita masukkan nilai 5 misalnya, Prosedur akan berlanjut tak berhenti-henti. Memang, untuk A = 5, akar bulatnya tidak ada. Terlihat bahwa Prosedur di atas merupakan sebuah Semi-algoritma; bukan Algoritma. Sebuah Semi-Algoritma selalu dapat disempurnakan, dengan suatu cara/modifikasi, menjadi sebuah Algoritma. Semi-Algoritma contoh kita di atas dengan mudah kita sempurnakan menjadi sebuah algoritma, dengan cara menyelipkan satu instruksi.

```
45 IF Y > A THEN PRINT "TAK ADA AKAR": END
```

pada program BASIC kita yang lalu.

Untuk maksud yang sama, silakan anda memodifikasi Diagram Alur sebelumnya.

Suatu komputer biasanya hanya mampu menerima instruksi dalam bahasa mesin. Bahasa mesin adalah terdiri dari untai 1 dan 0, atau "on" dan "off" yang dikenal sebagai untai binary digit atau bit.

Perancang Algoritma akan menemui kesulitan untuk menyusun instruksi dalam bahasa mesin seperti itu, karena dapat menimbulkan ketidakjelasan dalam menulis pokok-pokok Prosedur yang diinginkan. Namun, untunglah bahwa sekarang ini telah berhasil dibuat Bahasa Pemrograman tingkat tinggi, seperti bahasa BASIC, Pascal dan sebagainya, yang lebih sesuai bagi Pemrogram dalam mengalihkan Algoritma yang telah disusunnya tersebut, menjadi sebuah Program yang benar. Memang, Program dalam suatu Bahasa Pemrograman tingkat tinggi tidak langsung diterima komputer. Kita memerlukan suatu penterjemah, yang dalam hal ini kita kenal seperti Kompilator serta Interpreter, untuk menterjemahkan Program dalam bahasa tingkat tinggi tadi menjadi Program dalam bahasa tingkat rendah, bahasa mesin. Memang terdapat banyak Program yang berisi ribuan instruksi baik dalam bahasa tingkat tinggi maupun dalam bahasa tingkat rendah. Adalah tidak mudah untuk membuat Program siap pakai seperti itu. Kesemuanya itu harus dimulai dengan membuat Program yang sederhana dulu.

Contoh 1-4.

Di bawah ini terdapat sebuah Algoritma yang sederhana berisikan instruksi dalam bahasa Indonesia sehari-hari.

Masalah

Tentukan bilangan terbesar di antara 3 bilangan bulat yang diberikan.

Algoritma secara garis besarnya adalah :

1. Bandingkan bilangan bulat pertama dan kedua, kemudian tentukan mana yang lebih besar di antara mereka.
2. Bandingkan bilangan tadi (bilangan yang lebih besar hasil langkah 1) dengan bilangan ketiga. Bilangan yang lebih besar di sini merupakan bilangan terbesar yang dicari.

Algoritma dengan langkah yang lebih terperinci adalah sebagai berikut :

1. Baca (sebagai input) bilangan yang pertama; sebut sebagai NUM1.
2. Baca (sebagai input) bilangan yang kedua; sebut sebagai NUM2.
3. Bandingkan NUM1 dengan NUM2, kita ambil yang lebih besar; jika kedua bilangan tersebut sama besarnya, dapat kita ambil NUM1, dan sebut bilangan tersebut BESAR.
4. Baca (sebagai input) bilangan ketiga, sebut NUM3.
5. Bandingkan BESAR dengan NUM3 dan pilih yang lebih besar; jika keduanya sama besar, pilih BESAR; dan sebut bilangan tersebut BESAR.
6. Keluarkan, sebagai output, BESAR.
7. Selesai

Untuk jelasnya, mengenai pelaksanaan Algoritma di atas, misalkan ketiga bilangan tersebut adalah 3, 9, 7.

1. Kita baca bilangan yang pertama, yakni 3. Jadi, NUM1 adalah 3.
2. Kita baca bilangan yang kedua, yakni 9. Jadi, NUM2 adalah 9.
3. Kita bandingkan NUM1 dengan NUM2, yakni 3 dengan 9. Kita ambil yang lebih besar, yakni 9, kita sebut sebagai BESAR.
4. Kita baca bilangan ketiga, yakni 7, yang kita sebut sebagai NUM3.
5. Kita bandingkan BESAR dan NUM3, yakni 9 dan 7. Kita pilih yang lebih besar, yakni 9. Jadi, nilai BESAR adalah 9.
6. Kita keluarkan nilai BESAR, yakni 9, sebagai output.
7. Selesai.

CATATAN.

Perhatikan bahwa keputusan yang diambil didalam perancangan Algoritma dibuat sesuai pilihan dari perancang bersangkutan (misalnya, langkah ke-4 dapat dibuat mendahului langkah ke-3; jika dua bilangan tersebut sudah sama di langkah ke-3, pilih salah satu). Walaupun demikian, langkah-langkah lain tidak dapat diubah tanpa mengganggu kesatuan Algoritma tersebut (misalnya, langkah ke-4 harus selalu mendahului langkah ke-5).

CONTOH 1-5.

Berikut ini perluasan dari persoalan pada Contoh 1-4 di atas.

Masalah:
Tentukan Bilangan terbesar, di antara N buah bilangan yang diberikan, dengan N > 2.

ANALISIS:
Karena Algoritma harus dapat diaplikasikan pada setiap N > 2, N harus menjadi parameter. Oleh karenanya, besarnya N harus menjadi salah satu input untuk Program, bersama dengan bilangannya sendiri.

ALGORITMA:
1. Input (masukkan) N.
2. Input (masukkan) bilanganpertama; sebut NUM1.
3. Input (masukkan) bilangankedua; sebut BESAR.
4. Buatlah pencacah atau counter untuk menghitung banyaknya bilangan yang sudah dibaca, sebut pencacah tersebut HITUNG. Isilah sebagai harga awal dari HITUNG adalah 2.
5. Bandingkan NUM1 dengan BESAR, jika NUM1 lebih besar daripada BESAR, tempatkan BESAR = NUM1.
   Jika HITUNG = N, output adalah BESAR, kemudian berhenti. Jika HITUNG belum sama dengan N tambahkan 1 pada nilai HITUNG. Masukkan bilangan berikutnya sebut NUM1, kemudian laksanakan lagi langkah ke-5 ini.

CATATAN:

Di sini tidak ada maksud untuk menyimpan bilangan tersebut di dalam memori. Kita tidak memerlukan bilangan-bilangan lain yang bukan bilangan terbesar.

Silakan Anda laksanakan Algoritma pada Contoh 1-5 tersebut di atas untuk 7 buah bilangan berikut ini :

11 5 2 6 4 13 9.

Setelah itu, laksanakan untuk data yang Anda buat sendiri dan lihat ketepatan hasilnya.

1.2 Sifat-sifat Algoritma

Prosedur yang tidak memiliki sifat seperti tertera di dalam subbagian di bawah ini, bukanlah merupakan sebuah Algoritma, dan tidak dapat membuahkan hasil yang diinginkan jika Program penyajian dari Prosedur seperti itu dimasukkan ke dalam komputer.

1.2.1 Banyaknya langkah instruksi yang hingga

Pelaksanaan sebuah Algoritma yang terprogram harus dapat diselesaikan setelah pelaksanaan sejumlah langkah operasional yang hingga. Jika tidak demikian, kita tidak bisa mengharapkan bahwa pelaksanaan tersebut dapat membuahkan suatu hasil.

Contoh soal yang terdapat di dalam Contoh 1-4 di atas hasilnya didasarkan pada sifat tersebut. Di sini setiap langkah hanya dilaksanakan satu kali saja. Di dalam Contoh 1-5, langkah ke-5 akan diulang sebanyak N-1 kali. Karena N bersifat hingga; maka dapat ditarik kesimpulan bahwa pelaksanaan Prosedur tersebut akan selesai setelah dilaksanakan dalam beberapa langkah yang hingga. Kita perlu pula memperhatikan hal-hal sebagai berikut :

(a) Jumlah langkah yang sesungguhnya dibutuhkan, sebenarnya tergantung dari perincian isi (detail) dari Algoritma yang bersangkutan. Di dalam contoh 1-5 di atas, langkah ke-5 dapat dipecah ke dalam beberapa langkah. Sementara itu langkah pertama dan kedua dapat digabungkan. Jumlah langkah yang dibutuhkan akhirnya adalah sesuai dengan jumlah instruksi yang dilaksanakan oleh mesin. Walaupun demikian, hal tersebut juga tergantung dari model komputer yang dipakai.

(b) Jumlah langkah operasional yang dilaksanakan biasanya jarang sesuai dengan jumlah langkah yang terdapat di dalam Algoritma (atau jumlah instruksi dalam Program). Jumlah langkah yang benar-benar dilaksanakan sesuai Program tergantung dari data masukan (input) dan tidak selalu dapat dipastikan sebelumnya.

(c) Diperlukan suatu Algoritma yang menuju kepada suatu Program dan dapat terpakai dalam waktu cukup lama, katakanlah 100 tahun. Hal yang pokok dari metode Algoritma ialah terdiri dari ulangan langkah-langkah yang sama, mungkin dengan beberapa modifikasi, seringkali dilaksanakan selama suatu waktu tertentu. Program yang panjang sebenarnya tidak dapat memberikan petunjuk tentang kebutuhan waktu pelaksanaannya. Di dalam contoh 1-5 terlihat bahwa satu langkah (ke-5) dilaksanakan N-1 kali.

1.2.2 Tidak ada keraguan (harus jelas)

Penulisan dari setiap langkah yang terdapat di dalam sebuah Algoritma harus memiliki arti yang khusus atau spesifik. Penulisan langkah bagi komputer dapat berbeda dengan penulisan bagi manusia. Manusia akan mudah mengerjakan Algoritma yang terdiri dari simbol (misalnya memakai kode tertentu atau Diagram Alur); sedangkan komputer membutuhkan sebuah Algoritma dengan kode yang dituangkan ke dalam Program. Hal tersebut berarti bahwa setiap kali suatu Algoritma dilaksanakan dengan data masukan yang sama, maka akan diperoleh hasil yang sama pula.

1.2.3 Batasan dari rangkaian proses

Rangkaian proses yang berisi langkah-langkah dari suatu Algoritma yang akan dilaksanakan, harus ditetapkan dengan pasti. Sebuah Algoritma harus memiliki instruksi dasar tertentu (yang spesifik) dan setiap instruksi harus memiliki unsur pelaksana yang memproses data masukan. Di dalam spesifikasi secara algoritmik, termasuk di sini spesifikasi Program, instruksi dilaksanakan dari atas ke bawah, kecuali apabila ada ketentuen lain, seperti alih kendali bersyarat ataupun tidak bersyarat. Dalam Contoh 1-5, langkah ke-5 dapat dilaksanakan berulang-ulang kali tergantung dari nilai N-nya.

1.2.4 Batasan dari Input dan Output

Input merupakan data yang dimasukkan ke dalam Algoritma. Input tersebut harus sesuai dengan jenis Algoritma yang bersangkutan. Di dalam Contoh 1-4, terlihat bahwa Algoritma tersebut memerlukan tiga input. Di dalam Contoh 1-5 Algoritma dirancang untuk menerima N+1 input. Di dalam kedua contoh tersebut input harus terdiri dari integer (bilangan bulat). Output merupakan hasil yang dikeluarkan oleh komputer untuk kepentingan pihak di luar komputer sebagai hasil dari pelaksanaan Program. Algoritma haruslah menghasilkan suatu output (jika tidak, lalu apa gunanya ?). Algoritma yang terdapat di dalam contoh 1-4 dan 1-5 masing-masing memiliki satu output.

1.2.5 Efektivitas

Instruksi dari sebuah Algoritma dapat memerintahkan komputer agar hanya melaksanakan penugasan yang mampu dilaksanakannya saja. Komputer tidak dapat melaksanakan instruksi, jika informasinya tidak lengkap atau jika hasil dari pelaksanaan perintah tidak diberi batasan yang jelas.

Jika di dalam Contoh 1-5 langkah pertama dihilangkan, maka Algoritma tersebut tidak efektif lagi, karena terjadi kekurangan informasi. Di sini pada langkah ke-5 HITUNG tidak dapat dibandingkan dengan N yang tidak diketahui nilainya.

Juga, jika misalnya suatu instruksi Algoritma memerintahkan untuk membagi suatu integer dengan 0, maka hasilnya tidak dapat didefinisikan dan tidak jelas .

1.2.6 Batasan ruang lingkup

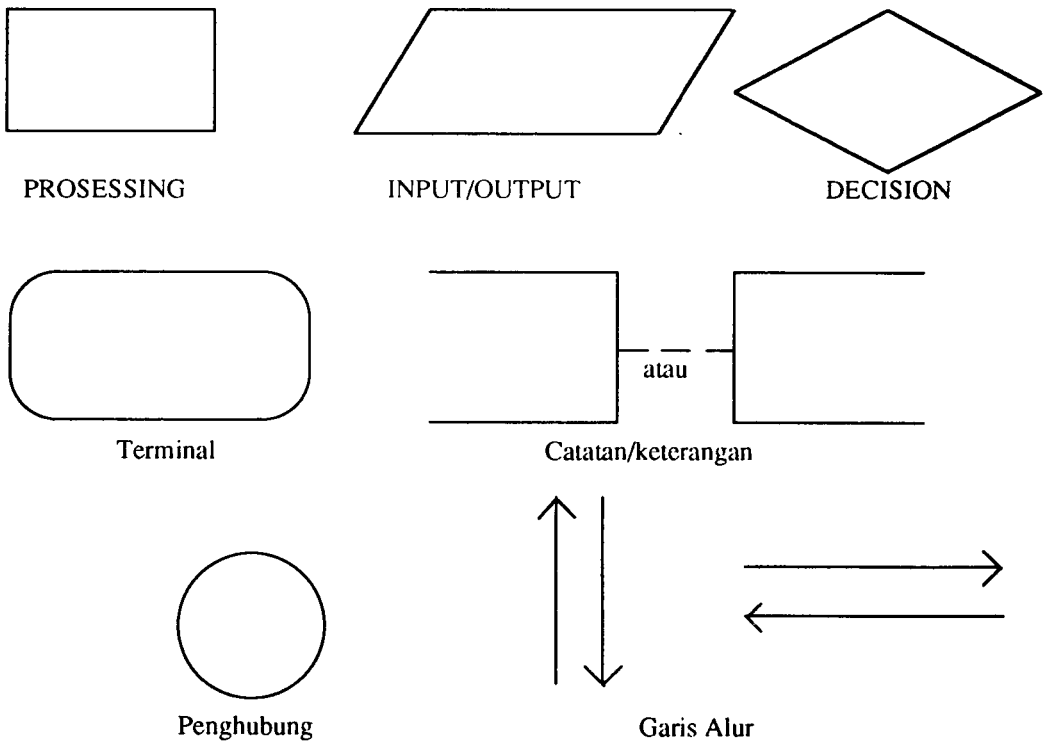
Sebuah Algoritma adalah diperuntukkan bagi suatu masalah tertentu. Susunan input harus ditentukan lebih dahulu. Susunan tersebut menentukan sifat umum dari Algoritma yang bersangkutan. Ruang lingkup Algoritma di dalam Contoh 1-5 bersifat lebih luas daripada Algoritma di dalam Contoh 1- 4. Di sini Algoritma pada contoh 1-4 merupakan suatu hal khusus dari Algoritma pada contoh 1-5 untuk N = 3.

1.3 DIAGRAM ALUR (FLOWCHART)

Pada buku ALGORITMA DAN PEMROGRAMAN jilid 1, kita telah banyak sekali membahas tentang Diagram Alur ini, yang merupakan salah satu cara penyajian Algoritma. Namun, tak ada salahnya bila dalam kesempatan ini kita bahas lagi secara singkat.

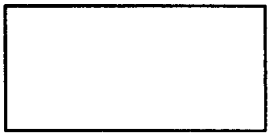
Berhubung komputer membutuhkan hal-hal yang terperinci, maka Bahasa Pemrograman bukan merupakan alat yang boleh dikatakan baik untuk merancang sebuah Algoritma awal.

Alat yang banyak dipakai untuk membuat Algoritma adalah Diagram Alur. Diagram Alur dapat menunjukkan secara jelas arus pengendalian Algoritma, yakni bagaimana rangkaian pelaksanaan kegiatan. Suatu Diagram Alur memberi gambaran dua dimensi berupa simbol-simbol grafis. Masing-masing simbol telah ditetapkan terlebih dahulu fungsi dan artinya. Mereka dipakai untuk menunjukkan berbagai kegiatan operasi dan jalur pengendalian. Seperangkat simbol Diagram Alur terdapat di dalam gambar 1-1. Enam dari simbol tersebut merupakan denah tetap (juga disebut kotak) dengan aneka macam bentuk. Kotak tersebut jika dipakai di dalam sebuah Diagram Alur, akan diisi dengan kata tertentu. Simbol lainnya menjadi alur proses yang akan menentukan rangkaian penugasan yang harus dilaksanakan.



GAMBAR 1-1. Simbol Diagram Alur.

Simbol tersebut di atas mempunyai arti sebagai berikut :



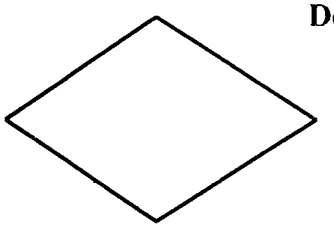
Prosesing :

satu atau beberapa himpunan penugasan yang akan dilaksanakan secara berurutan.

Input/Output :

data yang akan dibaca dan dimasukkan ke dalam memori komputer dari suatu alat input atau data dan harus melewati memori untuk dikeluarkan dari alat output.

Decision (keputusan) :

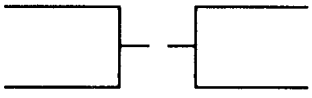


ada dua alternatif yang dapat ditentukan untuk melaksanakan jalur Diagram Alur. Jalur yang harus diikuti dipilih pada saat pelaksanaan Algoritma dengan mengadakan percobaan apakah langkah-langkah yang ditetapkan di dalam bagan sudah terpenuhi atau belum

Terminal :

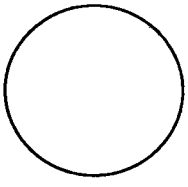
tampil pada awal Diagram Alur (berisi kata "Start") atau pada akhir proses (berisi kata "Stop").

Annotation :



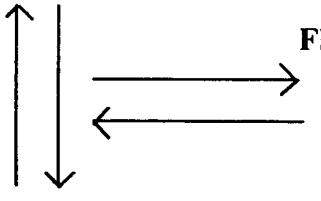
berisi catatan supaya mudah mengerti isi/tujuan Algoritma atau uraian data yang akan diproses.

Konektor :



tanda untuk memisahkan Diagram Alur menjadi beberapa bagian (bersambung ke tempat/halaman lain). Ditulis di tempat/halaman untuk menyambung bagian Diagram Alur yang terputus.

Flowline :

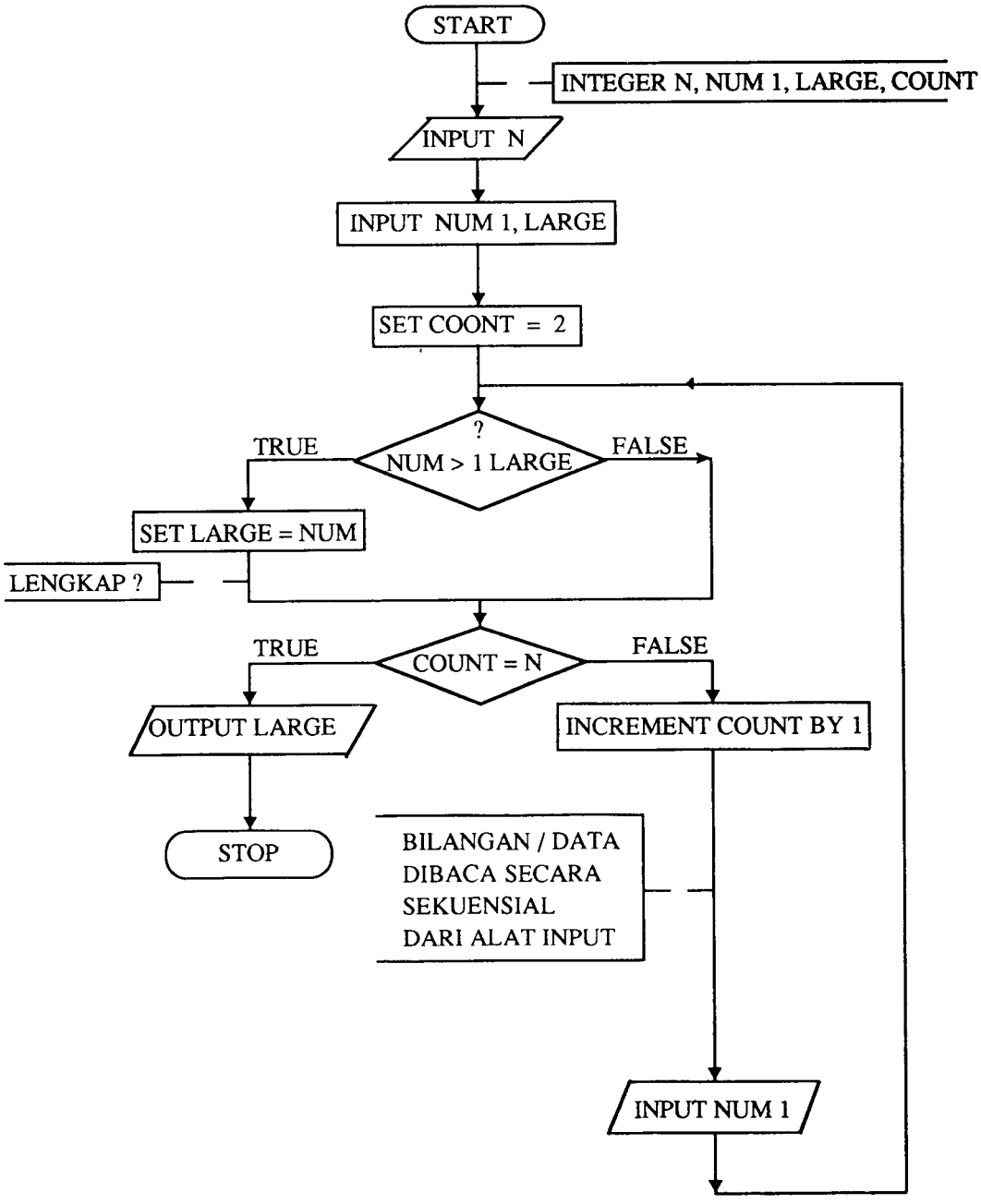


menunjukkan bagan instruksi selanjutnya.

Gambar 1 – 2

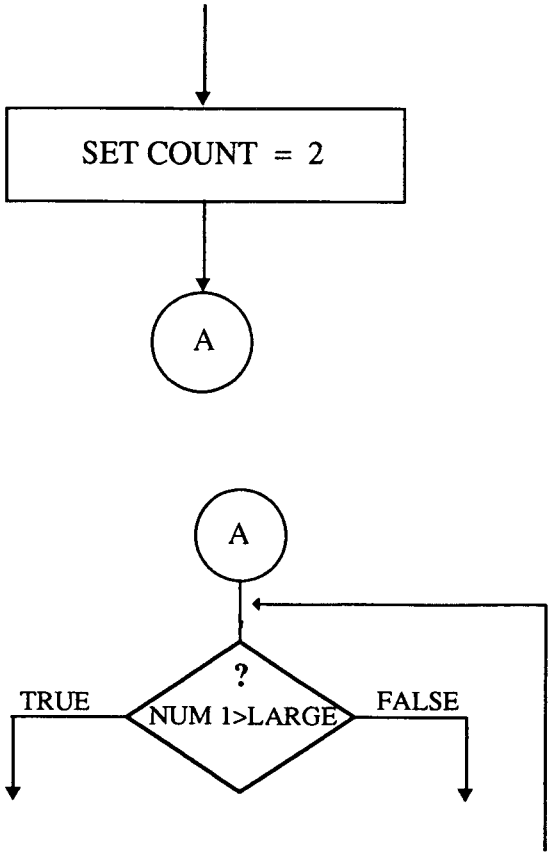
Contoh 1-6

Diagram alur dari Algoritma pada Contoh 1-5 terdapat di dalam gambar 1-3.



GAMBAR 1-3

Jika suatu Diagram Alur ingin dipecah menjadi dua bagian di tempat yang diberi tanda X, maka Konektor harus dipakai seperti terlihat pada Gambar 1-4.



GAMBAR 1-4.

Diagram alur memberikan petunjuk kepada pembaca agar lebih mudah mengikuti logika dari sebuah Algoritma, dibandingkan dengan apabila membaca uraian di dalam bahasa Indonesia sehari-hari. Mengingat bahwa tingkat kerincian dari sebuah Algoritma tergantung kepada perancang bersangkutan, Diagram Alur cocok sekali untuk metode perancangan seperti yang akan diuraikan di dalam Bab 3, yakni bahwa strategi umum dari suatu Algoritma ditetapkan dari awal, dan diperbaiki perlahan-lahan. Untuk membuat Algoritma dengan cara lain, anda dipersilakan membaca pembahasan di dalam bab 3, yakni menggunakan Kode-Tulisan atau Kode-Semu ataupun Pseudolanguage, Pseudocode, Methalanguage atau Bahasa Hipotetik.